Published in IET Electric Power Applications
Received on 9th December 2009
Revised on 27th July 2010
doi: 10.1049/iet-epa.2009.0296

ISSN 1751-8660

Parameter estimation of an induction machine using advanced particle swarm optimisation algorithms

D.C. Huynh   M.W. Dunnigan

Department of Electrical, Electronic and Computer Engineering, Heriot Watt University, EH14 4AS, UK
E-mail: m.w.dunnigan@hw.ac.uk

Abstract: This study proposes a new application of two advanced particle swarm optimisation (PSO) algorithms for parameter estimation of an induction machine (IM). The inertia weight, cognitive and social parameters and two independent random sequences are the main parameters of the standard PSO algorithm which affect the search characteristics, convergence capability and solution quality in a particular application. Two advanced PSO algorithms, known as the dynamic particle swarm optimisation (dynamic PSO) and chaos PSO algorithms modify those parameters to improve the performance of the standard PSO algorithm. The algorithms use the measurements of the three-phase stator currents, voltages and the speed of the IM as the inputs to the parameter estimator. The experimental results obtained compare the estimated parameters with the IM parameters achieved using traditional tests such as the dc, no-load and locked-rotor tests. There is also a comparison of the solution quality between a genetic algorithm (GA), standard PSO, dynamic PSO and chaos PSO algorithms. The results show that the dynamic PSO and chaos PSO algorithms are better than the standard PSO algorithm and GA for parameter estimation of the IM.

1 Introduction

Induction machines (IMs) are widely used in ac drive systems because of desirable features such as low cost, ruggedness, ease of maintenance and directly start ability. In most high-performance IM applications, accurate knowledge of the IM parameters is necessary. This directly affects the operational and control characteristics of the IM. Usually, these parameters are provided by the manufacturer or estimated by traditional tests such as the dc, no-load and locked-rotor tests. However, manufacturers may not supply all the parameter information. The traditional tests for obtaining the IM parameters are as follows. The stator resistance is directly calculated from the dc test by supplying a dc voltage to the IM stator windings. The locked-rotor test and the IM classification information allow calculation of the stator and rotor leakage reactances. In the case where the IM classification information is not known, the stator and rotor leakage reactances are assumed to be of equal value. Additionally, the rotor resistance is derived from this test as well. Eventually, the IM magnetising reactance is determined by performing the no-load test. In this case, a dc power source, a synchronous generator coupled with a dc motor and an auto transformer are usually necessary to perform the dc, no-load and locked-rotor tests at various frequencies and to control the input voltage to the stator [1]. For these reasons, a number of parameter estimation schemes have been proposed recently. This is a difficult problem because the mathematical model of an IM is non-linear and there are several state variables which cannot be measured directly, such as the rotor flux. However, this problem has been resolved using optimisation techniques such as the genetic algorithm (GA) [2–4], a local search algorithm (LSA), a simulated annealing (SA) approach and an evolution strategy (ES) [5]. The parameter estimation objective is transformed into an optimisation problem. In [2, 5–7], the parameter estimation problem is considered during the start-up phase using the non-linear differential equations. In [4], a mathematical model based on three torque functions of an IM which are the full load, locked-rotor and breakdown torques of an equivalent circuit is used for parameter estimation. In [8], the transient operation of an IM from standstill to a steady-state speed for a set period followed by successive free motion to standstill is considered. In [3], parameter estimation is performed using various machine load tests on the steady-state equivalent circuit. Simulations, rather than experimental tests, were performed to estimate the IM parameters in [2–8]. Recently, a particle swarm optimisation (PSO) algorithm has been introduced as one of the optimisation techniques used for parameter estimation. This algorithm is simpler and easier to implement than other evolutionary algorithms (GA, LSA, SA or ES) [2–5], as it only has a few parameters to adjust. So far, there have been researchers who have used the PSO algorithm with various variants for parameter estimation such as diversity-guided PSO algorithm [5], stretching PSO algorithm [8], PSO algorithm with a constriction factor [6] and PSO algorithm with a time-varying inertia weight [7]. Recently, other advanced variants of the PSO algorithm have been introduced such as a dynamic particle swarm optimisation (dynamic PSO) algorithm with time-varying cognitive and social parameters [9–11] and chaos PSO (chaos PSO) algorithm [10, 12–15]. This paper proposes a new application of the dynamic PSO and chaos PSO algorithms for parameter estimation of the IM. The experimental results of the estimated parameters using the GA, standard PSO, dynamic PSO and chaos PSO algorithms are compared with the parameters obtained from traditional tests such as dc, no-load and locked-rotor tests. Our method for estimation requires only one test with the machine running from a three-phase supply under steady-state conditions with the stator currents, stator voltages and rotor speed requiring measurement.

The remainder of this paper is organised as follows. The mathematical model of the parameter estimation problem is described in Section 2. A novel proposal using the dynamic PSO and chaos PSO algorithms for parameter estimation of an IM is presented in Section 3. The practical experiments and results then follow to confirm the validity of the proposed application in Section 4. Finally, the advantages of the novel application are summarised through comparison with several related existing approaches.

2 Parameter estimation of an IM

2.1 IM model

The dynamic model of an IM in the synchronously rotating reference frame is given as follows [16]

$$v_{qs} = R_s i_{qs} + \omega_e (L_s i_{ds} + L_m i_{dr}) + L_s \frac{d}{dt} i_{qs} + L_m \frac{d}{dt} i_{qr} \quad (1)$$

$$v_{ds} = R_s i_{ds} - \omega_e (L_s i_{qs} + L_m i_{qr}) + L_s \frac{d}{dt} i_{ds} + L_m \frac{d}{dt} i_{dr} \quad (2)$$

$$0 = R_r i_{qr} + (\omega_e - \omega_r)(L_r i_{dr} + L_m i_{ds}) + L_r \frac{d}{dt} i_{qr} + L_m \frac{d}{dt} i_{qs} \quad (3)$$

$$0 = R_r i_{dr} - (\omega_e - \omega_r)(L_r i_{qr} + L_m i_{qs}) + L_r \frac{d}{dt} i_{dr} + L_m \frac{d}{dt} i_{ds} \quad (4)$$

where v_{ds}, v_{qs}, i_{ds} and i_{qs} are the $d-q$ axis stator voltages and currents in the two-phase d_e-q_e synchronously rotating reference frame. i_{dr} and i_{qr} are the $d-q$ axis rotor currents in the two-phase d_e-q_e synchronously rotating reference frame. R_s and R_r are the stator and rotor resistances. L_s and L_r are the stator and rotor inductances. L_m is the magnetising inductance. ω_e is the synchronous speed. ω_r is the rotor electrical speed.

This model was simplified using the axes transformation [16] from the three-phase $a-b-c$ stationary reference frame to the two-phase d_s-q_s stationary reference frame and then to the two-phase d_e-q_e synchronously rotating reference frame. These are described through the voltage transformations as follows.

The transformation of the three-phase $a-b-c$ stationary reference frame to the two-phase d_s-q_s stationary reference frame is

$$v_{qs}^s = \frac{2}{3} v_a - \frac{1}{3} v_b - \frac{1}{3} v_c \quad (5)$$

$$v_{ds}^s = -\frac{1}{\sqrt{3}} v_b + \frac{1}{\sqrt{3}} v_c \quad (6)$$

where v_a, v_b and v_c are the three-phase stator voltages. v_{ds}^s and v_{qs}^s: the $d-q$-axis stator voltages in the two-phase d_s-q_s stationary reference frame. The transformation of the two-phase d_s-q_s stationary reference frame to the two-phase d_e-q_e synchronously rotating reference frame is

$$v_{qs}^e = v_{qs}^s \cos\theta_e - v_{ds}^s \sin\theta_e \quad (7)$$

$$v_{ds}^e = v_{qs}^s \sin\theta_e + v_{ds}^s \cos\theta_e \quad (8)$$

where v_{ds}^e and v_{qs}^e: the $d-q$ axis stator voltages in the two-phase d_e-q_e synchronously rotating reference frame. θ_e: the synchronous angle.

