Adaptive Maximum Torque per Ampere Control of Synchronous Reluctance Motors by Radial Basis Function Networks
Adaptive Maximum Torque per Ampere Control of Synchronous Reluctance Motors by Radial Basis Function Networks

Ludovico Ortombina, Fabio Tinazzi, Mauro Zigliotto

Abstract—As neodymium and other rare earth materials become a critical commodity, the exploitation of reluctance torque in synchronous motors is certainly an interesting option. Alternative motor topologies span from internal permanent magnet motors to pure reluctance machines. Anyway, any anisotropic structure suffers of some magnetic nonlinearity that call for more sophisticated models to get an efficient torque control. Neural network based algorithms are good candidates for modelling the current-to-flux linkages curves of synchronous reluctance motors, but so far their use was limited by the inherent complexity and the computational burden. This paper proposes the use of a special kind of neural networks, namely, the radial basis function networks, to get the magnetic model of any synchronous motor, including saturation and cross-coupling effects. Through experimental evidence, it will be shown that the structure is light enough to be implemented, trained and self-updated online on standard high-end ac drives. The model is used to track online the maximum torque-per-ampere working point of a synchronous reluctance motor drive.

Index Terms—Synchronous reluctance motors, Neural network, Maximum torque per Ampere, Online identification, Motor drives.

I. INTRODUCTION

As every scientific stream, the control of electrical drives is a system in a dynamic equilibrium among innovative push and technological constraints, with influencing parameters as cost and market demand. Permanent magnet (PM) synchronous motors drives play a key-role in a productive world that requires energy-awareness, reliability, high-performance and cheapness all together. Nevertheless, as far as rare-earth materials become a cost-critical commodity, new constructive topologies appears. Essentially, several motor manufacturers propose the gradual migration from isotropic PM motors to pure synchronous reluctance (SynR) motors. The lack of surface-mounted magnets and of the related equivalent airgap makes the new topologies more prone to iron magnetic saturation and cross-coupling.

Elaborated control techniques aim at exploiting the double torque generation mechanism (PM-based and reluctance) in the most efficient way ever, but they need reliable motor models - especially in case of fractional-slot motors [1]. On the other hand, such models are highly appreciated to give the drive additional self-commissioning and diagnostic features.

The choice of the model is driven by the required degree of completeness. The model can be either simplified by using approximation functions [2] (somewhat loosing accuracy) or tabulate [3] (somewhat loosing resolution). A comprehensive discussion about the existing conventional techniques can be found in [4], [5]. At the end, one may come to the conclusion that the complexity of the interaction in and between the magnetic axes is the ideal application for a black-box approach, which has been proven to accomplish the task in a rather smart way [6].

Actually, the use of artificial neural networks (ANN) in electric drives is not new [7]–[9], even as auxiliary mean for tuning other control techniques, as genetic algorithms [10]. So long, ANNs were considered too demanding for the standard hardware of an electric drive, but nowadays the computational power obtained from the combined use of fast floating point processors and FPGA (field programmable gate array) allows to overcome the obstacle.

A good proof of concept is given by [11], in which a two-degree of freedom PID for the speed control of a PM synchronous motor has been emulated by a neural speed controller, trained online on a high-end ac drive. In that case, the implementation on the laboratory prototype was eased by the adoption of a resilient back-propagation training algorithm, well suited for the application to a linear system and anyway modified on purpose to achieve the astaticism of the controller.

With the same focus on feasibility, the present work is based on the use of a particular class of artificial neural networks, namely the radial basis function (RBF) networks [12]. They use Gaussian curves as activating functions instead of conventional sigmoids and have some local properties which greatly enhance the online update capability, as it will be shown later in the paper. A black-box approach as that offered by RBF networks has proven to be effective and accurate even in case of the highly nonlinear and cross-coupled magnetic model of SynR motors, as detailed in [6]. The limit of that implementation was the complex training algorithm, that made it unsuitable for online implementation. Actually, the magnetic model was accurate, but it had to be necessarily computed offline. This paper proposes a different training algorithm, so that it is possible to exploit the steady state working conditions during the normal drive operations to update (online) the magnetic model of any synchronous reluctance motor, to account for the influence of exogenous and endogenous variables as temperature and ageing, respectively. The RBF training takes just a small portion of time in the lifespan of the SynR motor.
It can be carried out during the very first hours of the entire lifetime span. In this scenario, the RBF network training is a rather cheap price to pay in terms of time.

