On-line Self-Tuning of PI Controllers for High Performance PMSM Drives

Francesco Parasiliti, MarcoTursini, Daqing Zhang
Department of Electrical Engineering, University of L'Aquila
67040 - Poggio di Roio, L'Aquila, Italy
Phone (39) 862-434446 Fax (39) 862-434403 E-mail: tursini@ing.univaq.it

Abstract - **In this paper, a new and practical gain self-tuning method for PI controllers has been formulated and implemented, using the speed control of a Permanent Magnet Synchronous Motor (PMSM) drive system as testbed. While the novel strategy enhances the performance of traditional PI controller greatly and proves to be a completely model-free approach, it also preserves the simple structure and features of PI controllers. The essential idea is: 1). According to the system dynamics to step variation, define a performance index to evaluate the system response. 2). Based on the monotonous relationship between the performance index and an intermediate PI gain parameter, the intermediate PI gain parameter is estimated with a modified binary search algorithm in the direction of improving the performance index. 3). Finally PI gains are calculated and renewed according to the estimated intermediate gain parameter. Experiments have been thoroughly carried out to test the proposed method under different conditions. Besides being simple and easy to implement for real-time applications, the proposed method also possesses the features such as versatility, stability and effectiveness.**

I. INTRODUCTION

Nowadays, the vector control technique has made it possible to apply the Permanent Magnet Synchronous Motors (PMSM) in high performance industrial applications where only DC motor drives were previously available. Since most of the PMSM drive systems in industrial applications are controlled with PI and PID controllers, so there exists a growing demand to optimally set the PI and PID parameters on-line according to the varying operating conditions so that even better system performances can be achieved.

In the last decade, a lot of studies have been made in the area of gain self-tuning control in order to improve the performance of traditional PI and PID controllers. Roughly speaking, there are two group of thoughts to auto-tune the PI and PID gains so that the controllers can adapt to the varying conditions: model-based analytical method and model-free method [1].

For the model-based analytical method, the gain self-tuning law is based on an assumed mathematical model and some assumed conditions, its effectiveness depends naturally on the accuracy of the model and objectivity of the assumed conditions [2][3]. Because of the system non-linearity, disturbance, noises etc., very often a lot of assumptions in deriving the model are violated in real applications. When the estimated model is inaccurate in some conditions, the model-based method may not sound solid, and thus leads to degraded controller performance.

Model-free gain tuning strategy emerged a little later than its counterpart. Firstly expert systems were built to perform the on-line controller tuning [4][5][6], then fuzzy gain tuning methods were proposed based on the related experts' gain setting experiences [7][8][9][10][11], the underlying idea is just to emulate operators or experts' actions during the gain tuning procedure.

On-line gain tuning can be implemented with two kinds of information: step-based information and cycle-based information. The tuning strategies can be naturally classified into two categories, that is:

- Step-based gain tuning: using the information derived from each sampling step, tune the gains accordingly. In this case, the concerned step information include the error, the error derivative, the sum of the precedent errors, etc.
- Cycle-based gain tuning: using the information got from each step reference change, tune the gains so as to improve the performance of the step response. In this case, the common used cycle parameters include overshoot, rising time, etc.

Up till now, only simulation results have been reported for step-based gain-tuning controllers, while the existing commercial gain-tuning controllers are almost based on cycle information. As for the known self-tuning controllers, most part of them are verified in slow process control, few general and practical tuning methods have been reported for quick process control such as the case of electrical drives. In this paper, a new model-free gain tuning approach is proposed and implemented, the proposed approach is based on the cycle information and derived from a subtle analysis to the speed transient responses of a PMSM drive. The essential idea of the PI gain self-tuning mechanism is:

- For each cycle, according to the system dynamics to step variation, define a performance index to evaluate the system response.
- Based on the monotonous relationship between the performance index and an intermediate PI gain parameter, the intermediate PI gain parameter is

0-7803-3544-9/96 $5.00 © 1996 IEEE

estimated with a modified binary search algorithm in the direction of improving the performance index.

