Real-Time Gain Tuning of PI Controllers for High-Performance PMSM Drives

Marco Tursini, *Member, IEEE*, Francesco Parasiliti, and Daqing Zhang

***Abstract*—In this paper, a new and practical real-time gain-tuning method for proportional plus integral (PI) controllers has been formulated and implemented, using the speed control of a permanent-magnet synchronous motor drive system as a 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 as follows: 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, this latter parameter is estimated with a modified binary search algorithm in order to improve the performance index; and 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 features such as versatility, stability, and effectiveness.**

***Index Terms*—Gain tuning, permanent-magnet synchronous motors, proportional plus integral controllers.**

NOMENCLATURE

J_c	Gain-tuning parameter.
J_m	Inertia of the motor-load system.
m_e, m_l	Electromagnetic/load (disturbance) torque.
IE	Performance index.
OV	Overshoot of the speed step response.
abc	Motor natural three-phase system.
dq	Synchronously rotating reference frame.
i_{abc}, i_{dq}	Motor abc/dq currents.
n, θ	Motor (rotor) mechanical speed/position.
T_p	Equivalent time constant.
r	Subscript for reference (set point) values.

Paper MSDAD-A 02–07, presented at the 1996 Industry Applications Society Annual Meeting, San Diego, CA, October 6–10, and approved for publication in the IEEE TRANSACTIONS ON INDUSTRY APPLICATIONS by the Industrial Automation and Control Committee of the IEEE Industry Applications Society. Manuscript submitted for review October 15, 1996 and released for publication March 22, 2002.

M. Tursini and F. Parasiliti are with the Department of Electrical Engineering, University of L'Aquila, I-67040 L'Aquila, Italy (e-mail: tursini@ing.univaq.it; rock@ing.univaq.it).

D. Zhang is with Kent Ridge Digital Labs, Singapore 119613 (e-mail: daqing@lit.org.sg).

Publisher Item Identifier 10.1109/TIA.2002.800564.

I. INTRODUCTION

NOWADAYS, the vector control technique has made it possible to apply the permanent-magnet synchronous motors (PMSMs) 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 proportional plus integral (PI) and proportional–integral–derivative (PID) controllers, so there exists a growing demand to optimally set the PI and PID parameters in real time according to the varying operating conditions so that even better system performance 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 autotune 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-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 nonlinearity, disturbance, noises, etc., very often, many 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 real-time controller tuning [4]–[6], then fuzzy gain-tuning methods were proposed based on the related experts' gain-setting experiences [7]–[11]; the underlying idea is just to emulate operators or experts' actions during the gain-tuning procedure.

Real-time 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 the following two categories:

- 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.

Until now, only simulation results have been reported for step-based gain-tuning controllers, while the existing commercial systems are almost based on cycle information. Among them, the Foxboro EXACT and the SattControl gain-tuning controllers have been widely accepted in practical applications due to their simplicity and the good performance; they have often been employed by the researchers as benchmark systems to compare the performance of newly developed tuning strategies, [12], [13].

The Foxboro EXACT self-tuning controller employs a pattern recognition technique developed by Bristol [14], also adopted by Yokogawa. Its operation is based on the continuous monitoring of the error signal and the consequent adjustment of the controller setting in order to achieve the user-specified damping and overshoot constraints for the response of the controlled system, even in the presence of load disturbance or set-point changes. The presence or absence of relevant variations (peaks and/or troughs) on the error signal, the time between them, and the steady-state error are assumed as tuning inputs which are related to the overshot and the damping by a proper set of tuning rules. A pretune mode is also available, during which the controller is initialized by the evaluation of the open-loop response of the process. The main disadvantages of the Foxboro EXACT self-tuning controller are the need of a large set-point change to trigger the tuning process, the use of a large number of tuning rules, and the poor performance in the case of sinusoidal disturbance [15].

The SattControl is based on a relay-tuning technique first introduced by Astrom [16] and also employed by Fisher Control. The technique requires putting the system (process and controller) into controlled self-oscillations and to measure the amplitude and period of them. These quantities are related to the gain crossover position on the Nyquist curve of the process, and then they can be used to tune the PID controller. More exactly, due to the use of a relay with hysteresis, a combined gain and phase margin rule is used instead of the classical Ziegler–Nichols tuning formula, [2]. Among the more interesting features, the SattControl autotuner is able to prevent overshoots in the controlled value when a large set-point change is applied. As a drawback, one mentions the loss of the integral control during tuning, which requires a compensation by a self-biasing feature [17].

As for the mentioned systems and the other known self-tuning controllers, most of them are verified in slow process control (such as chemical processes); 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 as follows.