The result is the on-board availability of a comprehensive and continuously updated model of the relations between currents and flux linkages. Own to the explicit continuous mathematical formulation, the model is well suited for the implementation of advanced energy-savings techniques, as maximum torque-per-ampere (MTPA) or model-based predictive control (MPC), without resorting to any disrupting signal injection. As an effective experimental example, the paper shows the application to the online MTPA point tracking, for the sake of an energy efficient torque control of a SynR motor.

There are several MTPA techniques available in literature, but they can be categorised in two classes, namely the online and offline techniques. The former often requires the adoption of disrupting and continuous signal injection [13], [14] or the use of searching algorithms [15]. The latter are based on offline measurements whose results are stored into look-up tables [16], [17] or implemented for MTPA operation with the aid of polynomial functions [18]. The proposed MTPA technique belongs to the online category for what concern the searching of the MTPA point. However, the classical drawback of these techniques, i.e. the chattering due to the signal injection, is tackled with the aid of the RBF network.

The technique proposed in this paper shares the underlying idea of [6] about the adoption of the AC drive-oriented RBF network for estimating the magnetic flux linkages of a synchronous motor. However, two important innovations are the motivation of this paper. The first one is the study and implementation of a new and lean learning algorithm that enables the online training of the RBF network without any offline operation. The second innovation is the use of the RBF network-based flux linkages to the achievement of the real MTPA condition at any time. In particular, the online implementation of the RBF network yields a true adaptive MTPA-based current control. In the paper, the theoretical background is presented in Sect. II, with a deepening about the local property is given in Sect. II-B. The design of the adaptive RBF network, with the hints for a practical implementation are found in Sect. III and the related paragraphs. The adaptive MTPA tracking algorithm is described in Sect. IV. The practical implementation and validation are reported in Sect. V, with extensive comments and conclusions.

II. BASICS OF THE RBF-BASED SYN R MOTOR MODEL

Adopting the space vector notation, the torque \( \tau \) and voltage equations for a SynR motor in the \( dq \) reference frame fixed to the rotor are the following:

\[
\begin{align*}
\mathbf{u}_{dq} &= R_s \mathbf{i}_{dq} + \frac{d(\lambda_{dq}(\mathbf{i}_{dq}))}{dt} + j\omega_{me}\lambda_{dq}(\mathbf{i}_{dq}) \\
\tau &= \frac{3}{2} p (\lambda_d(\mathbf{i}_{dq})i_q - \lambda_q(\mathbf{i}_{dq})i_d)
\end{align*}
\]

where \( R_s \) is the stator resistance, \( \omega_{me} \) is the electric rotor angular speed, \( p \) are the pole pairs and \( \mathbf{u}_{dq} = u_d + ju_q, \mathbf{i}_{dq} = i_d + ji_q, \lambda_{dq} = \lambda_d + j\lambda_q \) are the stator voltage, current and flux linkages, respectively. Conventionally, the \( d \)-axis is fixed along the minimum reluctance path of the rotor and, just this once, all the (nonlinear) dependences on the stator currents are made explicit.