- PI gains are calculated and renewed according to the estimated intermediate gain parameter in each cycle.

With the above idea, a novel and practical gain self-tuning method for PI controllers has been implemented, representative experiments have been carried out to verify and evaluate the proposed method. The remarkable features of this self-tuning controller include: first, the traditional controller structure is preserved so that the existing systems can be easily upgraded just by adding a simple supervised level; second, the knowledge gained in the past in both the controller design and tuning can be fully exploited; third, the proposed method doesn't depend on a precisely defined mathematical model. While it is simple and easy to implement for real-time applications, it also possesses the characteristics such as versatility, stability, and effectiveness.

II. Formulation Of The PI Gain Tuning Idea

Fig. 1 shows the block diagram of a typical speed control scheme for high performance PMSM drive using standard PI controller.

In the figure R_{fr} is the speed reference filter, R_n refers to the PI speed controller and the cascade part (S^R_{em} + S_m) represents the controlled system, where S^R_{em} is the internal torque control loop (electromagnetic controlled system) and S_m the mechanical system. The scheme also shows the load torque disturbance m_l.

Standard design of the PI speed controller requires the knowledge of the characteristics of the mechanical system, i.e. the motor-load inertia.

Supposing the moment of inertia assumed in the controller is different from the actual one, the dynamics of the speed control loop in Fig.1 can be expressed in the Laplace domain by (1) according to [1].

$$\frac{n}{n_r} = \frac{1}{1 + s4T_{pe} + s^2 8T_{pe}^2 \frac{J_m}{J_c} + s^3 8T_{pe}^3 \frac{J_m}{J_c}} \qquad (1)$$

where n and n_r are the speed and the speed reference, T_{pe} is the equivalent time constant of the electromagnetic controlled system, J_c and J_m are the per-unit (pu) moment of inertia assumed in the controller and in the actual motor-load system, respectively.

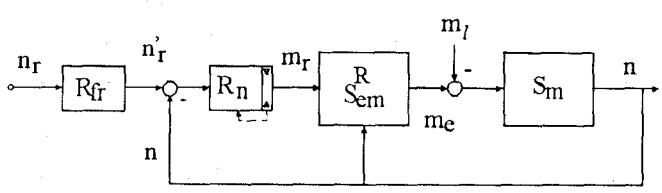

Fig.1. Speed control scheme of high performance PMSM drive

If J_c is assumed to be equal to J_m, the optimal response can be obtained with a rising time of about 7.5 T_{pe} and an overshoot of about 7.5%. If the system inertia J_m varies as the load changes, the optimal response doesn't hold any more. Fig. 2 demonstrates three typical system responses obtained from (1) corresponding to the cases J_c=J_m=1, J_c>J_m (J_c=10, J_m=1) and J_c<J_m (J_c=1, J_m=5), where the system is assumed to have T_{pe}=5.5 ms and the base value for inertia is $22.0 \cdot 10^{-5}$ kgm^2.

Obviously, the responses can be grouped in three categories. That is,

- When J_c = J_m, optimal response with target overshoot is obtained.
- When J_c < J_m, the response has an overshoot greater than the target one.
- When J_c > J_m, the response comes with an overshoot smaller than the target one or without overshoot.

For a practical PMSM drive system, in order to tackle the problems due to limited current toleration ability of the inverter, component saturation phenomena and the controller wind-up, a modified PI controller with current saturation and integral component correction mechanism has been applied to improve the system dynamics. With these modifications, system nonlinearities are introduced such that the actual system responses differ from the ideal case. Fortunately, the following conclusions hold in any case:

- For a fixed J_m in a certain condition, the system response can be monotonously changed to approach any "optimal" response with desired target overshoot by adjusting J_c monotonously.
- For a fixed J_c in a certain condition, if J_m increases or decreases suddenly, the system response holds the same monotonous overshoot changing characteristics. By adjusting J_c in the direction to approach J_m, one can always obtain the "optimal" response with desired target overshoot.

