Moto User Documentation

Version 0.1 Beta

Contents

1 Introduction

1.1 About Moto

Moto is a parameter estimation tool that can be used to determine the equivalent circuit parameters of induction machines. The tool is intended for use in dynamic time-domain simulations such as stability and motor starting studies.

Moto is unique in that it provides a selection of the most advanced parameter estimation algorithms, which are more accurate and robust than those found in commercial power systems analysis software.

1.2 About Sigma Power Engineering

Sigma Power Engineering Pty Ltd is an electrical engineering consultancy and software developer based in Perth, Australia.

We are a small team of power systems engineers with decades of experience in the utility, hydrocarbons and mining sectors of Australia, Asia and Europe. Our primary focus is to offer the full range of power system studies services, while also supporting our clients with our software tools, project management and design engineering capabilities.

For more details, visit our website www.sigmapower.com.au.

1.3 Release Notes

Version 0.1 Beta is a preliminary version of **Moto** that is restricted to estimating parameters for the double cage model with core losses. In future versions, single cage models and motor models without core losses will be implemented.

For any comments, suggestions or bug reports, please get in touch with us at contact@sigmapower.com.au.

1.4 BSD License

1.5 Disclaimer

THIS SOFTWARE IS PROVIDED BY SIGMA POWER ENGINEERING PTY LTD "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL SIGMA POWER ENGINEERING PTY LTD BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT

2 Technical Background

2.1 Induction Motor Equivalent Circuits

An induction machine can be viewed as a generalised transformer with an air gap and a rotating short-circuited secondary winding. The equivalent circuit of an induction motor is therefore similar to that of a transformer. The key difference is in the rotor equivalent circuit, where the voltages and currents are proportional to the slip frequency. This is commonly represented in the equivalent circuit by a variable (slip-dependent) rotor resistance (i.e. $\frac{R_r}{s}$).

For balanced steady-state analysis, it is acceptable to use a per-phase equivalent circuit. Analysis is also made simpler by working with per-unit values, where the scaling factors required to calculate polyphase quantities (such as polyphase power) are not required. The general induction motor per-phase equivalent circuit with all parameters referred to the stator is shown in Figure 1.

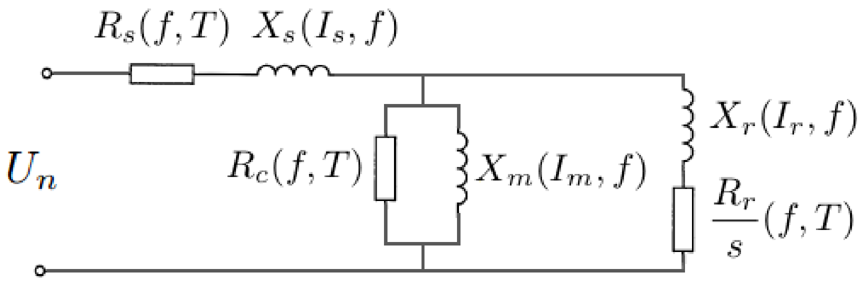


Figure 1: General induction motor equivalent circuit

The parameters of this equivalent circuit are as follows:

- R_s is the stator resistance (pu)
- X_s is the stator leakage reactance (pu)
- X_m is the magnetising reactance (pu)
- R_c is the core loss component (pu)
- X_r is the rotor leakage reactance (pu)
- R_r is the rotor resistance (pu)

The motor equivalent circuit in Figure 1 shows that the parameters vary with frequency / slip, current (i.e. saturation effects) and temperature, for the following reasons:

- AC resistances for the copper windings are temperature dependent [15]
- Inductances will vary due to eddy currents and saturation of teeth and core [13]
- Rotor resistance is frequency (slip) dependent due to eddy currents in deep rotor bars or double cage rotors [1]
- Impedances are affected by the skin effect at different frequencies [1]

For power system studies, it is desirable to use a motor equivalent circuit with constant parameters that are valid over the full range of motor speeds (i.e. from 0 to 1 pu). It is important to note that these parameters will not likely correspond to the real motor parameters, since as noted above, the real parameters are slip, current and temperature dependent. However, the set of constant equivalent circuit parameters will match the motor's performance characteristics over the full speed range (e.g. torque-slip and current-slip curves, power factor, efficiency, etc).

While others have attempted to apply motor models with variable parameters (for example, Haque in [2]), there does not appear to be significant gains to be had by this approach and it only makes the model more complicated and less manageable for standard power systems analysis programs. Therefore, only constant parameter models are considered in Moto. In the following subsections, the two most common constant parameter equivalent circuit models are presented.

2.1.1 Single Cage Model

The single cage model is simply the general equivalent circuit in Figure 1 with constant parameters. The single cage model is normally suitable to represent the performance characteristics of wound-rotor motors. Two single cage equivalent circuits are shown below in Figures 2 and 3 depicting the model with and without core losses respectively. Note that in Figure 3, the core loss component is lumped as a single shunt resistance placed at the input of the equivalent circuit. This is done for practical reasons as described in [12].