- 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 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.

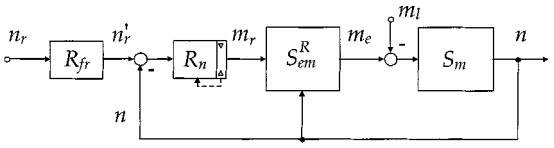

Fig. 1. Speed control scheme of high-performance PMSM drive.

With the above idea, a novel and practical gain self-tuning method for PI controllers has been implemented, and representative experiments have been carried out to verify and evaluate the proposed method. The remarkable features of this self-tuning controller include the following: 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; and third, the proposed method does not depend on a precisely defined mathematical model. While it is simple and easy to implement for real-time applications, it also possesses 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 a high-performance PMSM drive using a 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 ($S_{em}^R + S_m$) represents the controlled system, where S_{em}^R is the internal torque control loop (electromagnetic controlled system), and S_m the mechanical system.

Assuming that the whole inertia of the rotating masses of the motor-load system is constant with respect to the time, the mechanical system can be represented by a first-order differential equation as follows:

$$J_m \frac{dn}{dt} = m_e - m_l \tag{1}$$

where n is the rotor speed, m_e is the electromagnetic torque, m_l is the load torque (load effects are collected in a single term), and J_m the moment of inertia of the motor-load system.

Standard design of the PI speed controller requires the knowledge of the characteristics of the mechanical system, i.e., the motor-load inertia. Supposing that 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 (2) according to [1]

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

where n and n_r are the speed and the speed reference, T_{pe} is the equivalent time constant of the electromagnetic controlled system, and J_c is the moment of inertia assumed in the controller.

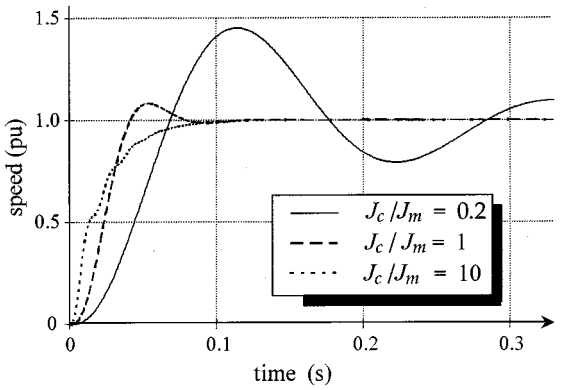

Fig. 2. Speed responses with different system and controller inertia.

If J_c is assumed to be equal to J_m, the optimal response can be obtained with a rising time of about $7.5T_{pe}$ and an overshoot of about 7.5%. Differently, if the system inertia is different from the design value (which happens if the load changes) the optimal response does not hold any more. Fig. 2 demonstrates three typical system responses obtained from (2) in the cases $J_c/J_m = 1$, $J_c > J_m(J_c/J_m = 10)$, and $J_c < J_m(J_c/J_m = 0.2)$, where the system is assumed to have $T_{pe} = 5.5$ ms. Obviously, the responses can be grouped into the following three categories.

- 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 windup, 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 observations, 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 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 is 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 need not to be defined in a very precise way. Actually, if the resulting response falls into a desired range, we can say that the controller works well. Therefore, in real implementations, a target overshoot range can be adopted instead of the single target overshoot value. The performance index IE can thus be defined as

$$IE = OV_a - OV_t \tag{3}$$

where OV_a and OV_t represent the actual and target overshoot, respectively. Since the target overshoot corresponds to a range $[OV_{\min}, OV_{\max}]$, therefore, 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 whether 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 pJ_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 drive system, the moment of inertia J_m should have a deterministic variation range $[J_{m-\min}, J_{m-\max}]$, with the boundary values corresponding to the no load and with the largest inertia conditions, respectively. Therefore, if J_c can be tuned properly in this range, the optimal speed response is expected to be obtained. As J_c has a monotonous relationship with defined IE when J_m and the other parameters are maintained constant, therefore, 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 data in an ordered sequence. Its rough idea is to 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 are found and the searching procedure ends successfully; if the searching data are bigger than the middle one, then continue the above procedure in the upper half of the sequence; if the searching data are smaller than the middle

```
Match Jm in the ordered sequence Jc = (Jc-min,..., Jc-max)
Initialisation:   Jc := (Jc-min+Jc-max)/2;
Reset:            Jc1 := Jc-min;  Jc2 := Jc-max;
Execution loop:
BEGIN
 IF Jc makes IE in target range THEN Jc is desirable,
    stop tuning and reset
 ELSE IF Jc makes IE bigger than target range THEN
                BEGIN
                  Jc1 := Jc;
                  Jc := (Jc1+Jc2)/2;
                END
 ELSE IF Jc makes IE smaller than target range THEN
                BEGIN
                  Jc2 := Jc;
                  Jc := (Jc1+Jc2)/2;
                END
END
```