With the above observation, a solution to improving the system response can be formulated by adjusting J_c so as to approach the target overshoot, with the idea that J_c is the

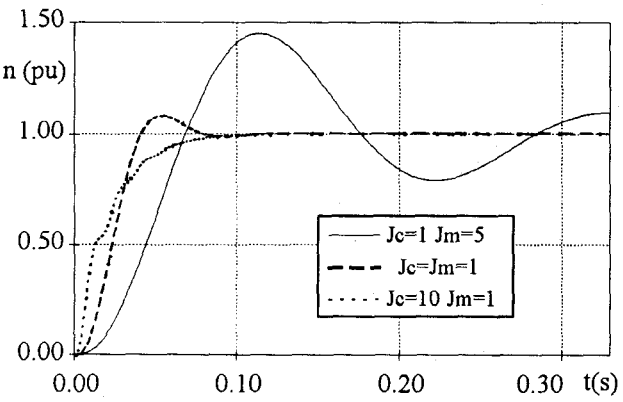

Fig. 2. Speed responses with different system and controller inertia

predominant parameter associating the PI gains with the system responses, the so-called "intermediate PI gains parameter".

If we define the performance index IE as the difference between the effective overshoot and the ideal one, then it is easy to formulate the gain tuning idea as follows:

- when IE=0, the overshoot is what required, so the value J_c is reasonable ;
- when IE>0, the overshoot is too big, one should increase J_c;
- when IE<0, contrary to the above case, one should decrease J_c;

For real applications, however, the target overshoot needn't to be defined in a very precise way. Actually, if the resulted response falls into an desired range, we can say that the controller works well. So in real implementations, a target overshoot range can be adopted instead of the single target overshoot value. The performance index IE can be thus defined as:

$$IE = OV_a - OV_t$$

where OV_a and OV_t represent the actual and target overshoot respectively. Since the target overshoot corresponds to a range $[OV_{min}, OV_{max}]$, so the OV_t can be further defined according to the following algorithm:

if $OV_a > OV_{max}$ *then* $OV_t = OV_{max}$
else if $OV_a < OV_{min}$ *then* $OV_t = OV_{min}$
else $OV_t = OV_a$

With the above definition, it is found that when the actual overshoot of the system response lies in the target overshoot range $[OV_{min}, OV_{max}]$, then IE = 0; while it is outside the target range, IE takes either a positive or negative value, depending on that the actual overshoot is bigger than the upper boundary of the range or smaller than the lower boundary of the range. Thus the above tuning algorithm remains the same as:

if IE=0 *then* Stop tuning, J_c is suitable
else if IE>0 *then* Decrease J_c
else if IE<0 *then* Increase J_c

III. Binary Search Gain Tuning Method

Knowing the tuning direction, the next question is: to what extent should the J_c be tuned in each step ? Since for a certain PMSM drive system, the moment of inertia of the system J_m should have a deterministic variation range $[J_{mmin}, J_{mmax}]$, with the boundary values corresponding to the no-load and with the largest inertia conditions, respectively. So if J_c can be tuned properly in this range, the optimal speed response is expected to obtain. As J_c has a monotonous relationship with defined IE when J_m and the other parameters maintain constant, so the J_c tuning problem is essentially a matching or searching problem in the J_m variation range in the direction of making IE approach zero. In this way, the binary search method in the searching theory can be adopted for the above J_c tuning problem.

The binary search method is intended to search a data in an ordered sequence. Its rough idea is, first, take the data in the middle of the ordered sequence and compare it with the searching data; if the searching data matches the middle one, then the data is found and the searching procedure ends successfully; if the searching data is bigger than the middle one, then continue the above procedure in the upper half of the sequence; if the searching data is smaller than the middle one, then continue the above procedure in the upper half of the sequence.