The parameter estimation problem in this paper is considered under steady-state conditions, and thus the IM model becomes (note that the superscript e has been removed in the following equations for convenience)

$$v_{qs} = R_s i_{qs} + \omega_e (L_s i_{ds} + L_m i_{dr}) \quad (9)$$

$$v_{ds} = R_s i_{ds} - \omega_e (L_s i_{qs} + L_m i_{qr}) \quad (10)$$

$$0 = R_r i_{qr} + (\omega_e - \omega_r)(L_r i_{dr} + L_m i_{ds}) \quad (11)$$

$$0 = R_r i_{dr} - (\omega_e - \omega_r)(L_r i_{qr} + L_m i_{qs}) \quad (12)$$

From (9) to (12), the IM model is derived as follows

$$\boldsymbol{v}_{qs} = \left(\boldsymbol{R}_s + \omega_e \frac{\boldsymbol{L}_m^2}{A\boldsymbol{L}_r}\right) i_{qs} + \left(\omega_e \boldsymbol{L}_s - \omega_e(\omega_e - \omega_r)\frac{\boldsymbol{L}_m^2}{A\boldsymbol{R}_r}\right) i_{ds} \tag{13}$$

$$\boldsymbol{v}_{ds} = -\left(\omega_e \boldsymbol{L}_s + \frac{\omega_e}{(\omega_e - \omega_r)}\frac{\boldsymbol{R}_r \boldsymbol{L}_m^2}{A\boldsymbol{L}_r^2} - \omega_e \frac{\boldsymbol{L}_m^2}{\boldsymbol{L}_r}\right) i_{qs} + \left(\boldsymbol{R}_s + \omega_e \frac{\boldsymbol{L}_m^2}{A\boldsymbol{L}_r}\right) i_{ds} \tag{14}$$

where

$$A = \frac{\boldsymbol{R}_r}{\boldsymbol{L}_r(\omega_e - \omega_r)} + \frac{\boldsymbol{L}_r(\omega_e - \omega_r)}{\boldsymbol{R}_r} \tag{15}$$

From (13)–(14), the d–q axis stator currents in the two-phase d_e–q_e synchronously rotating reference frame are given as follows

$$i_{qs} = \frac{a_{22}v_{qs} - a_{12}v_{ds}}{a_{11}a_{22} - a_{12}a_{21}} \tag{16}$$

$$i_{ds} = \frac{a_{21}v_{qs} - a_{11}v_{ds}}{a_{12}a_{21} - a_{11}a_{22}} \tag{17}$$

where

$$a_{11} = \left(\boldsymbol{R}_s + \omega_e \frac{\boldsymbol{L}_m^2}{A\boldsymbol{L}_r}\right) \tag{18}$$

$$a_{12} = \left(\omega_e \boldsymbol{L}_s - \omega_e(\omega_e - \omega_r)\frac{\boldsymbol{L}_m^2}{A\boldsymbol{R}_r}\right) \tag{19}$$

$$a_{21} = -\left(\omega_e \boldsymbol{L}_s + \frac{\omega_e}{(\omega_e - \omega_r)}\frac{\boldsymbol{R}_r \boldsymbol{L}_m^2}{A\boldsymbol{L}_r^2} - \omega_e \frac{\boldsymbol{L}_m^2}{\boldsymbol{L}_r}\right) \tag{20}$$

$$a_{22} = \left(\boldsymbol{R}_s + \omega_e \frac{\boldsymbol{L}_m^2}{A\boldsymbol{L}_r}\right) \tag{21}$$

2.2 Parameter estimation

The parameter estimation technique used in this paper is based on the output error method [17] which compares the response between the real system and an estimated parameter model using the same inputs. The response of the real system is as follows

$$\boldsymbol{y} = g(\boldsymbol{\theta}, \boldsymbol{u}) \tag{22}$$

where $\boldsymbol{y}$, $\boldsymbol{\theta}$ and $\boldsymbol{u}$ are the output, parameter and input vectors of the actual system.

The estimated parameter model has the same structure as the real system.

$$\hat{\boldsymbol{y}} = g(\hat{\boldsymbol{\theta}}, \boldsymbol{u}) \tag{23}$$

where $\hat{\boldsymbol{y}}$ and $\hat{\boldsymbol{\theta}}$ are the estimated output and parameter vectors.

A performance function (or fitness function) is then determined based on the real system response $\boldsymbol{y}$ and the estimated parameter model response $\hat{\boldsymbol{y}}$ that is quadratic in a simple case.

Fitness function

$$F(\hat{\boldsymbol{\theta}}) = \frac{1}{N}\sum_{t=1}^{N} \|\boldsymbol{y} - \hat{\boldsymbol{y}}\|^2 = \frac{1}{N}\sum_{t=1}^{N} \|g(\boldsymbol{\theta}, \boldsymbol{u}) - g(\hat{\boldsymbol{\theta}}, \boldsymbol{u})\|^2 \tag{24}$$

where N is the series of measurement samples.

The fitness function $F(\hat{\boldsymbol{\theta}})$ depends on $\hat{\boldsymbol{\theta}}$ and obtains its minimum, zero, when $\hat{\boldsymbol{\theta}} = \boldsymbol{\theta}$. In this case, the parameter estimation problem is considered as the following optimisation problem

$$\underset{\hat{\boldsymbol{\theta}}}{\text{Min}}\, F(\hat{\boldsymbol{\theta}}) \tag{25}$$

This paper assumes that $\boldsymbol{L}_s = \boldsymbol{L}_r$ [1] and the real system vectors are then defined as follows

$$\boldsymbol{y} = [\, i_{qs} \quad i_{ds} \,] \tag{26}$$

$$\boldsymbol{\theta} = [\, \boldsymbol{R}_s \quad \boldsymbol{R}_r \quad \boldsymbol{L}_s \quad \boldsymbol{L}_m \,] \tag{27}$$

$$\boldsymbol{u} = [\, \boldsymbol{v}_{qs} \quad \boldsymbol{v}_{ds} \,] \tag{28}$$

where i_{qs} and i_{ds} are the currents which are sampled and recorded from the IM system. $\boldsymbol{R}_s$, $\boldsymbol{R}_r$, $\boldsymbol{L}_s$ and $\boldsymbol{L}_m$ are the actual parameters of the IM. $\boldsymbol{v}_{qs}$ and $\boldsymbol{v}_{ds}$: the input voltages for both the actual system and the estimated parameter model.

The estimated system vectors are defined as

$$\hat{\boldsymbol{y}} = \left[\, \hat{i}_{qs} \quad \hat{i}_{ds} \,\right] \tag{29}$$

$$\hat{\boldsymbol{\theta}} = [\, \hat{\boldsymbol{R}}_s \quad \hat{\boldsymbol{R}}_r \quad \hat{\boldsymbol{L}}_s \quad \hat{\boldsymbol{L}}_m \,] \tag{30}$$

where $\hat{i}_{qs}$ and $\hat{i}_{ds}$: the estimated currents which are calculated using (16)–(17). $\hat{\boldsymbol{R}}_s$, $\hat{\boldsymbol{R}}_r$, $\hat{\boldsymbol{L}}_s$ and $\hat{\boldsymbol{L}}_m$: the estimated parameters of the IM.

Thus, the fitness function for the parameter estimation problem is rewritten as follows

$$F(\hat{\boldsymbol{\theta}}) = \frac{1}{N}\sum_{t=1}^{N} \|(i_{qs} - \hat{i}_{qs})\|^2 + \|(i_{ds} - \hat{i}_{ds})\|^2 \qquad (31)$$

This fitness function is non-linear and has many local optima. Obviously, it is difficult to determine the global optimum for the parameter estimation. In order to resolve this problem, this paper proposes using a stochastic optimisation technique. The dynamic PSO and chaos PSO algorithms are two advanced variants of the PSO algorithm which are one of the relatively new stochastic optimisation techniques. These algorithms are presented and then applied to the parameter estimation problem.

3 Dynamic and chaos PSO algorithms

The standard PSO algorithm is reviewed in Section 3.1 followed by descriptions of two advanced PSO algorithms: the dynamic PSO and chaos PSO algorithms in Sections 3.2 and 3.3 of this section, respectively.

3.1 PSO algorithm

The PSO algorithm is a population-based stochastic optimisation method which was developed by Kennedy and Eberhart in 1995 [18]. The algorithm was inspired by the social behaviours of bird flocks, colonies of insects, schools of fishes and herds of animals. The algorithm starts by initialising a population of random solutions called particles and searches for optima by updating generations through the following velocity and position update equations.