Fig. 3. Binary search algorithm.

one, then continue the above procedure in the lower 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. Assuming that J_{c-max} and J_{c-min} are the possible maximum and minimum values of J_c, respectively, the applied binary search algorithm is: first, take the middle value J_{c-ave} in range $[J_{c-min}, J_{c-max}]$, in the coming cycle if $IE = 0$, then J_{c-ave} 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 $[J_{c-min}, J_{c-ave}]$; in the opposite, try the middle value in the upper half of J_c as indicated by the range $[J_{c-ave}, J_{c-max}]$. 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 concerns the system noise. Since the calculation of the performance index IE is based on the right measurement of the system speed, the noise in the real system may influence the acquisition of the correct IE information and, thus, result in improper tuning of the PI gains.

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

1) Use filters to smooth the sampling values so as to tackle the system noise.
2) 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 does not converge in a certain number of cycles, all the parameters are reset to initial values so that the gain parameter can be tuned from the beginning in the full range. In this way, the algorithm is assured to converge all the time.

IV. Real-Time Gain-Tuning Implementation: Experimental Setup and Results

The block diagram and the experimental setup of the PMSM drive is shown in Fig. 4. The whole system consists of a host PC, a SIEMENS SAB80C186 microcontroller, a hysteresis analog current controller, a voltage-source inverter, and a three-phase PMSM equipped with an incremental position encoder (motor rated values are listed in Appendix B).

As for the motor loading, a commercial hysteresis brake has been used, which allows us to impose a constant load torque over the whole speed range including standstill.

The motor is controlled according to the principle of the orientation on the rotor flux (control in the rotor d–q reference frame, see Appendix A). As is known, in PMSMs, a linear relationship between the current amplitude and torque is guaranteed by setting at zero the reference of the direct axis current (i_{dr}), which assures also the maximization of the torque-to-current ratio. With such an assumption, the reference of the quadrature (torque) current (i_{qr}) can be obtained by the speed controller. A $dq \Rightarrow abc$ transformation gives the current references in the natural three-phase reference system, needed for the hysteresis regulators. 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, cyclic (step) 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 latest 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 are different. This sort of implementation can assure the required stability of the self-tuning process.

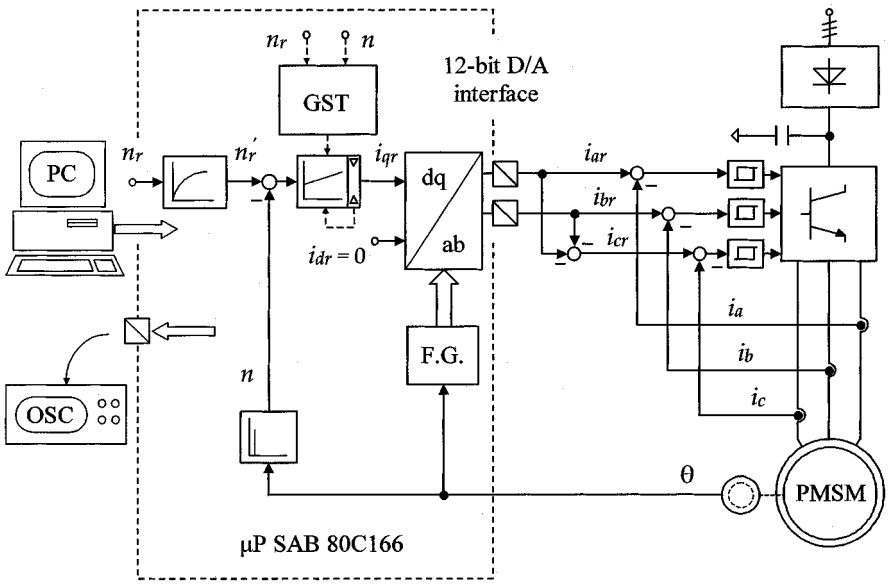

Fig. 4. Block diagram and experimental setup of the PMSM drive.