Borrowing the above idea, the J_c tuning problem then becomes searching a certain value (fixed as the system moment of inertia is fixed in a certain moment) in the sequence of J_c with IE as a justice mark. Assume J_{cmax} and J_{cmin} are the possible maximum and minimum values of J_c respectively, the applied binary search algorithm is: first, take the middle value J_{cave} in range $[J_{cmin}, J_{cmax}]$, in the coming cycle if IE = 0, then J_{cave} is chosen as the desirable imposed inertia for the controller and one gain tuning procedure converges successfully; if IE < 0, then in the next cycle, the middle value in the lower half of the J_c range is tested as the possible guess of the right system inertia, e.g. in range $[J_{cmin}, J_{cave}]$; in the opposite, try the middle value in the upper half of J_c as indicated by the range $[J_{cave}, J_{cmax}]$. The practical version of the modified algorithm applied in this experiment is described in Fig. 3.

In real applications, however, a problem may arise, e.g., when the tuning and searching algorithm is in process in a certain intermediate range, there may come a sudden change in load parameters so that any J_c in that range cannot make IE = 0, in this case with the pure binary search method the algorithm may never converge. Another problem is concerning about the system noise. Since the calculation of the performance index IE is based on the right measurement of the system speed, so the noise in the real system may influence the acquisition of the correct IE information and thus results in improper tuning of the PI gains.

In order to solve the above two problems, the following practical techniques have been adopted:

a. Use filters to smooth the sampling values so as to tackle the system noise.

b. Incorporate a limit counter in the binary search algorithm to guarantee the system stability when there is a sudden change in the system parameters. With this solution, when the system doesn't converge in a certain number of cycles, all the parameters are reset to initial values so that the gain parameter can be tuned from beginning in the full range. In this way, the algorithm is assured to converge all the time.

IV. On-Line Self-Tuning Implementation: Experimental Set-Up And Results

The block diagram and the experimental set-up of the PMSM drive is shown in Fig. 4. The whole system consists of a host personal computer, a SIEMENS SAB80C186 microcontroller, a hysteresis analogic current controller, a voltage source inverter and a three phase PMSM equipped with an incremental position encoder. The motor is controlled according to the principle of the orientation on the rotor flux (control in the rotor d,q reference frame).

As it is known, a linear relationship between current amplitude and torque is guaranteed by setting at zero the reference of the direct axis current. With such an assumption the reference of the quadrature current can be obtained by the speed controller. A d,q $\Rightarrow$ abc transformation gives the current references in the natural three phase reference system, needed for the hysteresis regulators.

Match J_m in the ordered sequence $J_c=(J_{cmin},\ldots, J_{cmax})$

Initialisation: $J_c:=(J_{cmin}+J_{cmax})/2;$

Reset: $J_{c1}:=J_{cmin};\ J_{c2}:=J_{cmax};$

Execution loop:

BEGIN

IF J_c makes IE in target range THEN J_c is desirable, stop tuning and reset

ELSE IF J_c makes IE bigger than target range THEN

BEGIN

$J_{c1}:=J_c;$

$J_c:=(J_{c1}+J_{c2})/2;$

END

ELSE IF J_c makes IE smaller than target range THEN

BEGIN

$J_{c2}:=J_c;$

$J_c:=(J_{c1}+J_{c2})/2;$

END

END

Fig. 3. Binary search algorithm

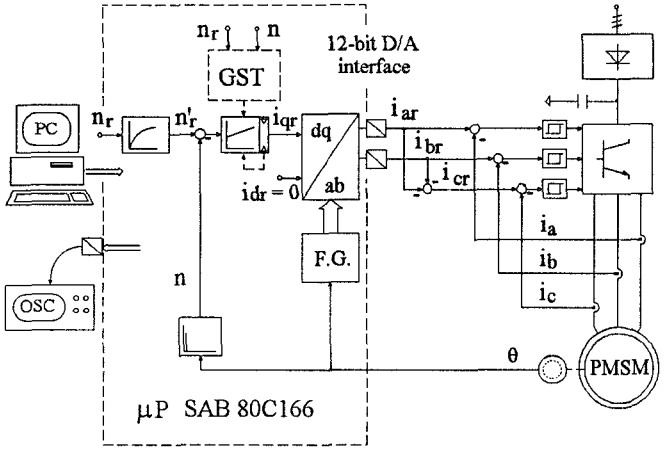

Fig. 4. Block diagram and experimental set-up of the PMSM drive

The speed controller is provided with current saturation. A correction of the integral component during saturated transients is also provided, in order to improve system dynamics.