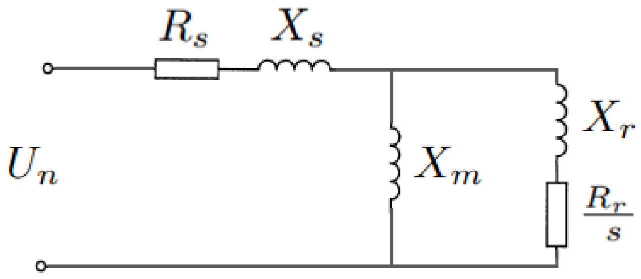


Figure 2: Basic single cage model equivalent circuit (5 parameters)

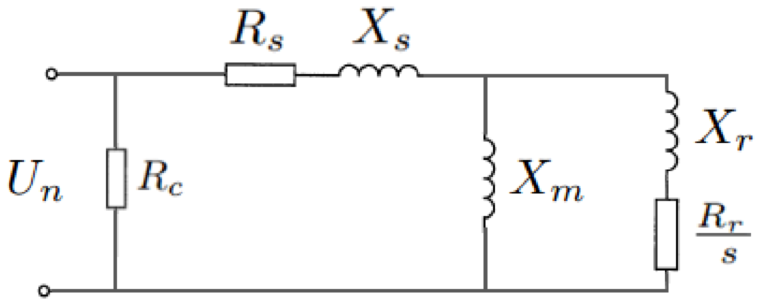


Figure 3: Single cage model with core losses (6 parameters)

2.1.2 Double Cage Model

To account for the effects of double-cage rotors or deep bar rotors (e.g. most squirrel-cage machines), a second rotor branch is added to the equivalent circuit of the single cage model. Figures 4 and 5 show the double cage model equivalen circuit with and without core losses. As in the single cage model, the core losses are represented as a single shunt resistance at the input of the equivalent circuit.

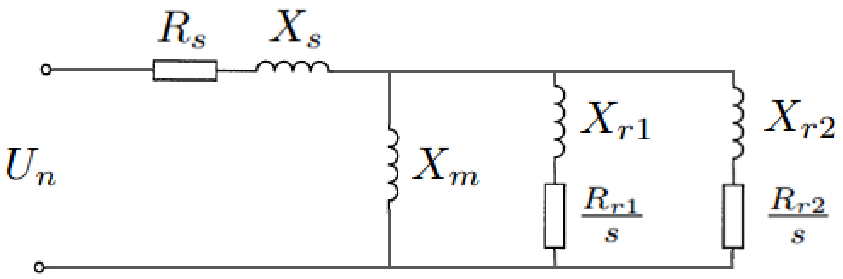


Figure 4: Basic double cage model equivalent circuit (7 parameters)

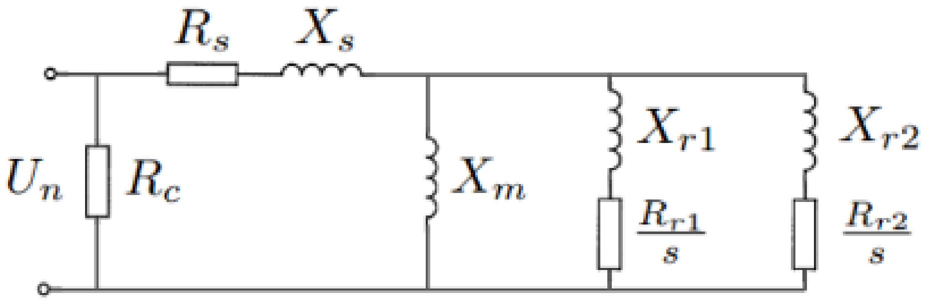


Figure 5: Double cage model with core losses (8 parameters)

In the equivalent circuit, the inner cage leakage reactance X_{r1} is always higher than the outer cage leakage reactance X_{r2}, but the outer cage impedance is typically higher than the inner cage impedance on starting. These conditions can be resolved by including the following two inequality constraints in the model [10]:

- $X_{r1} > X_{r2}$
- $R_{r2} > R_{r1}$

2.2 Calculating Torque and Current from the Equivalent Circuit

The electrical torque developed in an induction machine is proportional to the square of rotor current, i.e.

$$T = \frac{pq}{4\pi f} \frac{R_r}{s} I_r^2 \tag{1}$$

where T is the electrical torque developed (N-m)
p is the number of motor poles
q is the number of stator phases
f is the nominal frequency (Hz)
R_r is the equivalent rotor resistance (Ω)
s is the motor slip (pu)
I_r is the rotor current (A)

By using per-unit values, the constant terms can be eliminated and the equation above reduces to:

$$T = \frac{R_r}{s} I_r^2 \tag{2}$$

where all the quantities in this equation are in per-unit values.

It can be sees that for any given motor equivalent circuit, standard circuit analysis can be used to calculate the rotor current and therefore electrical torque. By way of example, the torque in the double cage model (without core losses) as shown in Figure 4 will be calculated.