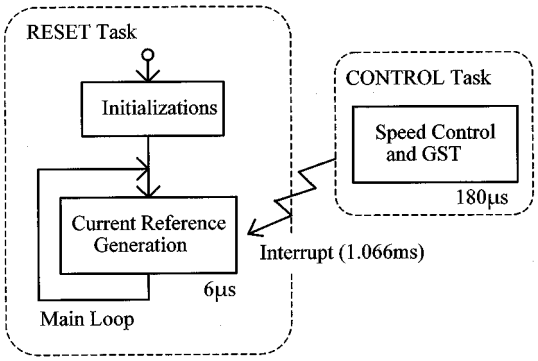

Fig. 5. Control algorithm architecture.

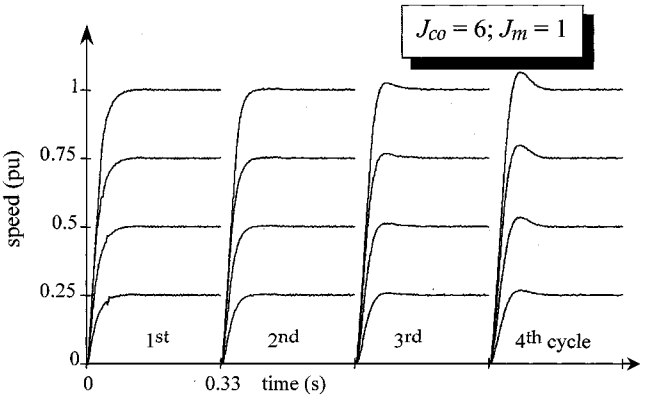

Fig. 6. Speed responses without initial overshoot (no load).

Details on the implementation of the control algorithm 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 inside a never-ending loop (main loop) which implements the generation of the (two) phase current references and their outputs through the D/A converters as follows. At first, the actual rotor position is calculated from the encoder signals read by the microcontroller's pulse-counting unit. Then, the current references are calculated, according to the instantaneous torque (q) current reference, the zero direct (d) current reference, and the axis transformation $dq \Rightarrow abc$ [see (4) in Appendix A]. The sinusoidal functions needed for such transformation are obtained by a lookup 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 of the control program is written in Assembler, using single-precision 16–bit arithmetic. The calculation time of the speed control and GST procedure is about 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. Results are discussed using per-unit (pu) values for the variables; the assumed base values are listed in Appendix C. In all the experiments reported below, the current limit is ± 1.28 pu, while the target overshoot range is chosen as [5.0%, 7.5%] and the J_c searching 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, and 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.

Fig. 7 shows the torque current reference (i_{qr}) and the speed (n) behavior in the case of the step response up to 0.75 pu.

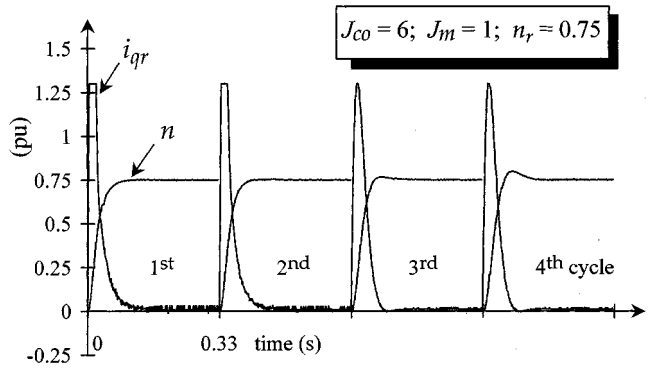

Fig. 7. Torque current reference (i_{qr}) and speed (n) responses without initial overshoot (no load).

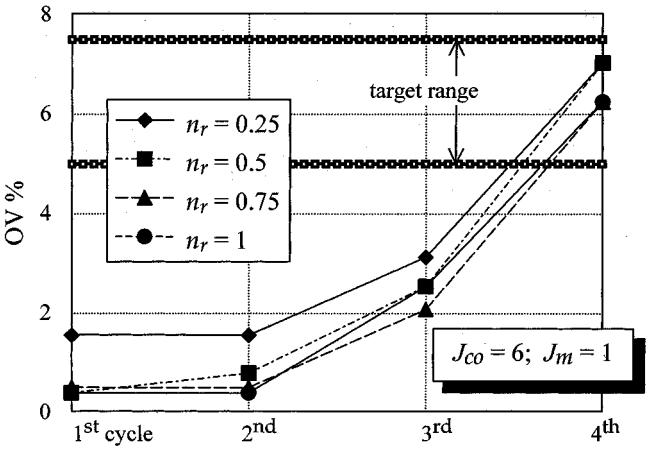

Fig. 8. Overshoot variations corresponding to Fig. 6.