The proposed online identification procedure takes advantage of the steady state working periods during the normal operations of the drive. Therefore, for the scope of the present discussion, the (steady state) voltage estimate is simply

\[
\hat{\mathbf{u}}_{dq} = R_s \mathbf{i}_{dq} + j\omega_{me}\hat{\lambda}_{dq}
\]

without the derivative terms, but with the cross-coupling terms still present. In the following, the main concepts about RBF networks will be briefly recalled, while an in-depth analysis can be found in [6] and [12]. As shown in Fig.1, the flux linkages are estimated by a particular ANN whose weights can be expressed in compact form as a vector of complex numbers:

\[
\mathbf{w}^{dq} = \begin{bmatrix} w_{d1} + jw_{q1}, w_{d2} + jw_{q2}, \ldots, w_{Kd} + jw_{Kq} \end{bmatrix}^T.
\]

The weights are updated according to a training algorithm that aims at minimising the error \( \epsilon_{dq} \) between the measured and the estimated motor voltage vector:

\[
\epsilon_{dq} = \epsilon_d + j\epsilon_q = \mathbf{u}_{dq} - \hat{\mathbf{u}}_{dq} = \mathbf{u}_{dq} - R_s\mathbf{i}_{dq} - j\omega_{me}\hat{\lambda}_{dq}
\]

The algorithm requires the a-priori knowledge of a precise and updated value of the stator resistance \( R_s \), which can be obtained by modern algorithms as for example those described in [19] and [20]. The training algorithm description is reported in Sect. III.

A. RBF first and second layers

As regards the neural network, a two layer structure, as that depicted in Fig. 1, is suitable for the flux linkages estimation. The first layer was designed to deal with any non-linear input-output behaviour. It is composed of \( K \) Gaussian functions (so called neurons) whose centres were regularly spaced in the \((i_d-i_q)\) plane. For any given input current vector \( \mathbf{i}_{dq} \) the output of the first layer can be expressed in a compact form as a vector of \( K \) real numbers

\[
\mathbf{a} = [a_1, a_2, \ldots, a_K]^T.
\]

In (5) each Gaussian function is activated (i.e. it produces an output) according to the following expression:

\[
a_k = e^{-\|\mathbf{i}_{dq} - \mathbf{a}_k\|^2}(k = 1 \ldots K)
\]
where \(\|i_{dq} - g_k\|\) is the Euclidean distance between the \(k\)-th Gaussian centre \(g_k = g_k^d + jg_k^q\) and the sampled input current vector, and \(b\) is a positive gain whose details are given in Sect. III-B.

The second layer calculates two weighted linear combinations (one for each of the two flux linkage components) of the hidden layer outputs \(a_k\):

\[
\tilde{\lambda}_d = \tilde{\lambda}_q = \sum_{k=1}^{K} a_k (u_k^d + ju_k^q) = a^T w^d_{dq}
\]  

(7)

Substituting (7) in (4) returns the explicit links between the voltage errors and the RBF weights, i.e. the input and the output of the learning algorithm of Fig. 1:

\[
e_{dq} = u_{dq} - R_d i_{dq} - j\omega_{me} a^T w^d_{dq}
\]  

(8)

B. Local property of the RBF network

One of the main reasons for using the RBF networks in online algorithms is the local property, which makes \(a_k\) close to zero whenever the input is far enough from the centre \(g_k\) of the related Gaussian function. As mentioned, this is very useful during the online training, which is performed in a certain steady state condition and not over the entire input range. Thanks to the local property, the update action remains limited to the surroundings of the steady state point, without (badly) influencing the other input regions. It has been found that this simplify the network training, reducing the number of computations. Of course, a complete update needs several local trainings on different steady state points As an example, Fig. 2 reports the \(a_k\) values, i.e. the output of the RBF hidden layer, calculated with the input \((i_d, i_q) = (0, 0)\). It is worth noting that only the Gaussian functions close to the input return a non-zero output.

III. THE RBF TRAINING AND DESIGN HINTS

The two main features of any neural network are the structure (layers and neurons) and the learning algorithm, i.e. the way a neural network is trained to perform its task. Actually, the learning algorithm is the key-factor, especially in case of an application that requires an online adaptation, as in the present work. Bearing in mind the reduced computational resources in industrial drives, the training should be selected as computationally light as possible. Although the Levenberg-Marquardt (LM) is the fastest algorithm in the minimisation of non-linear quadratic problems, and thus can be considered as a standard choice, it brings along heavy computations and it was discarded from the very beginning. Of course, the algorithm heaviness is far less important in case of offline implementations, as it was in [6]. It is important to distinguish between a conventional ANN and the special drive-oriented RBF network proposed in this paper. The former undergoes the conventional training algorithms (for example the back-propagation) that minimise the error iteratively. Conversely, the RBF network allows a direct (non-iterative) computation of the weights update as it will be shown below.