The velocity update equation

$$\begin{aligned}\boldsymbol{v}_i(k+1) = w\boldsymbol{v}_i(k) + c_1r_1(pbest_i(k) - \boldsymbol{x}_i(k))\\ + c_2r_2(gbest(k) - \boldsymbol{x}_i(k))\end{aligned} \qquad (32)$$

The position update equation

$$\boldsymbol{x}_i(k+1) = \boldsymbol{x}_i(k) + \boldsymbol{v}_i(k+1) \qquad (33)$$

where $\boldsymbol{v}_i(k)$ is the kth current velocity of the ith particle. $\boldsymbol{x}_i(k)$ is the kth current position of the ith particle. k is the kth current iteration of the algorithm, $\mathbf{1} \le k \le n$. n is the predefined maximum iteration number. i is the ith particle of the swarm, $\mathbf{1} \le i \le M$. M is the particle number of the swarm. Usually, $\boldsymbol{v}_i$ is clamped in the range $[-v_{\max}, v_{\max}]$ to reduce the likelihood that a particle might leave the search space. In this case, if the search space is defined by the bounds $[-x_{\max}, x_{\max}]$ then the $v_{\max}$ value will be typically set so that $v_{\max} = mx_{\max}$, where $0.1 \le m \le 1.0$ [19]. $pbest_i(k)$ is the best position found by the ith particle (personal best). $gbest(k)$ is the best position found by a swarm (global best, best of the personal bests). c_1 and c_2 are the cognitive and social parameters. c_2 regulates the step size in the direction of a global best particle and c_1 regulates the step size in the direction of a personal best position of that particle, c_1 and $c_2 \in [0, 2]$. With large cognitive and small social parameters at the beginning, particles are allowed to move around a wider search space instead of moving towards a population best. On the other hand, with small cognitive and large social parameters, particles are allowed to converge to the global optimum in the latter part of optimisation [9]. r_1 and r_2 are two independent random sequences which are used to influence the stochastic nature of the algorithm, $r_1 \in \mathrm{U}(0, 1)$ and $r_2 \in U(0, 1)$. w is the inertia weight [20].

The velocity update equation of the particle is considered as three parts: the first part is the previous velocity of the particle, $w\boldsymbol{v}_i(k)$; the second and the third parts, $c_1r_1(pbest_i(k){-}\boldsymbol{x}_i(k))$ and $c_2r_2(gbest(k){-}\boldsymbol{x}_i(k))$, contribute to the particle velocity change.

Without the first part of the velocity update equation, the particles' velocities are only determined by their current and best history positions and the PSO algorithm search process is similar to an LSA. Thus, the particles tend to move towards the same position and the final solution depends heavily on the initial population. The PSO algorithm only finds out the final solution when the initial search space includes the global optimum. By adding the first part, the particles have a tendency to expand the search space and explore the new area. Owing to this, the PSO algorithm becomes a global search algorithm. Nevertheless, for each problem, there is always a different trade-off between the local and global search abilities. This is why the inertia weight is used in the first part [20]. This value was set to 1 in the original PSO algorithm [18]. Shi and Eberhart investigated the effect of w values in the range [0, 1.4] as well as in a linear time-varying domain. Their results indicated that choosing $w \in [0.9, 1.2]$ results in faster convergence [20]. A larger inertia weight facilitates a global exploration and a smaller inertia weight tends to facilitate a local exploration [10]. Therefore careful choice of the inertia weight w during the evolution process of the PSO algorithm is necessary. This improves the convergence capability and search performance of the algorithm.

The two remaining parts of the velocity update equation also play an important role in updating the new velocities of the particles. The term $(pbest_i(k){-}\boldsymbol{x}_i(k))$ is the distance of its own best position from its current position, whereas the term $(gbest(k){-}\boldsymbol{x}_i(k))$ is the distance of the best position in the swarm from its current position. Without the second and third parts, the particles will keep their current speed in the same direction until they hit the boundary [20]. This affects the algorithm performance during the evolution process.

Eventually, the particle flies towards a new position according to the position update (33) using the previous position and new velocity of the particle.

The performance of each particle is based on a predefined fitness function which is related to the particular application (31).

There are four parameters R_s, R_r, L_s and L_m which need to be estimated in this parameter estimation application. Each particle is treated as a point in a four-dimensional space. For the standard PSO algorithm, the ith particle is represented as $\{\boldsymbol{R_{si}}, \boldsymbol{R_{ri}}, \boldsymbol{L_{si}}, \boldsymbol{L_{mi}}\}$.

$$[\boldsymbol{R_{si}}\ \ \boldsymbol{R_{ri}}\ \ \boldsymbol{L_{si}}\ \ \boldsymbol{L_{mi}}] = \left.\underbrace{\begin{bmatrix} \boldsymbol{R}_{s1}(k) & \dots & \boldsymbol{L}_{m1}(k) \\ \boldsymbol{R}_{s2}(k) & \dots & \boldsymbol{L}_{m2}(k) \\ \dots & \dots & \dots \\ \dots & \dots & \dots \\ \boldsymbol{R}_{s(M-1)}(k) & \dots & \boldsymbol{L}_{m(M-1)}(k) \\ \boldsymbol{R}_{sM}(k) & \dots & \boldsymbol{L}_{mM}(k) \end{bmatrix}}_{4}\right\}M \tag{34}$$

Using the voltages $\boldsymbol{v}_{ds}$, $\boldsymbol{v}_{qs}$ and parameter vector $\{\boldsymbol{R_{si}}(k), \boldsymbol{R_{ri}}(k), \boldsymbol{L_{si}}(k), \boldsymbol{L_{mi}}(k)\}$, the $d-q$ axis stator currents in the two-phase d_e-q_e synchronously rotating reference frame of the estimated parameter model, $\hat{i}_{qs}$ and $\hat{i}_{ds}$ are calculated using (16)–(17), whereas the $d-q$ axis stator currents in the two-phase d_e-q_e synchronously rotating reference frame of the real system, i_{qs} and i_{ds} are sampled and recorded from the IM system. The fitness function (31) is then used together with $\hat{i}_{qs}$, $\hat{i}_{ds}$, i_{qs} and i_{ds} to search the best position for the ith particle and the best position of the swarm. The best position found for the ith particle is represented as $\{pbest_{Rsi}(k), pbest_{Rri}(k), pbest_{Lsi}(k), pbest_{Lmi}(k)\}$. The best position found by the swarm is represented as $\{gbest_{Rs}(k), gbest_{Rr}(k), gbest_{Ls}(k), gbest_{Lm}(k)\}$. The update mechanism of the personal best is that if the fitness value of the parameter vector $\{\boldsymbol{R_{si}}(k), \boldsymbol{R_{ri}}(k), \boldsymbol{L_{si}}(k), \boldsymbol{L_{mi}}(k)\}$ is better than that of $\{pbest_{Rsi}(k-1), pbest_{Rri}(k-1), pbest_{Lsi}(k-1), pbest_{Lmi}(k-1)\}$ then the parameter vector $\{\boldsymbol{R_{si}}(k), \boldsymbol{R_{ri}}(k), \boldsymbol{L_{si}}(k), \boldsymbol{L_{mi}}(k)\}$ is set to $\{pbest_{Rsi}(k), pbest_{Rri}(k), pbest_{Lsi}(k), pbest_{Lmi}(k)\}$. For the update mechanism of the global best, if the fitness value of the parameter vector $\{\boldsymbol{R_{si}}(k), \boldsymbol{R_{ri}}(k), \boldsymbol{L_{si}}(k), \boldsymbol{L_{mi}}(k)\}$ is better than that of $\{gbest_{Rs}(k-1), gbest_{Rr}(k-1), gbest_{Ls}(k-1), gbest_{Lm}(k-1)\}$ then the updated global best, $\{gbest_{Rs}(k), gbest_{Rr}(k), gbest_{Ls}(k), gbest_{Lm}(k)\}$ is the parameter vector, $\{\boldsymbol{R_{si}}(k), \boldsymbol{R_{ri}}(k), \boldsymbol{L_{si}}(k), \boldsymbol{L_{mi}}(k)\}$.

The rate of the position change is the velocity for the ith particle which is represented as $\{\boldsymbol{v_{Rsi}}, \boldsymbol{v_{Rri}}, \boldsymbol{v_{Lsi}}, \boldsymbol{v_{Lmi}}\}$

$$\begin{aligned}&[\boldsymbol{v}_{Rsi}\ \ \boldsymbol{v}_{Rri}\ \ \boldsymbol{v}_{Lsi}\ \ \boldsymbol{v}_{Lmi}] \\ &= \left.\underbrace{\begin{bmatrix} \boldsymbol{v}_{Rs1}(k) & \dots & \boldsymbol{v}_{Lm1}(k) \\ \boldsymbol{v}_{Rs2}(k) & \dots & \boldsymbol{v}_{Lm2}(k) \\ \dots & \dots & \dots \\ \dots & \dots & \dots \\ \boldsymbol{v}_{Rs(M-1)}(k) & \dots & \boldsymbol{v}_{Lm(M-1)}(k) \\ \boldsymbol{v}_{RsM}(k) & \dots & \boldsymbol{v}_{LmM}(k) \end{bmatrix}}_{4}\right\}M\end{aligned} \tag{35}$$

This vector is calculated using (32) and simultaneously the position of the ith particle is updated through (33) as well.