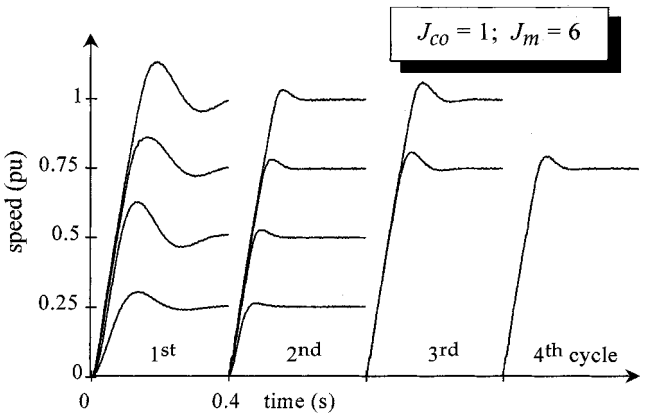

Fig. 9. Speed responses with initial overshoot ($m_l = 0.4$ pu).

One notices that the torque current is in limitation during the first two cycles, when the speed regulator operates in saturation, whereas it leaves such an operation progressively when the tuning process goes on.

Putting the corresponding overshoots of speed responses of Fig. 6 in Fig. 8, it can be shown that despite the fact that different overshoots are related to different working points initially, all the system responses converge quickly to the target range.

Fig. 9 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. 9 that, although the convergence rate varies from case to case due to the nonlinearity of the drive system, in all the cases the speed responses converge quickly to the target overshoot.

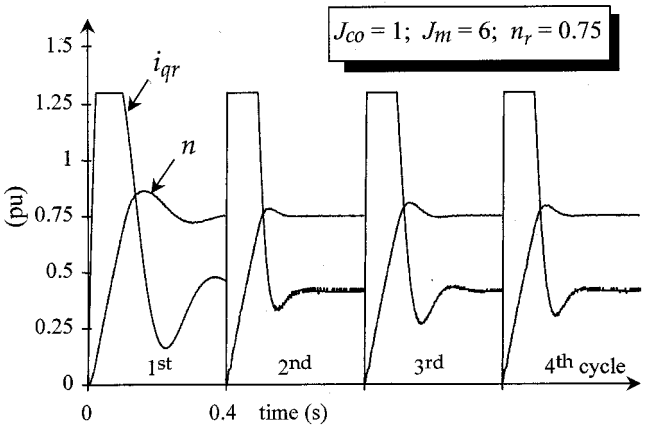

Fig. 10. Torque current reference (i_{qr}) and speed (n) responses with initial overshoot ($m_l = 0.4$ pu).

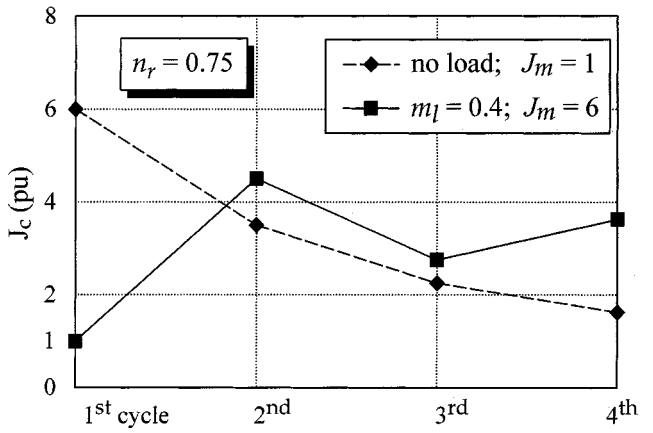

Fig. 11. Estimated controller inertia J_c over the tuning cycles.

Fig. 10 shows the torque current reference versus the speed behavior for the step response up to 0.75 pu presented in Fig. 9. Due to the (relatively) high values of the load torque and inertia, one can see that the torque current attains its limitation during the whole tuning process. Fig. 11 resumes the variations of the gain-tuning parameter J_c over the tuning cycles for the cases reported in Figs. 6 and 9, 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, although the tendency is to move toward the right value.

In order to test the robustness of the method to system nonlinearity, some experiments with different load torque have been carried out. While in Fig. 9 all the tests are performed with the load torque m_l equal to 0.4 pu, Fig. 12 demonstrates the system behaviors with the load torque equal to 0.6 pu. Apparently, the binary search gain-tuning method again yields ideal performance.