Recasting the impedances as admittances:

$$Y_s = \frac{1}{R_s + jX_s} \tag{3}$$

$$Y_m = \frac{1}{jX_m} \tag{4}$$

$$Y_{r1} = \frac{1}{\frac{R_{r1}}{s} + jX_{r1}} \tag{5}$$

$$Y_{r2} = \frac{1}{\frac{R_{r2}}{s} + jX_{r2}} \tag{6}$$

Applying Kirchoff's law, the voltage U_1 at the magnetising branch is:

$$(U_n - U_1)\,Y_s = U_1\,(Y_m + Y_{r1} + Y_{r2}) \tag{7}$$

$$U_n Y_s = U_1\,(Y_s + Y_m + Y_{r1} + Y_{r2}) \tag{8}$$

$$U_1 = \frac{U_n Y_s}{Y_s + Y_m + Y_{r1} + Y_{r2}} \tag{9}$$

The per-unit stator current I_s is therefore:

$$I_s = (U_n - U_1)\,Y_s \tag{10}$$

The per-unit rotor currents in each cage I_{r1} and I_{r2} are:

$$I_{r1} = U_1 Y_{r1} \tag{11}$$

$$I_{r2} = U_1 Y_{r2} \tag{12}$$

Finally, the per-unit electrical torque developed in the motor is:

$$T = \frac{R_{r1}}{s} I_{r1}^2 + \frac{R_{r2}}{s} I_{r2}^2 \tag{13}$$

A similar kind of analysis can be done for other motor equivalent circuit models to calculate the electrical torque and current of the machine.

2.3 Torque-Speed and Current-Speed Curves

Based on the torque and stator current equations developed in the previous section, torque-speed and current-speed curves can be constructed from the equivalent circuit for the full range of motor speeds (i.e. from standtill to synchronous speed).

Examples of motor torque-speed and current-speed curves are shown in Figure 6 and Figure 7 respectively.

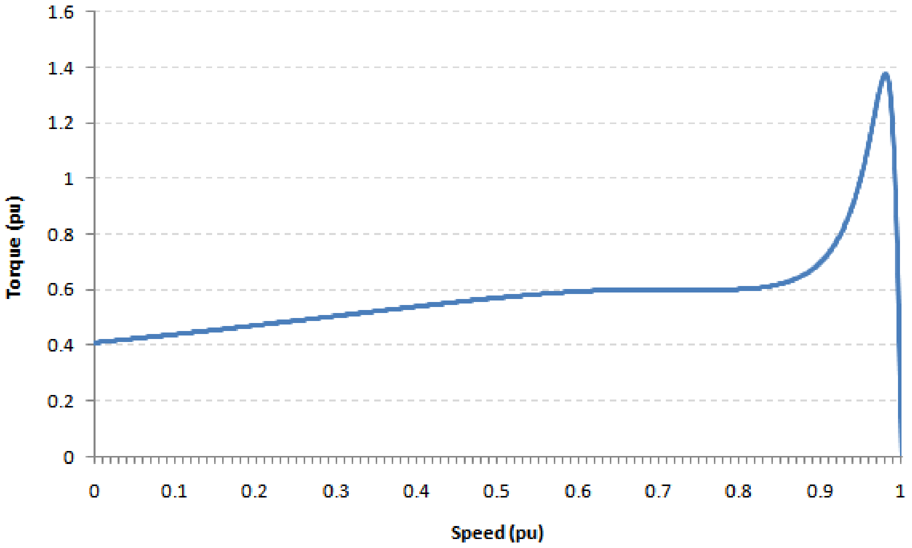


Figure 6: Motor torque-speed curve

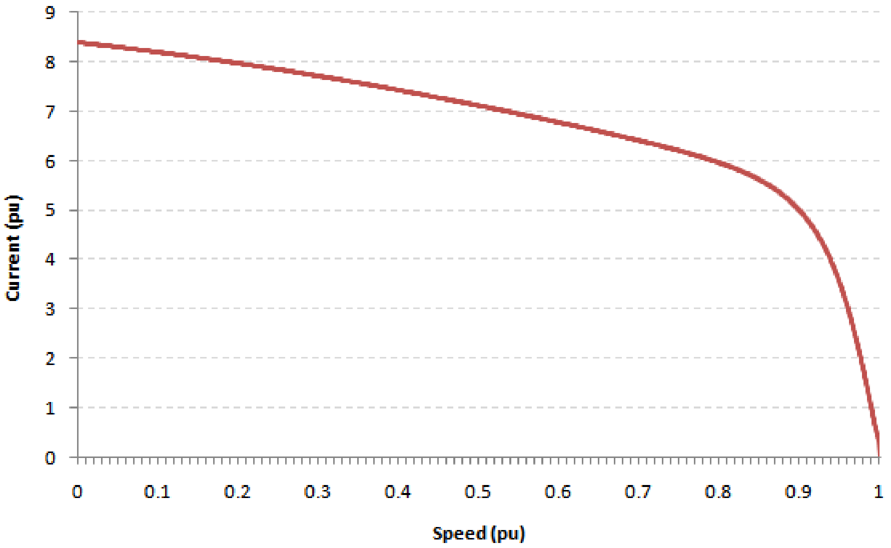


Figure 7: Motor current-speed curve

2.4 Parameter Estimation Problem

The characteristics of an induction motor are normally provided by manufacturers in the form of a standard set of performance parameters, with the following parameters being the most common:

- Nominal voltage, U_n (V)
- Nominal frequency, f (Hz)
- Rated asynchronous speed, n_{fl} (rpm)
- Rated (stator) current, $I_{s,fl}$ (A)
- Rated mechanical power, $P_{m,fl}$ (kW)
- Rated torque, T_n (Nm)
- Full load power factor, $\cos \phi_{fl}$ (pu)
- Full load efficiency, η_{fl} (pu)
- Breakdown torque, T_b/T_n (normalised)
- Locked rotor torque, T_{lr}/T_n (normalised)
- Locked rotor current, $I_{lr}/I_{s,fl})$ (pu)

From previous sections, we know that a set of equivalent circuit parameters can yield specific torque-speed and current-speed curves. So given a set of performance parameters that contain features on the torque-speed and current-speed curves (e.g. breakdown torque, locked-rotor current, etc), is it possible to determine the corresponding equivalent circuit parameters that yield these features? This is the crux of the parameter estimation problem and can be posed as follows - ”How can the motor performance parameters be converted into equivalent circuit parameters?”.