In this application, the positions and velocities of the ith particle are random sequences which are limited in the ranges $[x_{\min}, x_{\max}]$ and $[v_{\min}, v_{\max}]$, respectively. This means that

$$\begin{cases} \boldsymbol{R}_{si(\min)} \le \boldsymbol{R}_{si} \le \boldsymbol{R}_{si(\min)} \\ \boldsymbol{R}_{ri(\min)} \le \boldsymbol{R}_{ri} \le \boldsymbol{R}_{ri(\min)} \\ \boldsymbol{L}_{si(\min)} \le \boldsymbol{L}_{si} \le \boldsymbol{L}_{si(\min)} \\ \boldsymbol{L}_{mi(\min)} \le L_{mi} \le \boldsymbol{L}_{mi(\min)} \end{cases} \tag{36}$$

and

$$\begin{cases} \boldsymbol{v}_{Rsi(\min)} \le \boldsymbol{v}_{Rsi} \le \boldsymbol{v}_{Rsi(\min)} \\ \boldsymbol{v}_{Rri(\min)} \le \boldsymbol{v}_{Rri} \le \boldsymbol{v}_{Rri(\min)} \\ \boldsymbol{v}_{Lsi(\min)} \le \boldsymbol{v}_{Lsi} \le \boldsymbol{v}_{Lsi(\min)} \\ \boldsymbol{v}_{Lmi(\min)} \le \boldsymbol{v}_{Lmi} \le \boldsymbol{v}_{Lmi(\min)} \end{cases} \tag{37}$$

These conditions strongly depend on the particular problem and the user's experience, which affect the number of iterations needed as well as the convergence speed of the algorithm.

The inertia weight w is set to 0.9 in this application. The two independent random sequences, r_1 and r_2 are uniformly distributed in $U(0, 1)$.

After each iteration, the position and velocity vectors of the particles are as follows

$$\begin{aligned}&[\boldsymbol{R}_{si}\ \ \boldsymbol{R}_{ri}\ \ \boldsymbol{L}_{si}\ \ \boldsymbol{L}_{mi}] \\ &= \left.\underbrace{\begin{bmatrix} \boldsymbol{R}_{s1}(k+1) & \dots & \boldsymbol{L}_{m1}(k+1) \\ \boldsymbol{R}_{s2}(k+1) & \dots & \boldsymbol{L}_{m2}(k+1) \\ \dots & \dots & \dots \\ \dots & \dots & \dots \\ \boldsymbol{R}_{s(M-1)}(k+1) & \dots & \boldsymbol{L}_{m(M-1)}(k+1) \\ \boldsymbol{R}_{sM}(k+1) & \dots & \boldsymbol{L}_{mM}(k+1) \end{bmatrix}}_{4}\right\}M\end{aligned} \tag{38}$$

$$\begin{bmatrix} \boldsymbol{v}_{Rsi} & \boldsymbol{v}_{Rri} & \boldsymbol{v}_{Lsi} & \boldsymbol{v}_{Lmi} \end{bmatrix}$$

$$= \left.\underbrace{\begin{bmatrix} \boldsymbol{v}_{Rs1}(k+1) & \dots & \boldsymbol{v}_{Lm1}(k+1) \\ \boldsymbol{v}_{Rs2}(k+1) & \dots & \boldsymbol{v}_{Lm2}(k+1) \\ \dots & \dots & \dots \\ \dots & \dots & \dots \\ \boldsymbol{v}_{Rs(M-1)}(k+1) & \dots & \boldsymbol{v}_{Lm(M-1)}(k+1) \\ \boldsymbol{v}_{RsM}(k+1) & \dots & \boldsymbol{v}_{LmM}(k+1) \end{bmatrix}}_{4}\right\} M \tag{39}$$

This process is repeated until the user-defined end criterion is satisfied. In this parameter estimation problem, the end criterion is that the kth current iteration number reaches the maximum iteration number, n.

The solution of the parameter estimation is eventually

$$\begin{bmatrix} \hat{\boldsymbol{R}}_s & \hat{\boldsymbol{R}}_r & \hat{\boldsymbol{L}}_s & \hat{\boldsymbol{L}}_m \end{bmatrix} = \begin{bmatrix} gbest_{Rs}(n) & gbest_{Rr}(n) & \dots \\ \dots & gbest_{Ls}(n) & gbest_{Lm}(n) \end{bmatrix} \tag{40}$$

The standard PSO algorithm is obviously one of the simplest and most efficient global optimisation algorithms, especially in solving discontinuous, multimodal and non-convex problems. However, for local optima problems, the particles sometimes become trapped in undesired states during the evolution process which leads to a loss of the exploration abilities. Because of this disadvantage, premature convergence can take place in the standard PSO algorithm which affects the performance of the evolution process. This is one of the major drawbacks of the standard PSO algorithm.

In order to improve the evolution process performance of the standard PSO algorithm, the dynamic PSO and chaos PSO algorithms are presented in the next section and applied for parameter estimation of the IM.

3.2 Dynamic PSO algorithm

A dynamic PSO algorithm is one of the standard PSO algorithm variants which was introduced in [9] with time-varying cognitive and social parameters. For most of the population-based optimisation techniques, it is desirable to encourage the individuals to wander through the entire search space without clustering around local optima during the early stages of optimisation, as well as being important to enhance convergence towards the global optimum during the latter stages. The second part of the velocity update equation (32) is known as the cognitive component which represents the personal thinking of each particle. The cognitive component encourages the particles to move towards their own best positions, whereas the third part of the velocity update equation is known as the social component which represents the collaborative effect of the particles in the global optimal solution search. The social component always pulls the particles towards the global best particle [9]. Thus, it is obvious that the cognitive and social parameters in the velocity update equation are two of the parameters which support the algorithm to satisfy the requirements of enhancing the performance in the early and latter stages. Proper control of these two parameters is important to find the optimal solution accurately and efficiently. Using the modification of the cognitive and social parameters, the algorithm improves the global search capability of the particles in the early stage of the optimisation process and then directs particles to the global optimum at the end stage so that the convergence capability of the search process is enhanced. To achieve this, large cognitive and small social parameters are used at the beginning and small cognitive and large social parameters are used at the latter stage. The mathematical representation of this modification is given as follows [9]

$$\begin{aligned} \boldsymbol{v}_i(k+1) &= w\boldsymbol{v}_i(k) + c_1(k)r_1(pbest_i(k) - \boldsymbol{x}_i(k)) \\ &\quad + c_2(k)r_2(gbest(k) - \boldsymbol{x}_i(k)) \\ &\quad 1 \le i \le M \quad \text{and} \quad 1 \le k \le n \end{aligned} \tag{41}$$

where

$$c_1(k) = (c_{1\text{final}} - c_{1\text{initial}})\frac{k}{n} + c_{1\text{initial}} \tag{42}$$

$$c_2(k) = (c_{2\text{final}} - c_{2\text{initial}})\frac{k}{n} + c_{2\text{initial}} \tag{43}$$

$c_1(k)$ and $c_2(k)$ are the time-varying cognitive and social parameters. $c_{1\text{initial}}$ and $c_{1\text{final}}$ are the initial and final values, respectively, of the cognitive parameter. $c_{2\text{initial}}$ and $c_{2\text{final}}$ are the initial and final values, respectively, of the social parameter.

The dynamic PSO algorithm is applied for parameter estimation of the IM where the position and velocity of the ith particle are updated using (33) and (41), respectively. The velocity update equation uses the time-varying cognitive and social parameters. The parameter $c_1(k)$ is set to decrease linearly with $c_{1\text{initial}} = 2.5$ and $c_{1\text{final}} = 0.5$ during a run, whereas the parameter $c_2(k)$ is set to increase linearly $c_{2\text{initial}} = 0.5$ and $c_{2\text{final}} = 2.5$.

Thus, the cognitive parameter is large and the social parameter is small at the beginning. This enhances the global search capability in the early part of the optimisation process. Then, the cognitive parameter is decreased linearly and the social parameter is increased linearly until at the end of the search, the particles are encouraged to converge towards the global optimum with small cognitive and large social parameters. This modification improves the evolution process performance and overcomes premature convergence of the standard PSO algorithm. The initial positions and velocities of the ith particle are also initialised as random sequences which represent the estimated parameters as $\{\boldsymbol{R}_{si},$

$\boldsymbol{R}_{ri}$, $\boldsymbol{L}_{si}$, $\boldsymbol{L}_{mi}$} in this case. These parameters are updated using (41) with the velocity vector {$\boldsymbol{v}_{Rsi}$, $\boldsymbol{v}_{Rri}$, $\boldsymbol{v}_{Lsi}$, $\boldsymbol{v}_{Lmi}$}, respectively. The limitation range of the positions and velocities of the particles are [$\boldsymbol{x}_{\min}$, $\boldsymbol{x}_{\max}$] and [$\boldsymbol{v}_{\min}$, $\boldsymbol{v}_{\max}$], respectively, as defined similarly in (36) and (37). In this application, the inertia weight, w is set to 0.9; the two independent random sequences, r_1 and r_2 are uniformly distributed in $U(0, 1)$. The search process for the best position of the ith particle {$pbest_{Rsi}(k)$, $pbest_{Rri}(k)$, $pbest_{Lsi}(k)$, $pbest_{Lmi}(k)$} and the best position over the swarm {$gbest_{Rs}(k)$, $gbest_{Rr}(k)$, $gbest_{Ls}(k)$, $gbest_{Lm}(k)$} is implemented using the fitness function (31) and the personal best and global best update mechanism as well. This is stopped at the nth maximum iteration and the estimated parameters are as in (40). The flow chart of the standard PSO and dynamic PSO algorithms is described in Fig. 1.