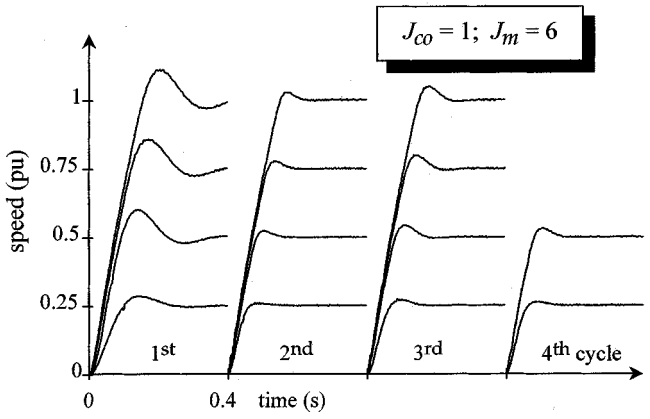

Fig. 12. Speed responses with initial overshoot ($m_l = 0.6$ pu).

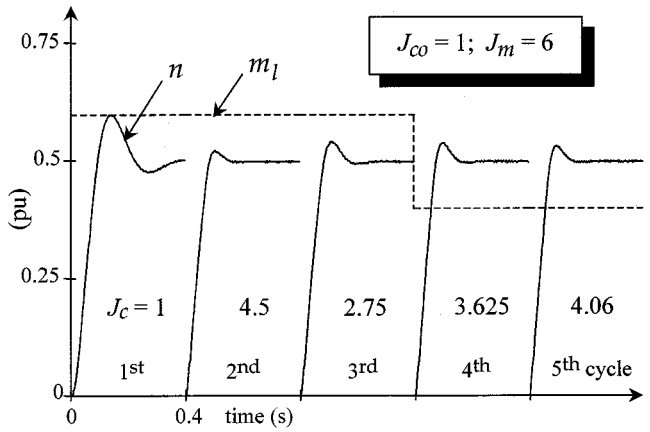

Fig. 13. Speed responses with sudden load change in tuning process.

The self-tuning ability of the controller to a sudden load variation is demonstrated in Fig. 13. 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. 13, where the setting point is 0.5 pu and the load torque varies from 0.6 to 0.4 pu. From Fig. 12, it is known that the system response should converge in four cycles in the case of a constant load $m_l = 0.6$ pu. However, due to the difference of the dynamic characteristics between the cases $m_l = 0.4$ 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. 14.

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 predefined limit number; in this case, the autotuner will automatically reset the searching range to the full one to restart the searching procedure. Therefore, the total convergence cycle number will accumulate to more than seven, but surely smaller than 14, when there is no more parameter variation in the successive cycle [18].

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, and 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, and the experimental results verify the simplicity, versatility, stability, and effectiveness of the proposed method.

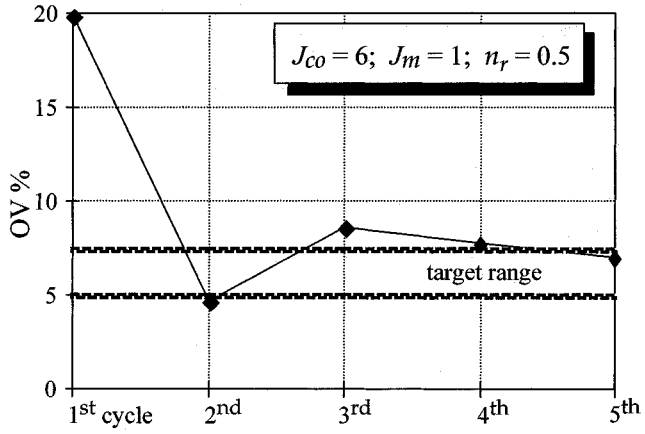

Fig. 14. Overshoot variations with sudden load change in tuning process.

With respect to the known commercial self-tuning controllers, most of which are verified in slow process control (such as chemical processes), the proposed method is suitable for application with fast process control such as electrical servo-drive applications, where the step response is in the order of a few tens of microseconds. The tuning algorithm requires no *a priori* knowledge about the process control. It is basically a binary search engine which involve few and simple calculations (comparisons and assignments), and it can be easily included as a procedure in the real-time control task without affecting the system performance. Moreover, the method is able to operate in the presence of the nonlinearity introduced by the saturation of the controllers, an operating condition often attained by high-performance servo systems.