While all of the performance parameters in the above set can be used in an estimation procedure, there are actually only six indpendent magnitudes that can be formed from them: $P_{m,fl}$, Q_{fl}, T_b, T_{lr}, I_{lr} and η_{fl} [10]. These independent magnitudes will thus form the basis of the problem formulation, where the independent magnitudes calculated from the equivalent circuit are matched with the performance parameters supplied by the manufacturer.

The basic double cage model shown in Figure 4 is used to illustrate how these six independent magnitudes can be calculated from the equivalent circuit model. Stator and rotor currents at slip s can be readily calculated from the equivalent circuit as shown in section 2.2.

Quantities for per-unit active power P, reactive power Q and power factor $\cos \phi$ at slip s can be calculated as follows:

$$S(s) = U_n I_s(s)^* \tag{14}$$

$$P(s) = T(s)(1 - s) \tag{15}$$

$$Q(s) = \Im\{S(s)\} \tag{16}$$

$$\cos \phi(s) = \frac{\Re\{S(s)\}}{||S(s)||} \tag{17}$$

Nominal speed n_s and full load slip s_f is calculated as follows:

$$n_s = \frac{120f}{p} \tag{18}$$

$$s_f = 1 - \frac{n_{fl}}{n_s} \tag{19}$$

where p is the number of motor poles
f is the nominal frequency (Hz)
n_{fl} is the asynchronous speed at full load (rpm)

Calculating the slip at maximum torque s_{max} is found by solving the equation:

$$\frac{dT}{ds} = 0 \tag{20}$$

(Under the condition that the second derivative $\frac{d^2}{ds^2} < 0$)

In the double cage model, the solution to this equation is not trivial and it is more convenient to use an estimate, e.g. based on an interval search between $s = 0$ and $s = 0.5$.

2.5 Problem Formulation Ignoring Core Losses

2.5.1 Single Cage Model (Ignoring Core Losses)

In the single cage model, the locked rotor torque T_{lr} and locked rotor current I_{lr} are not used because the single cage model does not have enough degrees of freedom to capture both the starting and breakdown torque characteristics without introducing significant errors [11]. As a result, it is more commonplace to only consider the breakdown torque T_b in the single cage model and simply ignore the torque and current characteristics at locked rotor. For wound-rotor motors, this yields sufficiently accurate results (i.e. in terms of the resulting torque-speed curve). However, a single-cage model is unable to accurately model the torque-speed characteristics of squirrel cage motors, especially those with deep bars, and thus a double cage model should be used for these types of motors.

Without taking into account core losses, the full load motor efficiency η_{fl} also cannot be used (see section 2.6 for more details). Therefore, there are only three independent parameters that can be used in the problem formulation: $P_{m,fl}$, Q_{fl} and T_b.

These independent parameters can be used to formulate the parameter estimation in terms of a non-linear least squares problem, with a set of non-linear equations of the form $\mathbf{F}(\mathbf{x}) = \mathbf{0}$:

$$\begin{align} f_1(\mathbf{x}) &= P_{m,fl} - P(s_f) = 0 \tag{21} \\ f_2(\mathbf{x}) &= \sin\phi - Q(s_f) = 0 \tag{22} \\ f_3(\mathbf{x}) &= T_b - T(s_{max}) = 0 \tag{23} \\ & \tag{24} \end{align}$$

where $\mathbf{F} = (f_1, f_2, f_3)$ and
$\mathbf{x} = (R_s, X_s, X_m, R_r, X_r)$ are the equivalent circuit parameters of the single cage model

2.5.2 Double Cage Model (Ignoring Core Losses)

In the double cage model, the locked rotor torque T_{lr} and locked rotor current I_{lr} are included as independent parameters. As in the single cage model, the full load motor efficiency η_{fl} cannot be used without taking into account core losses. Therefore, there are five independent parameters and the following non-linear least squares problem:

$$f_1(\mathbf{x}) = P_{m,fl} - P(s_f) = 0 \tag{25}$$
$$f_2(\mathbf{x}) = \sin\phi - Q(s_f) = 0 \tag{26}$$
$$f_3(\mathbf{x}) = T_b - T(s_{max}) = 0 \tag{27}$$
$$f_4(\mathbf{x}) = T_{lr} - T(s = 1) = 0 \tag{28}$$
$$f_5(\mathbf{x}) = I_{lr} - I(s = 1) = 0 \tag{29}$$
$$\tag{30}$$

where $\mathbf{F} = (f_1, f_2, f_3, f_4, f_5)$ and
$\mathbf{x} = (R_s, X_s, X_m, R_{r1}, X_{r1}, R_{r2}, X_{r2})$ are the equivalent circuit parameters of the double cage model

2.6 Problem Formulation Considering Core Losses

It was previously noted that without taking into account the core (and mechanical) losses, the motor full load efficiency η_{fl} cannot be used as an independent parameter in the problem formulation. This is because efficiency is calculated based on the ratio of output mechanical power to input electrical power. If the heat losses through the core and rotor frictional losses are not taken into account, then the equivalent circuit is not suitable to accurately estimate motor efficiency [2]. It follows that attempting to use the motor full load efficiency in the estimation of the equivalent circuit without a core loss component would cause errors in the parameter estimates (e.g. the stator resistance would be overestimated).