For the design of a computationally efficient learning algorithm it is worthwhile to present a few important considerations:

- As inferable from (6), any current vector measurement \(i_{dq}\) produces a vector \(a\) of real numbers. Once substituted in (8), together with the related voltage and speed measurements, it yields an error function whose components \(e_d\) and \(e_q\) are linear combinations of the RBF weights \(w^d_{dq}\).
- At steady state (i.e., for a constant set of voltage, current and speed vectors), the optimal RBF weights \(\hat{w}^d_{dq}\) are those that bring to zero the error vector (8):

\[
u_{dq} - R_d i_{dq} - j\omega_{me} a^T \hat{w}^d_{dq} = 0
\]  

(9)

The system is clearly underdetermined, since there are \(K\) unknown variables (the weights) and just one equation.

- Given the local property of the proposed network (Sect. II-B), the weight set should be updated aiming at improving the flux linkage estimation in the neighbourhood of the considered input only. In other words, it is meaningless to modify the weight linked to a Gaussian far from the present steady state point, since it is not activated (Fig. 2).

The new learning algorithm proposed in this paper aims at changing the existing weight vector \(w^d_{dq}\) into \(\hat{w}^d_{dq}\), so to satisfy the condition (9). One can write:
Replacing (12) and (11) in (10) finalises the proposed training rule for the RBF network:

$$\tilde{w}^{dq} = w^{dq} + \frac{\epsilon_{dq} A}{jw_{me} \sum_{k=1}^{K} a_k b_k}$$ (13)

Every time the ac drive enters in a steady state condition, the voltage error (4) is averaged on a suitable number of measurements, to reduce the possible errors due to spikes and noise, and then the weights of the second layer are updated according to (13).

A. Approximation of the Gaussian functions

The online implementation of (13) requires the computation of the exponential functions contained in (5) and the task may exceed the computational power of the drive. Therefore, an appropriate numeric approximation was considered in the present work.

First of all, it is worth noting that the exponents of $a_k$ in (6) ideally ranges from 0 (when an input exactly matches a Gaussian centre) to minus infinity, when the actual current measurement is infinitely far from the Gaussian centre. Accordingly, the exponents $a_k$ range from 1 to 0. In practice, it is reasonable to fix a lower limit $\xi$

$$\xi \leq a_k \leq 1 \rightarrow \ln(\xi) \leq - (\|I_{dq} - g_k\| b)^2 \leq 0$$ (14)

so that only the exponentials whose exponent is in the range indicated by (14) will be considered for approximation, while the others will be forced to zero. In this work, it was found that a good value was $\xi = 0.01$. The generic exponential function was approximated by a 5-th order polynomial, whose coefficients where computed offline by imposing the least mean square error in the range specified by (14):

$$e^x = c_5 x^5 + c_4 x^4 + c_3 x^3 + c_2 x^2 + c_1 x + c_0 \quad ( -4.06 \leq x \leq 0)$$ (15)

The values of the coefficients are reported in TABLE I. The choice of the order of the polynomial was a trade-off between the required computational power and the mean square approximation error.

B. Design hints for the RBF hidden layer

The shape of the Gaussian functions that describe each neuron of the hidden layer, and at large the local property of the RBF network, depends on the parameter $b$ in (6). It also influences the number $K$ of Gaussian functions, as described in [6]. For the sake of input space completeness, the tighter the Gaussian functions, the higher the number of hidden neurons. The choice of $b$ can be made so that a Gaussian function gives a negligible contribution ($a_k$ in (6)) for current vectors whose Euclidean distance from the Gaussian centre is greater than a design variable $r$ (Fig.3). In the present work, it was fixed to half of the rated current ($r = I_N/2$). According to (14), this rule of thumb is expressed by

$$b = \sqrt{\frac{-\ln(\xi)}{r}} = 2 \sqrt{\frac{-\ln(\xi)}{I_N}}$$ (16)

In literature, the number $K$ of hidden neurons is linked to $b$ and to the side $l_N$ of the squared training region (Fig.3) by the empirical relation [6], [12]:

$$\sqrt{K} = 4\sqrt{2} l_N b \rightarrow K = -128 \ln(\xi)$$ (17)

where the right-hand equation is obtained using (16).