3.3 Chaos PSO algorithm

In addition to the dynamic PSO algorithm, this paper proposes another novel application of a chaos PSO algorithm for parameter estimation of the IM which is also more efficient compared to the standard PSO algorithm. The chaos PSO algorithm is a combination between the standard PSO algorithm and a chaotic map which was presented in [10, 12–15].

Chaos is a common phenomenon in non-linear systems, which includes infinite unstable period motions. It is a stochastic and unpredictable process in a deterministic non-linear system.

A chaotic map is a discrete-time dynamical system [10] which is given as follows

$$x_k = f(x_{(k-1)}) \tag{44}$$

where

$$x_k \in (0, 1), \; k = 1, 2, \ldots$$

The sequences are generated by using one of the chaotic maps known as chaotic sequences. These sequences have the characteristics of the chaotic map such as randomness, ergodicity and regularity, so that no state is repeated. The chaotic sequences are recently considered as sources of random sequences which can be adopted instead of normally generated random sequences.

For the standard PSO algorithm, one of its main disadvantages is premature convergence, especially in local optima problems. Thus, in order to overcome this, the algorithm parameter sequences with a randomness-based choice are substituted by the chaotic sequences which are generated from a chaotic map. In this case, the chaotic sequences are obviously an appropriate tool to support the standard PSO algorithm so that it avoids getting stuck in a local optimum during the search process and overcomes the premature convergence phenomenon present in the standard PSO algorithm. There are many chaotic maps which have been introduced and used to improve the standard PSO algorithm [10]. Among them, the logistic map is one of the simplest and easiest maps to employ in the chaos PSO algorithm for parameter estimation of the IM.

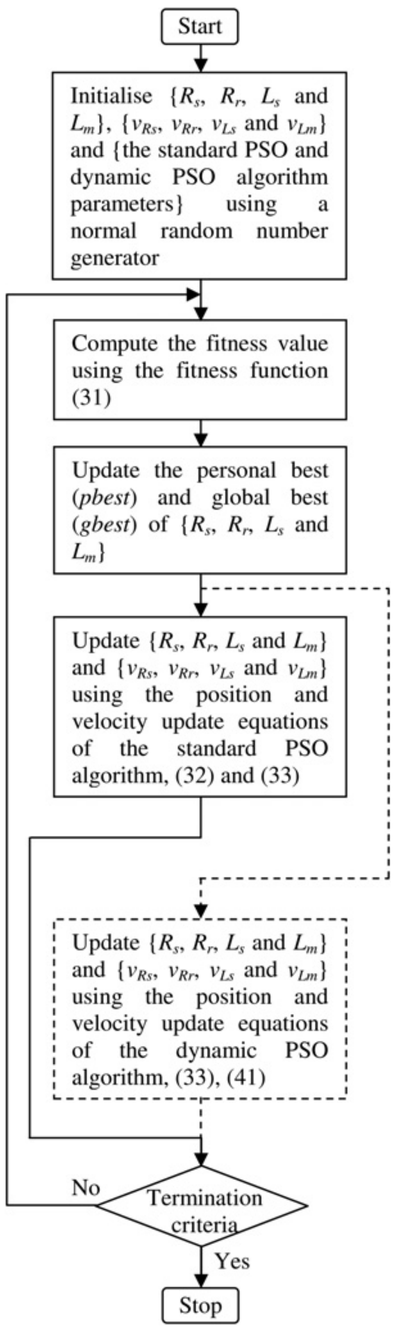
Start
Initialise {R_s, R_r, L_s and L_m}, {v_{Rs}, v_{Rr}, v_{Ls} and v_{Lm}} and {the standard PSO and dynamic PSO algorithm parameters} using a normal random number generator
Compute the fitness value using the fitness function (31)
Update the personal best (pbest) and global best (gbest) of {R_s, R_r, L_s and L_m}
Update {R_s, R_r, L_s and L_m} and {v_{Rs}, v_{Rr}, v_{Ls} and v_{Lm}} using the position and velocity update equations of the standard PSO algorithm, (32) and (33)
Update {R_s, R_r, L_s and L_m} and {v_{Rs}, v_{Rr}, v_{Ls} and v_{Lm}} using the position and velocity update equations of the dynamic PSO algorithm, (33), (41)
No
Termination criteria
Yes
Stop

Figure 1 *Flow chart of the standard PSO and dynamic PSO (dashed line) algorithms*

A logistic map is given as follows

$$X_k = aX_{(k-1)}(1 - X_{(k-1)}) \quad k = 1, 2, \ldots \tag{45}$$

where X_k is the kth chaotic number, $X_k \in (0, 1)$ with the following initial conditions: X_0 is a random number in the interval of (0, 1) and $X_0 \notin \{0.0, 0.25, 0.5, 0.75, 1.0\}$. a is the control parameter, usually set to 4 in the experiments [10].

The logistic map is used in the parameter estimation application for initialising the positions $\{\boldsymbol{R}_{si}, \boldsymbol{R}_{ri}, \boldsymbol{L}_{si}, \boldsymbol{L}_{mi}\}$ and velocities $\{\boldsymbol{v}_{Rsi}, \boldsymbol{v}_{Rri}, \boldsymbol{v}_{Lsi}, \boldsymbol{v}_{Lmi}\}$ of the ith particle as follows

$$\begin{cases} \boldsymbol{R}_{si}(1) = a\boldsymbol{R}_{s(i-1)}(1)(1 - \boldsymbol{R}_{s(i-1)}(1)) \\ \boldsymbol{R}_{ri}(1) = a\boldsymbol{R}_{r(i-1)}(1)(1 - \boldsymbol{R}_{r(i-1)}(1)) \\ \boldsymbol{L}_{si}(1) = a\boldsymbol{L}_{s(i-1)}(1)(1 - \boldsymbol{L}_{s(i-1)}(1)) \\ \boldsymbol{L}_{mi}(1) = a\boldsymbol{L}_{m(i-1)}(1)(1 - \boldsymbol{L}_{m(i-1)}(1)) \end{cases} \quad 1 \le i \le M \tag{46}$$

where $\boldsymbol{R}_{s0}(1)$, $\boldsymbol{R}_{r0}(1)$, $\boldsymbol{L}_{s0}(1)$ and $\boldsymbol{L}_{m0}(1)$: initial values to produce the first particle positions at the first iteration. They are random numbers in the interval of (0, 1) and $\boldsymbol{R}_{s0}(1)$, $\boldsymbol{R}_{r0}(1)$, $\boldsymbol{L}_{s0}(1)$ and $\boldsymbol{L}_{m0}(1) \notin \{0.0, 0.25, 0.5, 0.75, 1.0\}$.

$$\begin{cases} \boldsymbol{v}_{Rsi}(1) = a\boldsymbol{v}_{Rs(i-1)}(1)(1 - \boldsymbol{v}_{Rs(i-1)}(1)) \\ \boldsymbol{v}_{Rri}(1) = a\boldsymbol{v}_{Rr(i-1)}(1)(1 - \boldsymbol{v}_{Rr(i-1)}(1)) \\ \boldsymbol{v}_{Lsi}(1) = a\boldsymbol{v}_{Ls(i-1)}(1)(1 - \boldsymbol{v}_{Ls(i-1)}(1)) \\ \boldsymbol{v}_{Lmi}(1) = a\boldsymbol{v}_{Lm(i-1)}(1)(1 - \boldsymbol{v}_{Lm(i-1)}(1)) \end{cases} \quad 1 \le i \le M \tag{47}$$

where $\boldsymbol{v}_{Rs0}(1)$, $\boldsymbol{v}_{Rr0}(1)$, $\boldsymbol{v}_{Ls0}(1)$ and $\boldsymbol{v}_{Lm0}(1)$ are initial values to produce the first particle velocities at the first iteration. They are random numbers in the interval of (0, 1) with $\boldsymbol{v}_{Rs0}(1)$, $\boldsymbol{v}_{Rr0}(1)$, $\boldsymbol{v}_{Ls0}(1)$ and $\boldsymbol{v}_{Lm0}(1) \notin \{0.0, 0.25, 0.5, 0.75, 1.0\}$.