When core losses are included in the model, then the motor full load efficiency η_{fl} can also be used as an independent parameter. The problem formulations are restated below for the single cage and double cage models with core losses taken into account.

2.7 Single Cage Model (with Core Losses)

The non-linear least squares problem for the single cage model with core losses is as follows:

$$f_1(\mathbf{x}) = P_{m,fl} - P(s_f) = 0 \tag{31}$$
$$f_2(\mathbf{x}) = \sin\phi - Q(s_f) = 0 \tag{32}$$
$$f_3(\mathbf{x}) = T_b - T(s_{max}) = 0 \tag{33}$$
$$f_4(x) = \eta fl - \eta(s_f) = 0 \tag{34}$$

where $\mathbf{F} = (f_1, f_2, f_3, f_4)$ and
$\mathbf{x} = (R_s, X_s, X_m, R_r, X_r, R_c)$ are the equivalent circuit parameters of the single cage model (with core losses)

2.7.1 Double Cage Model (with Core Losses)

The non-linear least squares problem for the double cage model with core losses is as follows:

$$f_1(\mathbf{x}) = P_{m,fl} - P(s_f) = 0 \tag{35}$$
$$f_2(\mathbf{x}) = \sin\phi - Q(s_f) = 0 \tag{36}$$
$$f_3(\mathbf{x}) = T_b - T(s_{max}) = 0 \tag{37}$$
$$f_4(\mathbf{x}) = T_{lr} - T(s = 1) = 0 \tag{38}$$
$$f_5(\mathbf{x}) = I_{lr} - I(s = 1) = 0 \tag{39}$$
$$f_6(x) = \eta fl - \eta(s_f) = 0 \tag{40}$$

where $\mathbf{F} = (f_1, f_2, f_3, f_4, f_5, f_6)$ and
$\mathbf{x} = (R_s, X_s, X_m, R_{r1}, X_{r1}, R_{r2}, X_{r2}, R_c)$ are the equivalent circuit parameters of the double cage model (with core losses)

3 Parameter Estimation Algorithms

The parameter estimation problems formulated in the preceding sections can be solved by a variety of non-linear least squares solver algorithms. As with all non-linear least squares problems, closed form solutions are generally not available and iterative algorithms are used to converge on a solution by minimising error residuals.

Motor parameter estimation algorithms generally fall under three broad classes:

1. **Descent Methods**: are the class of algorithms based on variations of Newton's method for convergence to a solution, e.g. Newton-Raphson, Levenberg-Marquardt, etc
2. **Natural Optimisation Methods**: are the class of algorithms based on processes found in nature where successive randomised trials are filtered for "fitness" at each iteration, e.g. genetic algorithm, particle swarm optimisation, ant colony optimisation, simulated annealing, etc
3. **Hybrid Methods**: are the class of algorithms that use a combination of both descent and natural optimisation methods.

Moto provides the following seven (7) algorithms for solving parameter estimation algorithms:

- Newton-Raphson (Descent)
- Levenberg-Marquardt (Descent)
- Damped Newton-Raphson (Descent)
- Genetic Algorithm (Natural Optimisation)
- GA-NR (Hybrid)
- GA-LM (Hybrid)
- GA-DNR (Hybrid)

3.1 Newton-Raphson Algorithm

Of the class of descent methods used to solve non-linear least squares problems, the Newton-Raphson (NR) algorithm is probably the most straightforward. The NR algorithm is an iterative method where each iteration is calculated as follows:

$$\boldsymbol{x}^{k+1} = \boldsymbol{x}^{k} - h_n \mathbf{J}^{-1} \mathbf{F}(\boldsymbol{x}^{k}) \tag{41}$$

where $\boldsymbol{x}^{k+1}$ is the solution at the $(k+1)$th iteration, $\boldsymbol{x}^{k}$ is the solution at the kth iteration, h_n is the step-size coefficient (more on this later) and $\mathbf{J}$ is the Jacobian matrix evaluated with the parameters at the kth iteration, $\boldsymbol{x}^{k}$.

The Jacobian matrix $\mathbf{J}$ has the general form:

$$\mathbf{J} = \begin{pmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_6} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_6}{\partial x_1} & \cdots & \frac{\partial f_6}{\partial x_6} \end{pmatrix} \tag{42}$$

For systems where it is impractical to compute the exact partial derivatives analytically, a numerical approximation may be used with finite difference equations:

$$\frac{\partial f_i}{\partial x_j} \approx \frac{f_i(\mathbf{x} + \boldsymbol{\delta_j} h) - f_i(\mathbf{x})}{h} \tag{43}$$

where $\boldsymbol{\delta_j}$ is vector of zeros with a single non-zero value of 1 at the j-th element and h is a constant with a very small absolute value (e.g. 1×10^{-6}).