The host computer is connected with the microcontroller through its serial ports. It is in charge of the real-time command and state monitoring of the drive system. The microcontroller, on the other hand, is used to do all the real-time data processing involved in the system control and gain tuning.

Since the proposed gain self-tuning (GST) scheme is based on the cycle information, so cyclic reference changes are needed to do the gain tuning. A cycle is composed of a positive reference change followed by a negative one. First the target overshoot range needs to be determined and issued to the microcontroller through the host. When the microcontroller receives the command to start the gain-tuning process, at each positive reference variation it computes the effective overshoot, calculates the value of the performance index IE and the gain-tuning parameter J_c; the gains of the PI controller are renewed at the next negative reference variation, based on the lately estimated J_c. In this way, the PI gains are tuned according to the overshoot obtained by the positive step changes only. This is quite reasonable because the values of the overshoot for positive and negative reference changes under the same operating condition is different. This sort of implementation can assure the required stability to the self-tuning process.

Details on the implementation of the control algoritmh on the SAB 80C166 microcontroller are shown in Fig. 5. Two interrupt released procedures (tasks) are used, the "reset task" and the "control task". The "reset task" is run at startup. After the needed initializations this procedure enters in a never-ending loop (main loop) which implements the generation of the phase current references and their outputs through the D/A converters. The actual rotor position

elaborated from the encoder signals is read from the microcontroller's pulse counting unit. Then the d,q ⇒ a,b

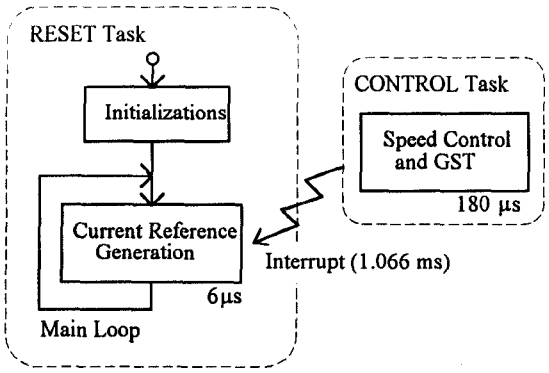

Fig. 5. Control algorithm architecture

transformation is calculated, according to the actual torque current reference. The sinusoidal functions needed for such transformation are obtained by a look-up-table addressed with the actual position value. This assures the loop the fast calculation (about 6 μs) required for vector control. The main loop is synchronously interrupted by a second procedure ("control task") which implements the PI speed regulator and the gain self-tuning algorithm. This procedure reads the speed reference received from the host PC and updates the torque current reference used in the main loop. If the gain-tuning is enabled and a speed reference change is detected then the GST algorithm is entered and the PI controller's parameters are updated. The speed sampling time is equal to the period of the interrupt releasing the speed control task, about 1.066 ms in this application. All the control program is written in assembler, using single precision 16 bits arithmetics. The calculation time of the speed control and GST procedure is environ 180 μs.

In order to verify the effectiveness of the proposed method, representative tests have been performed under different setting points and extreme initial conditions. In all the experiments reported below the current limit is ±1.5 pu, while the target overshoot range is chosen as [5.0 %, 7.5 %] and the J_c serching range is [1 pu, 8 pu]. Fig. 6 shows the speed responses to different working points with the binary search tuning method in one extreme case, the tested condition is under no-load J_m=1 pu, speed reference n_r=0.25, 0.5, 0.75, and 1.0 pu. The initial value of the controller inertia J_{co} is set to 6 pu in order to make the initial speed responses have very small overshoots and demonstrate the convergence capability of the method starting from the too small overshoot extreme case. Putting the corresponding overshoots of speed responses of Fig. 6 in Fig. 7, it can be shown that despite different overshoots are related to different working points initially, all the system responses converge quickly to the target range.