Appendix A
PM Motor Model

According to the Park theory, a PMSM can be described in terms of rotor fixed axis components dq, where the (direct) d-axis components refer to an equivalent circuit whose magnetic axis is (usually) aligned with the rotor magnetization axis, and the (quadrature) q-axis components refer to an equivalent circuit whose magnetic axis is orthogonal and advances the direct axis in the sense of rotation (Fig. 15). The dq variables and the actual abc ones in the natural three-phase system are related by the axis transformation

$$x_{abc} = \begin{bmatrix} \cos p\theta & -\sin p\theta & 1 \\ \cos(p\theta - 2\pi/3) & -\sin(p\theta - 2\pi/3) & 1 \\ \cos(p\theta + 2\pi/3) & -\sin(p\theta + 2\pi/3) & 1 \end{bmatrix} \cdot x_{dq} \quad (4)$$

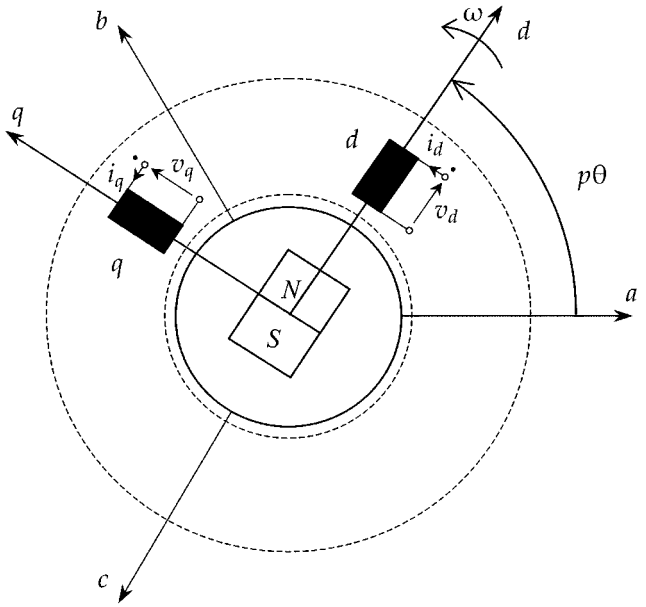

Fig. 15. Schematic d–q modeling of PMSM.

where $p\theta$ represents the rotor axis (electrical) position, and p means the pole pairs number.

The electrical equations of voltages and flux linkages in terms of equivalent d–q circuits are

$$\begin{aligned} v_d &= R_s i_d + \frac{d\psi_d}{dt} - \omega\psi_q \\ v_q &= R_s i_q + \frac{d\psi_q}{dt} + \omega\psi_d \\ \psi_d &= L_d i_d + \hat{\psi}_M \\ \psi_q &= L_q i_q \end{aligned} \tag{5}$$

with R_s being the resistance, L_d and L_q the direct and quadrature synchronous inductances respectively, $\hat{\psi}_M$ the amplitude of the flux linkage by the stator windings due to the presence of the rotor magnet, and ω the (electrical) rotor speed.

The expression of the electromagnetic torque is obtainable by the power balance associated to (5). It gives

$$m_e = p(\psi_d i_q - \psi_q i_d). \tag{6}$$

In the case of a magnetically isotropous rotor structure, which is the usual case in machine tool applications, one has $L_d = L_q = L_s$, *synchronous inductance*, from which the simple relation is obtained

$$m_e = p\hat{\psi}_M i_q. \tag{7}$$

APPENDIX B
TEST MACHINE

The experiments described in this paper were carried out using the following PMSM machine:

pole pairs	4;
rated power	630 W (at 200 Hz);
rated voltage	180 Vrms (line to line);
rated (base) current	2.5 Arms;
rated (base) speed	3000 r/min;
rated torque	2 N·m;
rotor (base) inertia	$22{\cdot}10^{-5}$ kg·m^2.

APPENDIX C
SCALING VALUES

The following base values have been considered for displaying the experimental results:

speed	1 pu $\rightarrow$ 3000 r/min;
current	1 pu $\rightarrow$ 2.5 Arms;
inertia	1pu $\rightarrow$ $220{\cdot}10^{-6}$ kg·m^2.

REFERENCES

[1] E. Chiricozzi, F. Parasiliti, M. Tursini, and D. Q. Zhang, "Fuzzy self-tuning PI control of PM synchronous motor drives," *Int. J. Electron.*, vol. 80, no. 2, pp. 211–221, 1996.

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

[3] C. C. Hang, K. J. Aström, and W. K. Ho, "Refinements of the Ziegler–Nichols tuning formulas," *Proc. Inst. Elect. Eng.*, pt. D, vol. 138, pp. 111–118, Mar. 1991.

[4] J. Litt, "An expert system to perform on-line controller tuning," *IEEE Contr. Syst. Mag.*, vol. 11, pp. 18–23, Apr. 1991.

[5] B. Porter, A. H. Jones, and C. B. McKeown, "Real-time expert tuners for PI controllers," *Proc. Inst. Elect. Eng.*, pt. 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 Eng.*, vol. 31, pp. 106–111, 1984.