Similarly, the ith particle positions and velocities are also limited as in (36) and (37).

Additionally, the inertia weight, w in the standard PSO algorithm is one of the main parameters which affects the convergence capability and search performance of the algorithm and these aspects are improved when using the chaos PSO algorithm. A chaotic sequence is generated using the logistic map for the inertia weight, w in the velocity update equation of the chaos PSO algorithm. This creates the best balance for the inertia weight during the evolution process of the chaos PSO algorithm between the local and global search processes during a run which results in the best convergence capability and search performance.

The chaotic inertia weight is

$$w_k = aw_{(k-1)}(1 - w_{(k-1)}), \quad 1 \le k \le n \tag{48}$$

where w_k is the kth chaotic inertia weight, $w_k \in (0, 1)$ has the following initial conditions: w_0 is a random number in the interval of (0, 1) and $w_0 \notin \{0.0, 0.25, 0.5, 0.75, 1.0\}$.

Furthermore, the two independent random sequences, r_1 and r_2 are also key factors which affect the algorithm performance. Thus, the logistic map is also used in this application to improve the diversity in the solution space through the two independent chaotic random sequences, r_1 and r_2 in the velocity update equation.

The two independent chaotic random sequences are

$$r_k^1 = ar_{(k-1)}^1(1 - r_{(k-1)}^1), \quad 1 \le k \le n \tag{49}$$

$$r_k^2 = ar_{(k-1)}^2(1 - r_{(k-1)}^2), \quad 1 \le k \le n \tag{50}$$

where r_k^1 and r_k^2 are the two kth independent chaotic random sequences, r_k^1 and $r_k^2 \in (0, 1)$ have the following initial conditions, r_0^1 and r_0^2 are random numbers in the interval of (0, 1) and r_0^1 and $r_0^2 \notin \{0.0, 0.25, 0.5, 0.75, 1.0\}$.

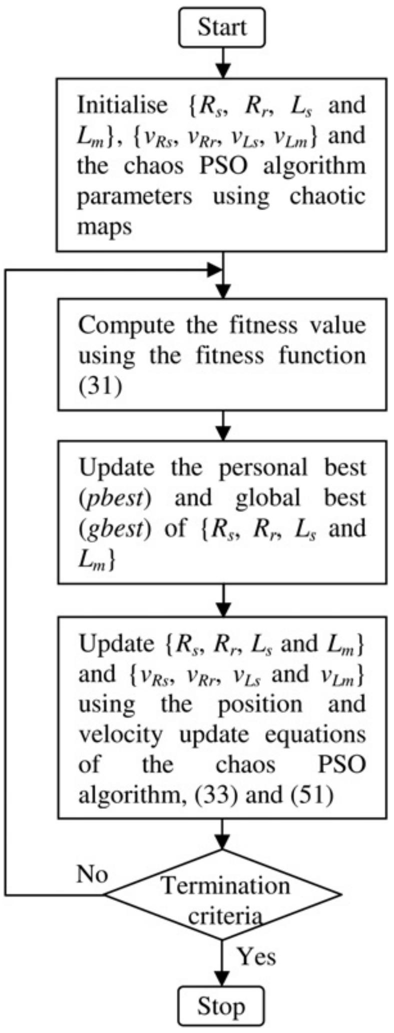
Start
Initialise $\{R_s, R_r, L_s$ and $L_m\}$, $\{v_{Rs}, v_{Rr}, v_{Ls}, v_{Lm}\}$ and the chaos PSO algorithm parameters using chaotic maps
Compute the fitness value using the fitness function (31)
Update the personal best (pbest) and global best (gbest) of $\{R_s, R_r, L_s$ and $L_m\}$
Update $\{R_s, R_r, L_s$ and $L_m\}$ and $\{v_{Rs}, v_{Rr}, v_{Ls}$ and $v_{Lm}\}$ using the position and velocity update equations of the chaos PSO algorithm, (33) and (51)
Termination criteria
No
Yes
Stop

Figure 2 *Flow chart of the chaos PSO algorithm*

Then, the velocity update equation of the standard PSO algorithm is rewritten as follows

$$\boldsymbol{v}_i(k+1) = w_k\boldsymbol{v}_i(k) + c_1 r_k^1(pbest_i(k) - \boldsymbol{x}_i(k) + c_2 r_k^2(gbest(k) - \boldsymbol{x}_i(k)) \quad 1 \le i \le M \quad \text{and} \quad 1 \le k \le n \tag{51}$$

where w_k, r_k^1, r_k^2 are the logistic maps.

In this case, the cognitive and social parameters, c_1 and c_2 are set to 2. The best position of the ith particle $\{pbest_{Rsi}(k)$, $pbest_{Rri}(k)$, $pbest_{Lsi}(k)$, $pbest_{Lmi}(k)\}$ and the best position over the swarm $\{gbest_{Rs}(k)$, $gbest_{Rr}(k)$, $gbest_{Ls}(k)$, $gbest_{Lm}(k)\}$ is obtained at each kth iteration using the fitness function (31) and the update mechanism for the personal and global bests. The evolution process of the chaos PSO algorithm is implemented according to the position and velocity update equations, (33) and (51), respectively. Eventually, the chaos PSO algorithm stops at the nth maximum iteration number and the IM parameters are estimated as in (40). The flow chart of the chaos PSO algorithm is described in Fig. 2.

4 Experimental results

The main elements of the practical experiments are the IM system, data acquisition PCI-6251 card, block connector BNC-2110 together with LabVIEW software for sampling and recording the three-phase stator currents, voltages and the speed of the IM system which is connected directly to a three-phase variac. As this parameter estimation method cannot separate the stator and rotor leakage inductances, they are assumed to be the same because a wound rotor machine is used [1].

The sample rate is 1 kHz and data are recorded over 0.1 s (100 samples). The measurement data processing and parameter estimation are performed in MATLAB.

This paper has compared the parameters estimated using the chaos PSO and dynamic PSO algorithms with the parameters achieved using the DC, no-load and locked-rotor tests. The reason for this approach is that the machine we used for testing is over 50-years old and we do not have the manufacturer's supplied parameter information. The nameplate does not have the stator and rotor resistances, the stator and rotor inductances and the magnetising inductance.

As a result, Table 1 presents the set of the parameters which are applied in the standard PSO, dynamic PSO and chaos PSO algorithms for parameter estimation. In all three algorithms, the particle number of a generation is 80 and the maximum iteration number is set to 200. Each algorithm is run 100 times. The parameter estimation using the GA is implemented by the GA MATLAB toolbox with the same conditions as the standard PSO, dynamic PSO and chaos PSO algorithms. The flow chart of the GA is described in Fig. 3.

Figs. 4–8 are the best fitness of the GA, standard PSO, dynamic PSO and chaos PSO algorithms against the iteration step number and show the convergence capability of each algorithm. Fig. 5 is the enlargement of Fig. 4.

It can be observed that there is a basic difference between the standard PSO and dynamic PSO algorithms from Table 1. The cognitive and social parameters are time-varying variables in the velocity update equation of the dynamic PSO algorithm. This results in a significant improvement in the convergence value of the dynamic PSO algorithm as shown in Figs. 6 and 7. Table 2 shows that the convergence value of the standard PSO algorithm is 0.193 45, whereas that of the dynamic PSO algorithm is 2.151×10^{-6}.

Similarly, several differences also exist between the standard PSO and chaos PSO algorithms in Table 1 such as the initialisation of the particles' positions and velocities using the chaotic map, the chaotic inertia weight and the two chaotic independent random sequences in the velocity update equation of the chaos PSO algorithm. These enhance the solution quality of the algorithm. The convergence value of the chaos PSO algorithm is better than that of the standard PSO algorithm as shown in

Table 1 Parameters in the standard PSO, dynamic PSO and chaos PSO algorithms

Algorithm	Standard PSO	Dynamic PSO	Chaos PSO
initial particles' positions	random numbers $\in (0, 1)$	random numbers $\in (0, 1)$	chaotic maps, using (46)
initial particles' velocities	random numbers $\in (0, 1)$	random numbers $\in (0, 1)$	chaotic maps, using (47)
inertia weight, w	w = constant = 0.9	w = constant = 0.9	a chaotic map, using (48)
acceleration coefficients, c_1 and c_2	$c_1 = c_2$ = constant = 2	time-varying variables, using (42) and (43)	$c_1 = c_2$ = constant = 2
independent random sequences, r_1 and r_2	random numbers $\in (0, 1)$	random numbers $\in (0, 1)$	chaotic maps, using (49) and (50)