A modified form of the NR algorithm proposed in [12] for the double cage model is shown in Figure 8. This algorithm was selected because of its completeness, numerical accuracy and robustness compared to previously proposed methods (for example, in [4], [13] and [15]). Furthermore, the algorithm can be applied using commonly available manufacturer data, whereas other algorithms require more detailed data that may not be readily available (for example, the full torque-speed curve in [6]). Two other features of the algorithm that aid its robustness are worth highlighting:

3.1.1 Need for Linear Constraints

It can be seen from the problem formulations in the previous chapter that in each case, the number of paramters to be estimated (i.e. unknown variables) exceeds the number of simultaneous equations. In other words, the systems of equations are all underdetermined. Therefore, in order to make the systems exactly determined and solvable with descent algorithms such as the NR algorithm, we must either:

1. Fix two parameters a priori (i.e. parameters are "known")
2. Impose two constraints on the problem formulations, e.g. linear constraints

The use of linear constraints was found to be superior to fixed parameters and therefore, the descent algorithms in **Moto** include two linear restrictions by default.

It was shown in [10] that the stator resistance R_s was the least sensitive parameter in the equivalent circuit, i.e. variations in the value of R_s had the least significant effect on the resulting torque-speed and current-speed curves. Therefore, R_s can be subject to a linear restriction by linking it to the rotor resistance, leading to the first linear constraint:

- $R_s = k_r R_r$ (for the single cage model)
- $R_s = k_r R_{r1}$ (for the double cage model)

Where k_r is a constant linear constraint

Moreover, it is assumed that the rotor reactance is linearly related to the stator reactance, leading to the second linear constraint.

- $X_r = k_x X_s$ (for the single cage model)
- $X_{r2} = k_x X_s$ (for the double cage model)

Where k_x is a constant linear constraint

3.1.2 Parameter Constraints

The inequality constraints of the double cage model ($X_{r1} > X_{r2}$ and $R_{r2} > R_{r1}$) can be implicitly included into the formulation by a simple change of variables [10]:

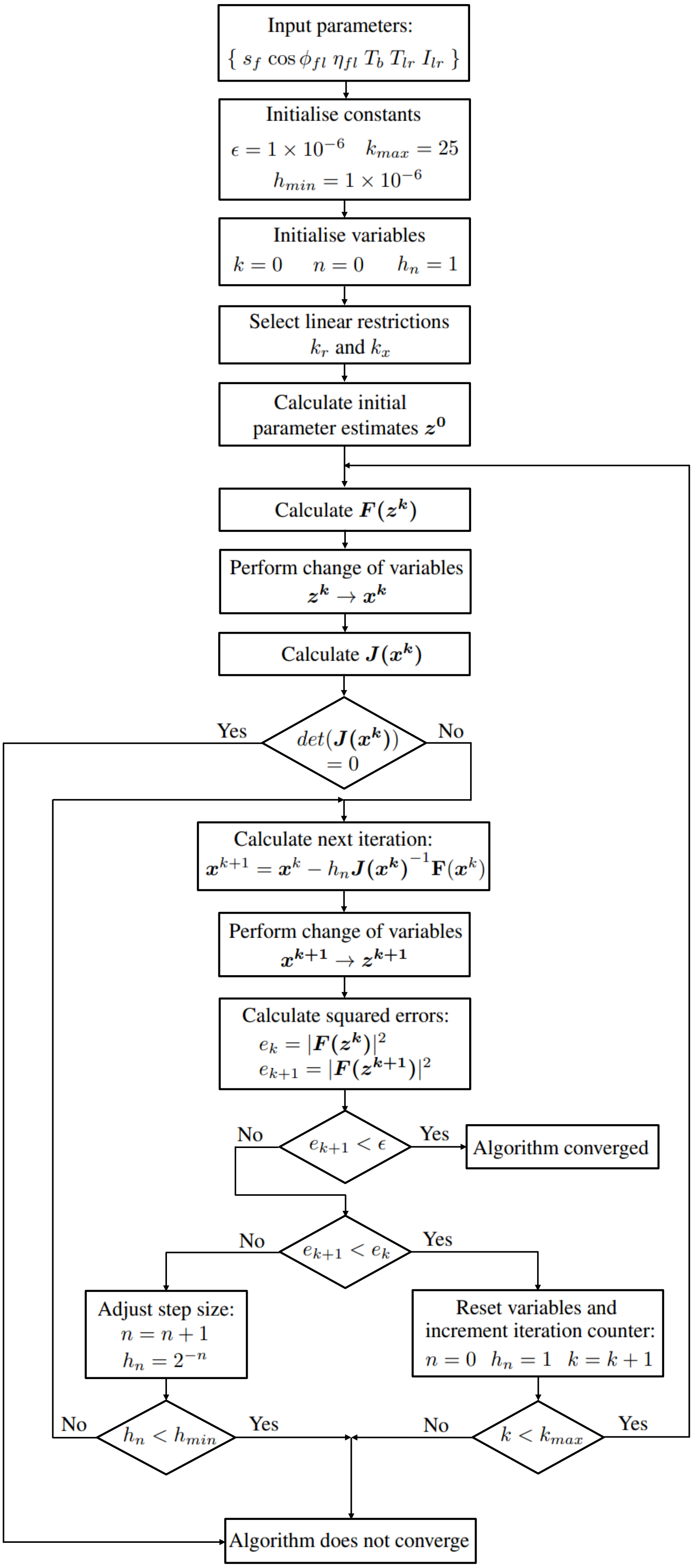

Input parameters:
$\{\ s_f\ \cos\phi_{fl}\ \eta_{fl}\ T_b\ T_{lr}\ I_{lr}\ \}$
Initialise constants
$\epsilon = 1 \times 10^{-6}$ $k_{max} = 25$
$h_{min} = 1 \times 10^{-6}$
Initialise variables
$k = 0$ $n = 0$ $h_n = 1$
Select linear restrictions
k_r and k_x
Calculate initial
parameter estimates $\boldsymbol{z^0}$
Calculate $\boldsymbol{F}(\boldsymbol{z^k})$
Perform change of variables
$\boldsymbol{z^k} \to \boldsymbol{x^k}$
Calculate $\boldsymbol{J}(\boldsymbol{x^k})$
Yes
$det(\boldsymbol{J}(\boldsymbol{x^k}))$
$= 0$
No
Calculate next iteration:
$\boldsymbol{x}^{k+1} = \boldsymbol{x}^k - h_n \boldsymbol{J}(\boldsymbol{x^k})^{-1}\mathbf{F}(\boldsymbol{x}^k)$
Perform change of variables
$\boldsymbol{x^{k+1}} \to \boldsymbol{z^{k+1}}$
Calculate squared errors:
$e_k = |\boldsymbol{F}(\boldsymbol{z^k})|^2$
$e_{k+1} = |\boldsymbol{F}(\boldsymbol{z^{k+1}})|^2$
No
$e_{k+1} < \epsilon$
Yes
Algorithm converged
No
$e_{k+1} < e_k$
Yes
Adjust step size:
$n = n + 1$
$h_n = 2^{-n}$
Reset variables and
increment iteration counter:
$n = 0$ $h_n = 1$ $k = k + 1$
No
$h_n < h_{min}$
Yes
No
$k < k_{max}$
Yes
Algorithm does not converge

Figure 8: Flowchart for conventional NR algorithm

$$\begin{aligned}
x_1 &= R_{r1} \\
x_2 &= R_{r2} - R_{r1} \\
x_3 &= X_m \\
x_4 &= X_s \\
x_5 &= X_{r1} - k_x X_s \\
x_6 &= R_c
\end{aligned}$$

Furthermore, only the absolute values of the parameter estimates are used to ensure that no negative parameters are estimated.

3.1.3 Adaptive Step Size

The step size h_n in equation 41 is a scaling term that determines how far the algorithm should go along the descent direction $\mathbf{J}^{-1}\mathbf{F}(\boldsymbol{x}^k)$. Choosing a step size that is too large risks the algorithm not converging. On the other hand, choosing a step size that is too small can cause the algorithm to converge too slowly. An adaptive step size can avoid both these problems by starting with a high step size and only reducing it if the algorithm does not converge (refer to the flowchart in Figure 8).

3.1.4 Initial Conditions

For the base case NR algorithm, the initial parameter estimates are selected as follows [10]:

$$\begin{aligned}
R_{r1} &= \frac{U_n s_f}{P_{m,fl}} \\
X_m &= \frac{U_n}{Q_{fl}} \\
X_s &= 0.05 X_m \\
R_s &= k_r R_{r1} \\
R_{r2} &= 5 R_{r1} \\
X_{r1} &= 1.2 X_s \\
X_{r2} &= k_x X_s \\
R_c &= 10
\end{aligned}$$

3.2 Levenberg-Marquardt Algorithm

The Levenberg-Marquardt (LM) algorithm, sometimes called the damped least-squares algorithm, is another popular technique for solving least-saures problems [5] [7]. In the LM algorithm, each iteration is calculated as follows:

$$\boldsymbol{x}^{k+1} = \boldsymbol{x}^k - \left[\mathbf{J}^T\mathbf{J} + \lambda\,\mathbf{diag}(\mathbf{J}^T\mathbf{J})\right]^{-1}\mathbf{J}^T\mathbf{F}(\boldsymbol{x}^k) \tag{44}$$

where $\boldsymbol{x}^{k+1}$ is the solution at the $(k+1)$th iteration, $\boldsymbol{x}^k$ is the solution at the kth iteration, λ is the damping parameter (more on this later) and $\mathbf{J}$ is the Jacobian matrix evaluated with the parameters at the kth iteration, $\boldsymbol{x}^k$ (as described previously in Equation 42).

Parameter constraints as implemented in the Newton-Raphson algorithm are also applied in the LM algorithm (refer to Section 3.1.2). The initial conditions are also selected in the same way as the NR algorithm.

The selection of the damping parameter λ affects both the direction and magnitude of an iteration step. If the damping parameter is large, then the algorithm will move at short steps in the steepest descent direction. This is good when the present iteration is far away from the solution. On the other hand, if the damping parameter is small, then the algorithm approaches a Gauss-Newton type method, which exhibits good convergence in the neighbourhood of the solution.

Therefore, the damping parameter should be updated at each iteration depending on whether the algorithm is far or close to the solution. The "gain ratio" method suggested for adjusting the damping factor by Marquardt in [7]was not found to be effective in induction motor problems. Therefore, **Moto** uses a damping factor adjustment method based only on the error term (i.e. the numerator of the gain ratio).

The damping parameter is updated as follows:

$$\lambda = \begin{cases} \lambda \times \beta, & \text{if } \mathbf{F}(\boldsymbol{x}^k) - \mathbf{F}(\boldsymbol{x}^{k+1}) < 0 \\ \frac{\lambda}{\gamma}, & \text{if } \mathbf{F}(\boldsymbol{x}^k) - \mathbf{F}(\boldsymbol{x}^{k+1}) > 0 \end{cases} \tag{45}$$

Where β and γ are algorithm control parameters. In **Moto**, the algorithm control parameters are $\beta = 3$ and $\gamma = 3$.