Fig. 8 reports the speed variations to different working points over the tuning cycles in another extreme case, that is, the same four setting points are tested under a load condition with J_m= 6 pu but the initial value of the controller inertia J_{co} is set to 1 pu in order to make the initial speed responses have big overshoots and demonstrate the convergence capability of the method starting from the too big overshoot extreme case.

It can be seen from Fig. 8 that although the convergence rate varies from case to case due to the non-linearity of the drive system, in all the cases the speed responses converge quickly to the target overshoot.

Fig. 9 resumes the variations of the gain tuning parameter J_c over the tuning cycles for the cases reported in Fig. 6 and Fig. 8, at the same speed 0.75 pu. It can be noticed that the final values of the controller inertia are different from the respective actual ones altough the tendency is to move towards the right value.

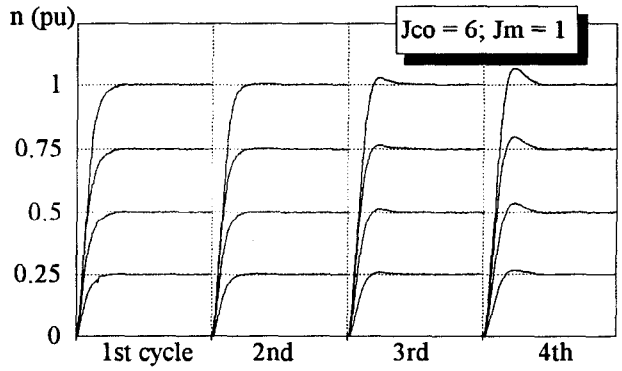

Fig. 6. Speed responses without initial overshoot (no-load)

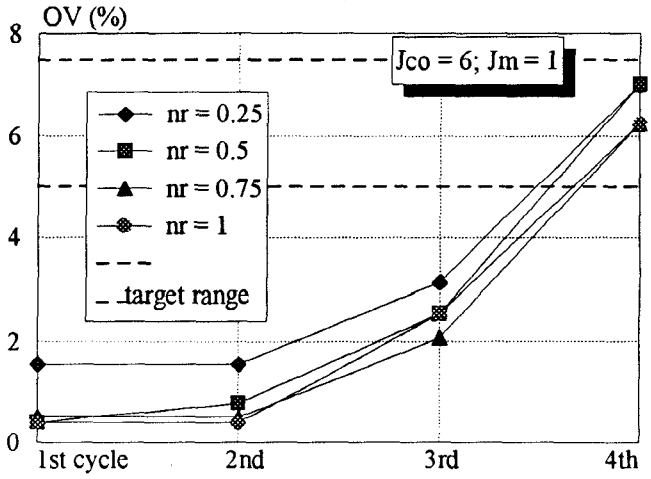

Fig. 7. Overshoot variations corresponding to Fig. 6

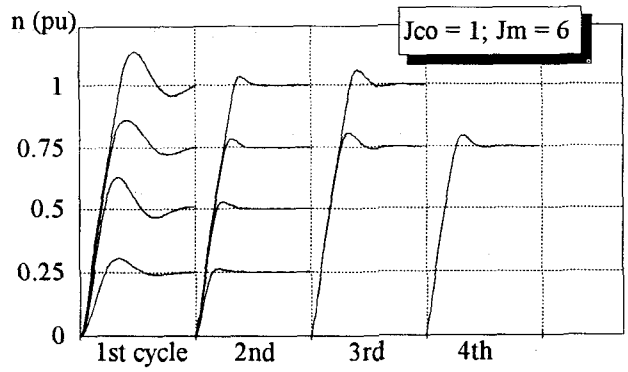

Fig. 8. Speed responses with initial overshoot (m_l = 0.4 pu)