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

[8] Z. Y. Zhao, M. Tomizuka, and S. Isaka, "Fuzzy gain scheduling of PID controllers," *IEEE Trans. Syst., Man, Cybern.*, vol. 23, pp. 1392–1398, Sept./Oct. 1993.

[9] K. Izumi, M. Tsuji, J. Oyama, and E. Yamada, "Improvement of fuzzy auto-tuning method of DC chopper system using manipulated values," in *Proc. 19th Int. Conf. Industrial Electronics Control and Instrumentation*, 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," in *Proc. 6th European Conf. Power Electronics and Applications*, vol. 3, Seville, Spain, 1995, pp. 440–445.

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

[12] C. C. Hang and K. K. Sin, "On-line auto tuning of PID controllers based on the cross-correlation technique," *IEEE Trans. Ind. Electron.*, vol. 38, pp. 428–437, Dec. 1991.

[13] A. B. Rad, W. L. Lo, and K. M. Tsang, "Self-tuning PID controller using Newton–Raphson search method," *IEEE Trans. Ind. Electron.*, vol. 44, pp. 717–725, Oct. 1997.

[14] E. H. Bristol, G. R. Inaloglu, and J. F. Steadman, "Pattern recognition: An alternative to parameter identification in adaptive control," *Automatica*, vol. 13, pp. 197–202, mar. 1977.

[15] K. W. Lim and C. C. Hang, "Performance study of an adaptive PID controller," in *Proc. Int. Workshop Applications of Adaptive Systems Theory* Yale Univ., New Haven, CT, May 1985, pp. 30–35.

[16] K. J. Aström, "Ziegler–Nichols auto tuners," Dept. Automat. Control, Lund Inst. Technol., Lund, Sweden, Rep. TFRT-3167, 1982.

[17] C. C. Hang and K. J. Aström, "Practical aspects of PID autotuners based on relay feedback," in *Proc. Int. Symp. Adaptive Control of Chemical Processes*, Copenaghen, Denmark, Aug. 1988, pp. 149–154.

[18] D. Q. Zhang, "Fuzzy logic control of electric drives," Ph.D. dissertation, Univ. L'Aquila, L'Aquila, Italy, 1995.

Marco Tursini (M'99) was born in S.Pio delle Camere, Italy, in 1960. He received the M.S. degree in electrical engineering from the University of L'Aquila, L'Aquila, Italy, in 1987.

In 1987, he joined the Department of Electrical Engineering, University of L'Aquila, as an Associate Researcher. He became an Assistant Professor of power converters, electrical machines, and drives in 1991, and an Associate Professor of electrical machines in 2002. In 1990, he was a Research Fellow in the Industrial Electronics Laboratory, Swiss Federal Institute of Technology, Lausanne, Switzerland, where he conducted research on sliding-mode control of permanent-magnet synchronous motor drives. His research interests are focused on advanced control of ac drives, including vector, sensorless, and fuzzy logic control, digital motion control, DSP-based systems for real-time implementation, and modeling and simulation of electrical drives. He has authored more than 60 technical papers on these subjects.

Francesco Parasiliti was born in Tortorici, Italy, in 1956. He received the M.S. degree in electrical engineering from the University of Rome, Rome, Italy, in 1981.

In 1983, he joined the Department of Electrical Engineering, University of L'Aquila, L'Aquila, Italy, as an Assistant Professor. He became an Associate Professor of electrical drives in 1992, and a Full Professor in 2000. From 1987 to 1988, he was a Research Fellow at the Swiss Federal Institute of Technology, Lausanne, Switzerland. His studies deal with design optimization of induction, permanent-magnet synchronous, and reluctance motors, modeling and parameter observation of induction and synchronous machines, and digital control of electrical drives including vector, sensorless, and fuzzy logic control.

Daqing Zhang received the B.E. degree in computer engineering from China Mining University, Xian, China, in 1984, the M.E. degree in computer engineering from China Aeronautical Computing Technique Institute, Xian, China, in 1987, and the Ph.D. degree in electrical engineering from the University of Rome "La Sapienza," Rome, Italy, and the University of L'Aquila, L'Aquila, Italy, in 1996.

From 1987 to 1990, he worked on parallel processing and distributed processing at the China Aeronautical Computing Technique Institute. From 1990 to 1992, he was a Visiting Researcher at the University of L'Aquila, working on a transputer-based real-time control system. Since 1996, he has worked at the National University of Singapore, Data Storage Institute, and Kent Ridge Digital Labs (KRDL), Singapore. He is currently a member of the research staff at KRDL, where he is involved in research on pervasive computing, home networking, mobile networking, and service gateway.