In general, the value returned by (17) is a real number. It is necessary to approximate it to the nearest perfect square, so that the centre of the Gaussian functions will be disposed evenly and symmetrically over the training region, which is a square with side equal to $2l_N$. In this paper, as $\xi = 0.01$, the number of Gaussian functions was $K = 576$. Once $K$ is set, the coordinates $g_k$ of the Gaussian centres in the $dq$ current plane can be easily derived (Fig. 3).

As a final comment, it is worth noting that all the first layer parameters, i.e. the number of Gaussian functions and their shapes, are chosen during the design phase, while the RBF weights that constitute the heart of the second layer are obtained by online training and adjustment.

IV. THE MTPA TRACKING ALGORITHM

Substituting (6) in (7) makes clear that the flux linkages estimates are continuous and derivable functions of the stator currents. This is peculiar to the proposed algorithm and quite useful in case of control algorithms that make use of flux linkages derivatives, as in the case of MTPA tracking schemes as that presented hereafter.

For a given reference current vector $i_{dq}^* = I^* e^{j\varphi^*}$ of fixed amplitude $I^*$ and variable phase $\varphi^*$, the maximum torque-per-amperes condition is obtained by imposing a null derivative of the torque (1) with respect to $\varphi^*$:

$$\frac{\partial T}{\partial \varphi} = \frac{3}{2} \frac{\partial (\lambda_{d} i_d^* - \lambda_{q} i_q^*)}{\partial \varphi} = 0$$ (18)
Substituting $i_d^* = I^* \cos(\vartheta^*)$ and $i_q^* = I^* \sin(\vartheta^*)$ and by means of the explicit expression of the flux linkages estimation (7), the condition (18) becomes

$$\frac{3}{2} p I^* \left( \lambda_{dq} \sin(\vartheta^*) + \dot{\lambda}_{dq} \sin(\vartheta^*) + \sum_{k=1}^{K} \left( w_{d_k}^2 \sin(\vartheta^*) - w_{q_k}^2 \cos(\vartheta^*) \right) \frac{\partial a_k}{\partial \vartheta^*} \right) = 0 \quad (19)$$

Actually, the $a_k$ terms, defined by (6), depend on the stator current. Using the polar notation, the derivative of $a_k$ respect to the current phase angle is

$$\frac{\partial a_k}{\partial \vartheta^*} = 2 a_k b^2 I^* (g_k^2 \sin(\vartheta^*) - g_k^2 \cos(\vartheta^*)) \quad (20)$$

where $g_k^d + jg_k^q$ is the centre of the $k$-th Gaussian function in the $(i_d, i_q)$ plane, as already mentioned above.

Anyway, the straightforward computation of $\vartheta^*$ might be burdensome. The left-hand term of (19) is greater than zero for current phase angles below the correct one, and vice-versa. Actually, it can be found that the mismatch computed by (19) is proportional to the required current phase angle modification. This is exploited by making the current phase modification proportional to the result of (19). In turn, when the result of (19) is big, the MTPA loci is far. Thus, the angle $\vartheta^*$ can be modified proportionally to the result of (19). It is worth noting that the online MTPA condition is obtained without any disturbing signal injection, as a key-feature of the proposed method.

V. EXPERIMENTAL RESULTS

The experiments were performed on a SynR motor prototype, whose parameters are shown in TABLE II. The motor was fed by a two-level three-phase IGBT voltage inverter, connected to a variable voltage DC bus and controlled by a fast control prototyping system featuring a dSpace DS1104 controller board, programmed in C-language. The motor phase-to-phase voltages (two out of three) were measured with a custom digital measurement system, based on fast PWM signal oversampling, post-processed by a dedicate FPGA chip.