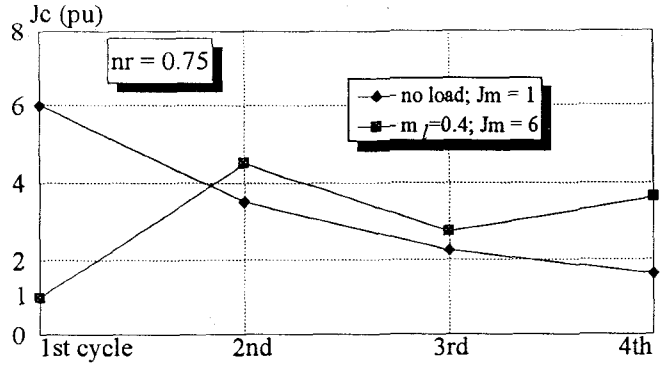

Fig. 9. Estimated controller inertia Jc over the tunig cycles

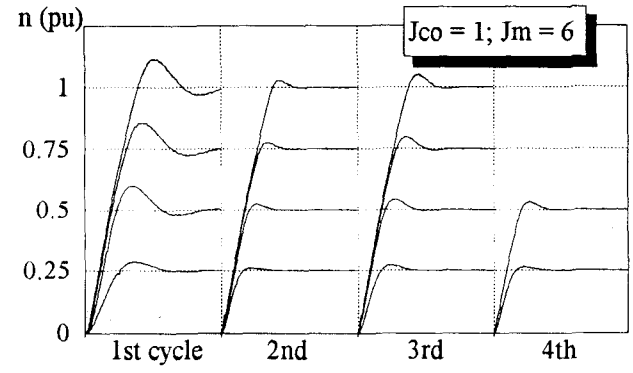

Fig. 10. Speed responses with initial overshoot (m_l = 0.6 pu)

In order to test the robustness of the method to system non-linearity, same experiments with different load torque have been carried out. While in Fig. 8 all the tests are performed with the load torque m_l equal to 0.4 pu, Fig. 10 demonstrates the system behaviours with the load torque equal to 0.6 pu. Apparently, the binary search gain tuning method yields again ideal performance.

The self-tuning ability of the controller to a sudden load variation is demonstrated in Fig. 11. This test is intended to simulate an extreme case when the PI gain tuning is being carried out within a certain searching range while a sudden load change occurs. Assume the load change occurs at the third cycle of Fig. 11, where the setting point is 0.5 pu and the load torque varies from 0.6 pu to 0.4 pu. As from Fig. 10 it is known that the system response should converge in four cycles in the case of a constant load m_l = 0.6 pu. But due to the difference of the dynamic characteristics between the case m_l = 0.4 pu and m_l = 0.6 pu, the system converges after being tuned five times instead of four times in the constant case. The corresponding overshoot variation is reflected in Fig. 12.

It should be noted that the worst case happens when the load change cannot make the system converge in seven times which corresponds to the pre-defined limit number, in this case the auto-tuner will automatically reset the searching range to the full one to restart the searching procedure. So the total convergence cycle number will accumulate to more than 7, but surely smaller than 14 when there is no more parameter variation in the successive cycle [12].