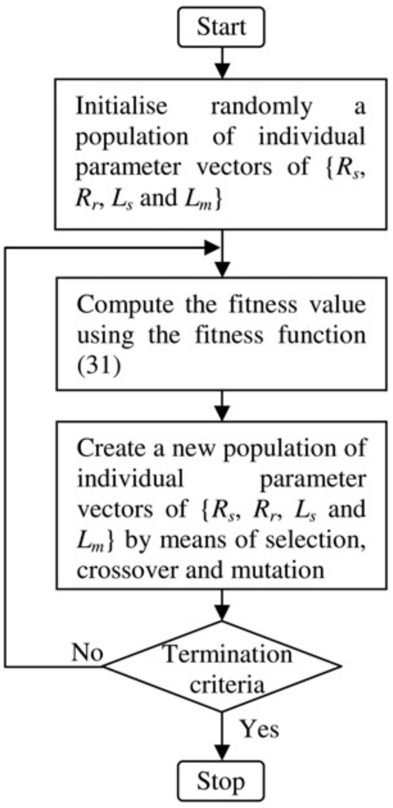
Start
Initialise randomly a population of individual parameter vectors of {R_s, R_r, L_s and L_m}
Compute the fitness value using the fitness function (31)
Create a new population of individual parameter vectors of {R_s, R_r, L_s and L_m} by means of selection, crossover and mutation
Termination criteria
No
Yes
Stop

Figure 3 *Flow chart of the GA*

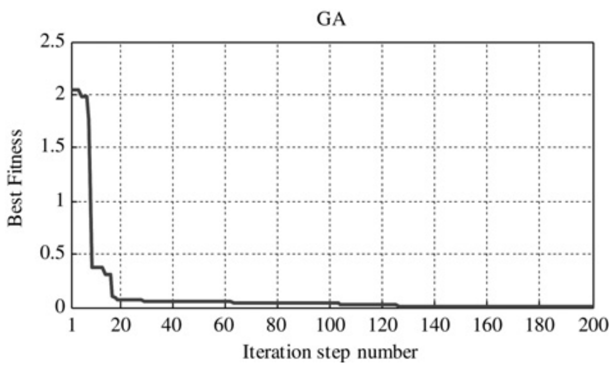

Figure 4 *Best fitness against the iteration step number of the GA*

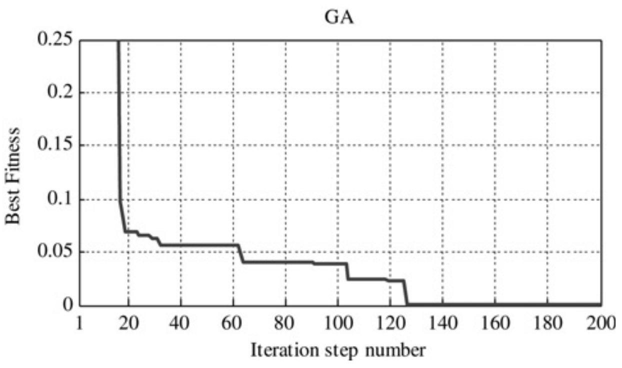

Figure 5 *Enlargement of Fig. 4*

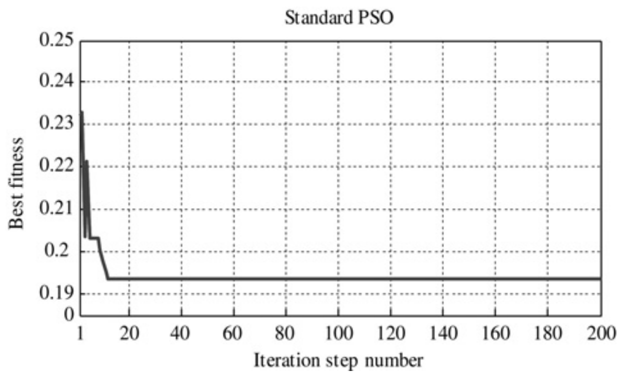

Figure 6 *Best fitness against the iteration step number of the standard PSO algorithm*

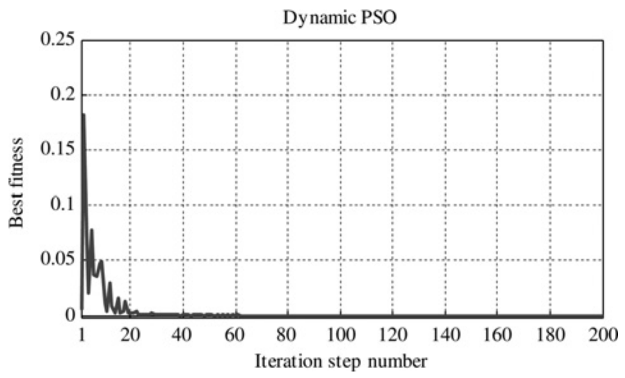

Figure 7 *Best fitness against the iteration step number of the dynamic PSO algorithm*

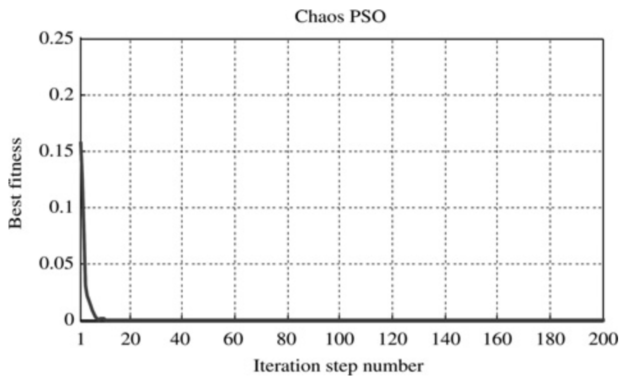

Figure 8 *Best fitness against the iteration step number of the chaos PSO algorithm*

Figs. 6 and 8. Table 2 shows that the convergence value of the standard PSO algorithm is 0.193 45, whereas that of the chaos PSO algorithm is 1.877×10^{-6}.

All these features in both the dynamic PSO and chaos PSO algorithms improve the performance as well as avoiding premature convergence in the standard PSO algorithm as illustrated in Figs. 6 to 8. The dynamic PSO and chaos PSO algorithms are therefore better than the standard PSO algorithm.

Table 2 Convergence value of algorithms

Algorithm	GA	Standard PSO	Dynamic PSO	Chaos PSO
convergence value	5.714×10^{-5}	0.19345	2.151×10^{-6}	1.877×10^{-6}

Additionally, when the standard PSO algorithm is compared with the GA, the standard PSO algorithm converges to the best fitness value faster than the GA in Figs. 5 and 6; however, this does not mean that the standard PSO algorithm is better than the GA. The standard PSO algorithm became stuck in a local optimum during the search process and resulted in premature convergence. Table 3 shows that the standard PSO algorithm converges at the 11th iteration step, whereas the GA converges at the 127th iteration step.

When the GA is compared with the dynamic PSO and chaos PSO algorithms, it is observed that the performance of the dynamic PSO and chaos PSO algorithms are better than the GA in terms of both the convergence speed and value in Figs. 5, 7 and 8. Table 2 shows that the convergence value of the GA is 5.714×10^{-5}, whereas that of the dynamic PSO and chaos PSO algorithms are 2.151×10^{-6} and 1.877×10^{-6}, respectively. Furthermore, Table 3 shows that the dynamic PSO and chaos PSO algorithms converge at the 63rd and 12th iteration steps, respectively, whereas the GA converges at the 127th iteration step.

Table 3 Convergence speed of algorithms

Algorithm	GA	Standard PSO	Dynamic PSO	Chaos PSO
iteration step number	127	11	63	12

Table 4 shows the experimental results of the estimated parameters when using the GA, standard PSO, dynamic PSO and chaos PSO algorithms, whereas Table 5 shows the error percentages of the estimated parameters using the four algorithms.

Fig. 9 (parameter values) and Fig. 10 (percentage errors) show a comparison between the IM parameters obtained using traditional tests such as the dc, no-load and locked-rotor tests and estimated parameters using the GA, standard PSO, dynamic PSO and chaos PSO algorithms. The errors produced by the two new algorithms are always less than 5% and less than the errors achieved when using the GA and standard PSO algorithm. This shows that both the dynamic PSO and chaos PSO algorithm are better than the GA and standard PSO algorithm for parameter estimation of an IM.