### Table II

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nominal current ($I_N$)</td>
<td>4 A</td>
</tr>
<tr>
<td>Nominal speed ($\omega_{in,N}$)</td>
<td>1500 rpm</td>
</tr>
<tr>
<td>Nominal Torque ($T_N$)</td>
<td>8 Nm</td>
</tr>
<tr>
<td>Pole pairs ($p$)</td>
<td>2</td>
</tr>
<tr>
<td>Stator resistance ($R_s$)</td>
<td>4.76 Ω</td>
</tr>
<tr>
<td>d-axis inductance ($L_d$)</td>
<td>380 mH</td>
</tr>
<tr>
<td>q-axis inductance ($L_q$)</td>
<td>85 mH</td>
</tr>
<tr>
<td>DC link voltage</td>
<td>540 V</td>
</tr>
<tr>
<td>PWM switching frequency</td>
<td>10 kHz</td>
</tr>
</tbody>
</table>

Fig. 4. The SynR motor (left) coupled to a PMSM-based virtual load (right).

A. Validation of the RBF-based flux linkages model

This test evaluates the capability of the proposed RBF network to model the flux linkages of the SynR motor under test. The algorithm requires the measurements of motor voltages and currents, as well as the availability of a precise value of the stator resistance. The experiment is performed offline, that is, with the SynR motor mounted on a laboratory test bench instead of on a real application. It should be understood that this part is only functional to the validation of the RBF network, while it can be omitted in case of direct implementation on an industrial application. The hardware setup is shown in Fig. 4. The SynR motor was current-controlled in the synchronous $d, q$ reference frame, and rigidly coupled to a speed-controlled load motor. A torque transducer provides an accurate measurement of the transmitted torque.

The RBF weights training procedure starts by the acquisition of voltages, currents and speed measurements at each steady state condition, followed by the training algorithm (13). Since the training procedure is carried out during steady state condition, the updating of the RBF weights is suspended during transients. However, the torque and current control use the MTPA curve "as it is", i.e. as trained up to the moment in which the transient starts. The two procedure (MTPA adaptation and torque control) do not interfere each other. The current references to the SynR motor drive were generated in correspondence to the crossings of a gridded region in the $(i_d, i_q)$ plane, as in [6], to cover the whole training region (Fig. 3). The grid resolution was fixed to $I_N/10$. The training was performed by the rule (13), with the simplifying assumptions discussed in Sect. III-A.

The validation of the magnetic model was performed by comparison with the benchmark (offline) method [3]. Let the normalised errors be defined by a compact vector notation as

$$e_{N}^{dq} = \frac{\hat{\lambda}_{dq} - \lambda_{dqR}}{\max(\lambda_{dqR})} \cdot 100 \quad (21)$$

where $\hat{\lambda}_{dq}$ are the RBF network estimates and $\lambda_{dqR}$ are the values obtained by the reference method proposed in [3]. The errors are referred to the maximum value of each flux linkage in the considered working region. The plots of both flux linkages, in case of either null or maximum cross-coupling are reported in Fig. 5, along with the estimation errors, computed according to (21).

The error magnitude remains almost within ±3.5% for both flux linkages, if one excludes the case $\lambda_q(0, i_q)$ in the low-current range, where it rises up to 7% due to a certain weakness in the RBF network around zero currents. It has been found that this issue can be solved by increasing the number of neurons in the region, in trade-off with the computational burden.
It is also worth mentioning that the benchmark method is of discrete type, i.e. it produces a finite set of values, arranged in a couple of matrix look-up tables (LUTs). On the contrary, the proposed method returns the two-variables continuous functions indicated by (7). Therefore, a fair comparison is performed only in correspondence to a finite set of points. Elsewhere, even the benchmark method could introduce errors due to the linear interpolation among points.