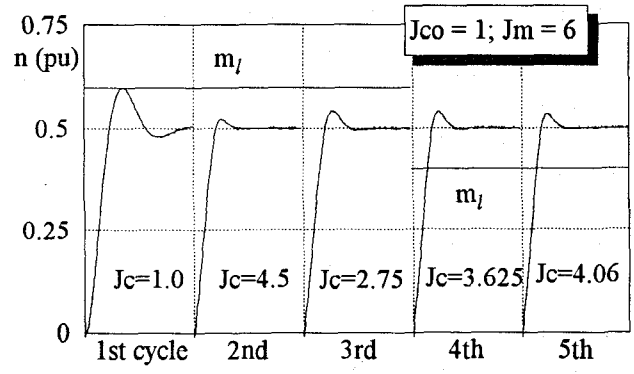

Fig. 11. Speed responses with sudden load change in tuning process

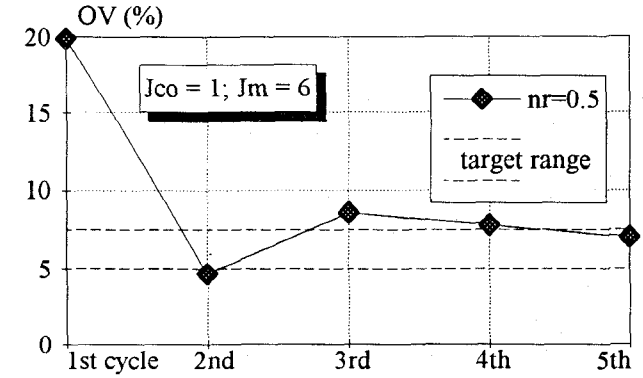

Fig. 12. Overshoot variations with sudden load change in tuning process

V. Conclusions

In this paper, a new and practical model-free gain self-tuning method for the PI speed controller of a PMSM drive

system has been proposed and implemented, its base principle, tuning algorithm and experimental results have been described. The novel method is derived from the analysis of the speed step response of a PMSM drive system and transplants the binary search algorithm to the tuning of PI gain parameters. The proposed method can assure the convergence of the gain-tuning algorithm in all cases, it is also independent of the variations in system parameters such as load torque, speed reference and sampling time. The method has been fully tested with the PMSM drive system, the experimental results verify the simplicity, versatility, stability and effectiveness of the proposed method.

REFERENCES

[1] E. Chiricozzi, F. Parasiliti, M. Tursini, and D.Q. Zhang, "Fuzzy self-tuning PI control of PM synchronous motor drives," *Int. J. Electronics*, vol. 80, n. 2, 1996, pp. 211-221.

[2] K.J. Aström, and Hagglund, "Automatic tuning of simple regulators with specifications on phase and amplitude margins," *Automatica*, vol. 20, 1984, pp. 645-651.

[3] C.C. Hang, K.J. Aström, and W.K. Ho, "Refinements of the Ziegler-Nichols tuning formulas," *Proc. of IEE*, Part. D, vol. 138, pp. 111-118, March 1991.

[4] J. Litt, "An expert system to perform on-line controller tuning," *IEEE Control Systems*, pp. 18-23, April 1991.

[5] B. Porter, A. H. Jones, and C. B. McKeown, "Real-time expert tuners for PI controllers," *Proc. IEE*, Part D. vol. 134, pp. 260-263, July 1987.

[6] T. W. Kraus and T. J. Myron, "Self-tuning PID controllers based on a pattern recognition approach," *Control Engineering*, 1984, pp. 106-111.

[7] S. Z. He, S. Tan, F. L. Xu, and P. Z. Wang, "Fuzzy self-tuning of PID controllers," *Fuzzy Sets and Systems*, 56, 1993, pp. 37-46.

[8] Z.Y. Zhao, M. Tomizuka, and S. Isaka, "Fuzzy gain scheduling of PID controllers," *IEEE Trans. on SMC*, vol. 23, no. 5, 1993, pp. 1392-1398.

[9] K. Izumi, M. Tsuji, J. Oyama, and E. Yamada, "Improvement of fuzzy auto-tuning method of DC chopper system using manipulated values", *Proceedings of the 19th Int. Conf. on Industrial Electronics Control and Instrumentation*, Hawaii, USA, 1993, pp. 224-229.

[10] E. Chiricozzi, F. Parasiliti, M. Tursini, and D. Q. Zhang, "Implementation of a fuzzy self-tuning controller for electrical drives," *Proceedings of the 6th European Conf. on Power Electronics and Applications*, Sevilla, Spain, vol. 3, pp. 440-445, 1995.

[11] S. Tzafestas and N. P. Papanikolopoulos, "Incremental fuzzy expert PID control," *IEEE Trans. Indus. Electron.*, vol. 37, pp. 365-371, Oct. 1990.

[12] D.Q. Zhang, "Fuzzy logic control of electric drives," *Ph.D. thesis*, University of L'Aquila, Italy, 1995.