Table 4 Value of estimated parameters

Parameter	Traditional tests	GA	Standard PSO	Dynamic PSO	Chaos PSO
R_s, Ω	0.550	0.502	0.702	0.5372	0.5480
R_r, Ω	0.720	0.652	0.537	0.7272	0.7020
L_s, H	0.068	0.073	0.074	0.0679	0.0686
L_r, H	0.068	0.073	0.074	0.0679	0.0686
L_m, H	0.063	0.049	0.061	0.0631	0.0633

Table 5 Percentage errors of the estimated parameters

Error percentage	GA	Standard PSO	Dynamic PSO	Chaos PSO
error percentage of R_s	8.690	27.670	2.3200	0.380
error percentage of R_r	9.450	25.450	1.0040	2.480
error percentage of L_s	8.220	8.290	0.0104	0.860
error percentage of L_r	8.220	8.290	0.0104	0.860
error percentage of L_m	22.710	2.910	0.1163	0.470

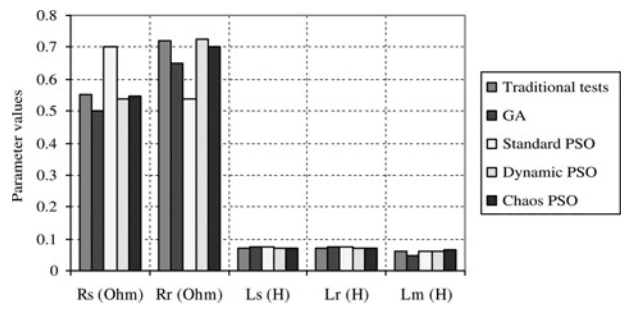

Figure 9 *Comparison between parameters of the traditional tests and estimated parameters using the GA, standard PSO, dynamic PSO and chaos PSO algorithms*

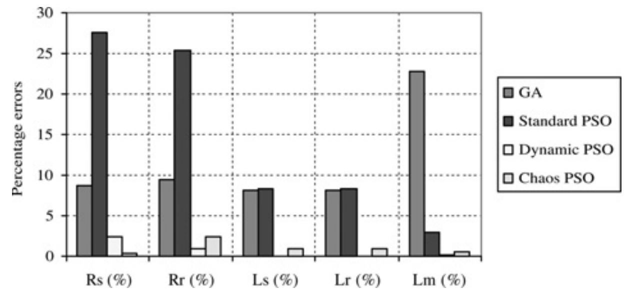

Figure 10 *Percentage errors of estimated parameters using the GA, standard PSO, dynamic PSO and chaos PSO algorithms*

5 Conclusions

In this paper, two novel applications of the dynamic PSO and chaos PSO algorithms have been proposed for parameter estimation of an IM.

The dynamic PSO algorithm is one of the standard PSO algorithm variants, which modifies the cognitive and social parameters in the velocity update equation of the standard PSO algorithm as linear time-varying parameters. Large cognitive and small social parameters are used in the early part for enhancing the global search capability and then small cognitive and large social parameters are utilised at the end stage to improve the convergence of the algorithm.

The combination of the standard PSO algorithm and the chaotic map is known as the chaos PSO algorithm. The randomness-based parameters of the chaos PSO algorithm are initialised using the logistic map for the initial positions and velocities of the particles, the inertia weight and the two independent random sequences in the velocity update equation. The inertia weight in the chaos PSO algorithm was created for the best balance during the evolution process to produce the best convergence capability and search performance. Furthermore, the algorithm has also been improved because of the diversity in the standard PSO algorithm solution space using two independent chaotic random sequences.

The experimental results of the estimated parameters obtained are compared with the IM parameters achieved using traditional tests such as the dc, no-load and locked-rotor tests. Additionally, the results of the estimated parameters using the GA, standard PSO, dynamic PSO and chaos PSO algorithms have been also compared. The results confirm the benefits of the latter two algorithms. The errors produced by the new algorithms are always less than 5% and less than the errors obtained when using the GA and standard PSO algorithm for parameter estimation of an IM.

As a consequence, it can be realised that the PSO algorithms, in common with other types of stochastic optimisation algorithms, cannot be used to solve real-time problems, because they always require time to search for the final optimum solution. They cannot estimate, online, the IM parameters. Obviously, this is the domain of techniques such as the Kalman filter, Luenberger observer, recursive least-squares algorithms and the numerous other methods that have been proposed. Therefore the use of the PSO algorithms for real-time parameter estimation of the IM is impossible.

Additionally, it is obvious that the objective of this paper is to estimate, off-line, the IM parameters. The parameter estimation problem in this paper has been considered under steady-state conditions and the steady-state model has been used in the formulation of the output error expression. The IM parameters have been assumed to be constant in this application. The technique has not been suitable for transient conditions, such as the machine being driven deep into saturation, as the problem is formulated under steady-state conditions.

6 References

[1] AYASUN S., NWANKPA C.O.: 'Induction motor tests using MATLAB/Simulink and their integration into undergraduate electric machinery courses', *IEEE Trans. Educ.*, 2005, **48**, pp. 37–46

[2] HUANG K.S., WU Q.H.: 'Effective identification of induction motor parameters based on fewer measurements', *IEEE Trans. Energy Convers.*, 2002, **17**, pp. 55–60

[3] BISHOP R.R., RICHARDS G.G.: 'Identifying induction machine parameters using a genetic optimization algorithm'. Proc. IEEE, Southeastcon '90, 1990, pp. 476–479

[4] SAG T., CUNKAS M.: 'Multiobjective genetic estimation to induction motor parameters'. Proc. Int. Aegean Conf. Elect. Mach. and Power Electron., ACEMP '07 & Electromotion '07, Bodrum, Turkey, 10–12 September 2007, pp. 628–631

[5] URSEM R.K., VADSTRUP P.: 'Parameter identification of induction motors using stochastic optimization algorithms', *J. Appl. Soft Comput.*, 2004, **4**, (1), pp. 49–64

[6] KARIMI A., CHOUDHRY M.A., FELIACHI A.: 'PSO-based evolutionary optimization for parameter identification of an induction motor'. 39th North American Power Symp., NAPS 2007, 2007, pp. 659–664

[7] GUANGYI C., WEI G., KAISHENG H.: 'On line parameter identification of an induction motor using improved particle swarm optimization'. Proc. 26th Chinese Control Conf., CCC 2007, Zhangjiajie, Hunan, China, July 2007, pp. 745–749

[8] PICARDI C., ROGANO N.: 'Parameter identification of induction motor based on particle swarm optimization'. Int. Symp. Power Electronic, Electrical Drives, Automation, and Motion, SPEEDAM 2006, 2006, pp. 32–37

[9] RATNAWEERA A., HALGAMUGE S.K., WATSON H.C.: 'Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients', *IEEE Trans. Evol. Comput.*, 2004, **8**, pp. 240–255

[10] ALATAS B., AKIN E., OZER A.B.: 'Chaos embedded particle swarm optimization algorithms', *J. Chaos, Solitons Fractals*, 2009, **40**, (4), pp. 1715–1734

[11] ARUMUGAM M.S., CHANDRAMOHAN A., RAO M.V.C.: 'Competitive approaches to PSO algorithms via new acceleration co-efficient variant with mutation operators'. Proc. Sixth Int. Conf. Computational Intelligent and Multimedia Application, ICCIMA '05, 16–18 August 2005, pp. 225–230

[12] LIU B., WANG L., JIN Y.H., TANG F., HUANG D.X.: 'Improved particle swarm optimization combined with chaos', *J. Chaos, Solitons Fractals*, 2005, **25**, (5), pp. 1261–1271

[13] MENG H.J., ZHENG P., WU R.Y., HAO X.J., XIE Z.: 'A hybrid particle swarm algorithm with embedded chaotic search'. Proc. 2004 IEEE Conf. Cybernate and Intelligent System, Singapore, 1–3 December 2004, pp. 367–371

[14] FENG Y., TENG G.F., WANG A.X., YAO Y.M.: 'Chaotic inertia weight in particle swarm optimization'. Second Int. Conf. Innovative Computing, Information and Control, ICICIC '07, 5–7 September 2007, pp. 475–478

[15] FENG Y., YAO Y.M., WANG A.X.: 'Comparing with chaotic inertia weights in particle swarm optimization'. Proc. Sixth Int. Conf. Machine Learning and Cybernetics, Hong Kong, 19–22 August 2007, pp. 329–333

[16] BOSE B.K.: 'AC machines for drives' in 'Modern Power Electronics and AC Drives' (Prentice Hall PTR, 2002), pp. 29–97

[17] PROCO A.B., KEYHANI A.: 'Induction motor parameter identification from operating data electric drive applications'. Proc. 18th Digital Avionics System Conf., 1999, 24–29 October 1999, pp. 1–6

[18] KENNEDY J., EBERHART R.: 'Particle swarm optimization'. Proc. IEEE Int. Conf. Neural Networks, 1995, pp. 1942–1948

[19] BERGH F.V.D.: 'An analysis of particle swarm optimizers'. PhD thesis, Pretoria University, South Africa, 2001

[20] SHI Y., EBERHART R.: 'A modified particle swarm optimizer'. Proc. IEEE Int. Conf. Evolution Computation, Piscataway, New Jersey, 4–9 May 1998, pp. 69–73