It is important to note that the proposed training algorithm is extremely lighter than the one based on the LM algorithm in [6], enabling the implementation on a standard ac drive hardware. It can be found that the time complexity (big O notation) of the LM algorithm is \( O(K^2N) \), whereas the one proposed in this paper is \( O(K) \), as inferable from (13). Of course, it is worth reminding that the proposed algorithm needs more iterations to train the RBF network with respect to the LM based algorithm.

The online training feature will be exploited to obtain an efficient MTPA control strategy starting from a blank RBF network. The results are reported in Sect. V-C. In order to accelerate the construction of the fluxes maps, it could also be possible to take advantage of the symmetries and anti-symmetries proper of those functions. The SynR motors magnetic fluxes are anti-symmetric functions of the currents. Therefore, the third quadrant of the \( dq \) current plane is actually the same of the first quadrant, but with opposite sign. The properties of the fluxes and currents relationships help in speeding-up the RBF network training, but they do not alter the concept of the proposed method.

B. Online RBF training with suboptimal MTPA control

The previous sections have demonstrated the ability of RBF networks to cope with the complexity of the SynR motor nonlinear magnetic model. The training of the network is performed online, exploiting every single steady state condition during the normal drive operations.

To get an accurate flux linkages estimate, the stator resistance needs to be a known parameter. In the present work, an online tracking algorithm similar to that proposed by [20] was implemented. Essentially, it is based on a DC current injection over the \( q \)-axis and allows the tracking of the resistance with high accuracy.

A first experiment was performed still out of the normal operations, in this case to explore the capability of the network to evolve from a completely blank situation. Six successive load torque steps (from 0 to the nominal torque \( T_N \) where imposed to the SynR motor by a virtual active load (Fig. 4), while the SynR motor was speed-controlled at a constant speed of 350 rpm. A simple sub-optimal \((i_d = i_q)\) current control strategy was selected, and the flux linkages estimation was activated at each step. In this way it was possible to investigate the local character of the RBF network. Fig. 6 shows the maps of \( \lambda_{dq}(i_d, i_q) \) after the six-step procedure. The local property of the network implies that all (and only) the weights close to the \( i_{dq} \) points are actually updated. As shown in Fig. 6, the estimated flux linkages are not null closely around the \( i_d = i_q \) line. However, the correctness of the estimation rapidly decrease as one deviates from the \( i_d = i_q \) line, due to incompleteness of the training for those points. In particular, the flux linkages estimation has not meaning below zero.

The effectiveness of the magnetic model obtained online was verified by comparing the estimated flux linkages of Fig. 6 with those obtained by the benchmark offline method [3] under the same operating conditions, i.e. along the line \( 0 \leq i_d = i_q \leq I_N \). The results are reported in Fig. 7.
The errors still remain within a narrow band of 2%, except at very low currents ($i_d = i_q \leq 0.1 I_N$), where more neurons should be necessary. On the other hand, at such low values of current the SynR motor does not suffer of any saturation, so that a simple $i_d = i_q$ current control is adequate.

### C. Online adaptive MTPA control

The last test carried out on the experimental setup was related to the identification of the real MTPA condition and its online tracking, based on the algorithm explained in Sect. IV. Thanks to the light training algorithm (13), the RBF network of Fig. 1 is continuously updated, so that an accurate and comprehensive magnetic model of the SynR motor is always available and it can be used to update the MTPA curve.

The procedure can be explained with the aid of Fig. 8. An outer speed control loop is usually present and it is supposed to produce the torque reference $\tau^*$. The torque reference is made proportional to the current reference amplitude $I^*$, by a constant $K_I = T_N/I_N$ obtained from TABLE II. The phase voltages and currents $u_{abc}, i_{abc}$ are sampled at each control cycle and transformed into the synchronous reference frame by a Park transformation ($dq/abc$). Their values are used to update the RBF-based flux linkages estimations $\hat{\lambda}_{dq}$. It is worth to remark that the RBF network weights were initialised supposing known and constant inductances $L_d, L_q$ (TABLE II), so that the 3D plots $\hat{\lambda}_d, \hat{\lambda}_q$ as function of currents were two simple inclined planes. The MTPA algorithm represented by (19) and (20) is then evaluated on the actual current request and the estimated flux linkages. The algorithm updates the phase $\theta^*$ of the reference current vector, according to (19), first approaching and then maintaining the SynR motor drive in a dynamically updated MTPA condition. As an example, Fig. 9 reports the situation after the training in a single working point P. Should a transient occur, the MTPA curve would be the modified one. As testified by the output of (19), also reported in Fig. 9, the transient would be performed out of the MTPA condition except in the neighbourhood of point P. Over time, the RBF network will get completely trained and the MTPA trajectory detected by (19) will coincide with the true one. A polar-to-rectangular (P/R) transformation block returns the $i^*_{dq}$ reference vector to the current control loop, which drives the inverter-fed SynR motor.
The first experiment about adaptive MTPA was performed by applying five different load torque levels to the SynR motor drive. For each load torque level, i.e. at steady state, the adaptive MTPA algorithm ((19) and (20)) makes the current reference to leave the default $i_d = i_q$ condition, by slightly modifying the phase $\varphi^*$ of the current vector itself. In parallel, a new set of RBF network weights are obtained by training the RBF network using the acquired voltages and currents. The new complete magnetic maps are reported in Fig.10. The five MTPA points reached by the adaptive algorithm are denoted by diamond-shaped marks in Fig. 11, which also reports as a solid line the real MTPA curve, obtained with offline measurements and the aid of a torque meter (which is not an option in the real application, of course). The dashed line refers to the initial default $i_d = i_q$ condition. The accuracy is rather good, if one considers that it is obtained without any signal injection (and the related chattering problems). In this sense too, the proposed technique is innovative.

The MTPA condition (19) trusts on a correct flux linkages map. It takes time to the AC drive to get a well-trained RBF network in a sufficiently wide portion of the working plane. A second experiment was designed to evaluate the time needed to find a true MTPA point including the training of the RBF network. Anyway, in the real application it is worth noting that these two procedures (the RBF network training and the MTPA tracking) are independent, so that the former does not influence the dynamic of the latter. Thus, during torque transients the training is stopped and the MTPA condition is evaluated on the flux linkages map available at that moment.

The drive was started by imposing, with a virtual load, four different initial working points ($P_1, \ldots, A$ of Fig. 12a). Then, the adaptive MTPA algorithm described in the previous experiment was started. The real MTPA points were reached with the dynamic reported in Fig. 12b. The true MTPA curve in Fig. 12a was obtained with the aid of a torque-meter. The algorithm takes the current vector amplitude to its minimum, as shown in Fig. 12b. The convergence time can be reduced by processing the output of (19) with an additional regulator. This was not implemented because out of the scope of this work. Anyway, it is worth noting that the convergence is requested only once per point. After a while, the RBF network will get trained over the whole MPTA curve.

VI. CONCLUSION

An accurate maximum torque per ampere control allows the minimisation of the copper losses, with the related benefits in terms of ac drive efficiency. This is particularly true for SynR motors, whose marked magnetic nonlinearity makes the usual $i_d = i_q$ strategy quite imprecise. In this paper, a novel very light training algorithm made possible both the implementation and the online training of an RBF neural network. It has been found that the local property of the network makes it suitable for a precise flux linkages estimation and online tracking, starting from the measurement of motor currents and voltages. The availability of the magnetic model yielded the implementation of an adaptive MTPA control, without any signal injection. The experiments on a real SynR motor drive have clearly demonstrated the feasibility of the proposed control. To increase the reference value, mathematical insights and design hints have been included in the paper. They should help the implementation despite the complexity of the subject. A similar approach could be used for electric drives that require flux-weakening operations, by properly turning the MTPA condition (19) into a maximum torque-per-voltage condition.

ACKNOWLEDGMENT

This work was supported by Fondazione Studi Universitari Vicenza.
REFERENCES