3.3 Damped Newton-Raphson Algorithm

The damped Newton-Raphson algorithm is a variation of the conventional NR algorithm where a damping factor is applied to help get around problems with near-singular and/or ill-conditioned Jacobian matrices. In the damped NR algorithm, each iteration is calculated as follows:

$$\boldsymbol{x}^{k+1} = \boldsymbol{x}^k - h_n(\mathbf{J}^{-1} + \lambda I)\mathbf{F}(\boldsymbol{x}^k) \tag{46}$$

Where the damping parameter λ is adjusted at each iteration based on the error term as follows:

$$\lambda = \begin{cases} \lambda \times \beta, & \text{if } \mathbf{F}(\boldsymbol{x}^k) - \mathbf{F}(\boldsymbol{x}^{k+1}) < 0 \\ \frac{\lambda}{\gamma}, & \text{if } \mathbf{F}(\boldsymbol{x}^k) - \mathbf{F}(\boldsymbol{x}^{k+1}) > 0 \end{cases} \tag{47}$$

Where β and γ are algorithm control parameters. In **Moto**, the algorithm control parameters are $\beta = 3$ and $\gamma = 3$.

All other aspects of the damped NR algorithm are the same as per the conventional NR algorithm described in Section 3.1 (e.g. parameter constraints, adaptive step sizes, etc)

3.4 Genetic Algorithm

The genetic algorithm (GA) is part of the class of evolutionary algorithms modelled on natural selection and evolutionary processes to optimise non-linear cost functions. It was developed in the 1960s and 1970s, but only gained widespread popularity in the late 1980s when advances in computational processing power made the algorithm more practical to apply on desktop computers [3].

The goal of the genetic algorithm is to minimise a non-linear cost function. For non-linear least squares problems, this can be interpreted as minimising the squared error residuals. The general methodology can be summarised under four broad headings - 1) Initialisation, 2) Fitness and Selection, 3) Breeding and Inheritance and 4) Termination.

1. **Initialisation**: an initial population of candidate solutions to minimise the cost function is first generated, typically via random sampling. The size of the population is an algorithm setting and largely depends on the nature of the problem.
2. **Fitness and Selection**: the population is ranked according to the fitness of its members. The fitness of each member is normally calculated from the cost function, where lower values signal higher fitness. The fittest set of members are selected to evolve / breed the next generation.

3. **Breeding and Inheritance**: the members chosen in the selection stage are designated as "parents" and evolved to create the next generation of candidate solutions ("children"). There are three common methods for generating children - elite children, crossover and mutation.

   Elite children are simply clones of the fittest-ranked parents. In the crossover operation, pairs of parents are bred together by randomly selecting traits from each parent that are passed on to the children. In the mutation operation, the traits of a parent are randomly altered (mutated) and then passed on to the children. Crossover and mutation are obviously inspired by nature and in the genetic algorithm, they can either be implemented simultaneously or as separate processes.

4. **Termination**: once the next generation of candidate solutions has been produced, the population is then ranked again according to their fitness. Since the least fit members of the previous generation have been discarded, the average fitness of the new generation should be higher. Successive generations are bred until either a converging solution is found (i.e. a squared error $< 1 \times 10^{-5}$) or the maximum number of generations is reached.

In the context of motor parameter estimation, the genetic algorithm is used to minimise the squared error of the problem formulation vector **F**. In GA terminology, the squared error is the fitness function and is calculated as follows:

$$fitness = \mathbf{F}\mathbf{F}' \tag{48}$$

where $\mathbf{F} = (f_1, f_2, f_3, f_4, f_5, f_6)$

Genetic algorithms can be binary coded where the solution paramaters are quantized into binary strings (for example, in [8], [16] and [9]). However, the equivalent circuit parameters in a motor are continuous parameters and not naturally quantized. Thus, binary coding necessarily imposes limits on the precision of the parameters (i.e. due to the chosen length of the binary string). For this reason, a continuous parameter algorithm is used instead.

An initial population of n_{pop} parameter estimates are randomly sampled from a uniform distribution with upper and lower limits as shown in Table 1.

Parameter	Range of Initial Estimate (pu)	
	Lower Bound	Upper Bound
R_s	0	0.15
X_s	0	0.15
X_m	0	5
R_{r1}	0	0.15
X_{r1}	0	0.30
R_{r2}	0	0.15
X_{r2}	0	0.15
R_c	0	100

Table 1: Range of initial parameter estimates

The fitness of each member in the population is then calculated and ranked. The lowest fitness members are discarded and the rest are retained to form the mating pool for the next generation (there are n_{pool} members in the mating pool).

The fittest n_e members in the mating pool are retained for the next generation as **elite children**.

Of the remaining $n_{pop} - n_e$ children to be created for the next generation, $c_f\%$ will be produced by crossover and the rest $(1 - c_f\%)$ by mutation. The proportion c_f is called the **crossover fraction**.

1. **Crossover**: in the crossover process, two members of the mating pool are randomly selected and combined by taking a random blend of each member's parameters, e.g. the crossover of parameter R_s:

   $$R_{s,child} = \alpha R_{s,parent1} + (1-\alpha) R_{s,parent2} \tag{49}$$

   where α is a random variable selected from a uniform distribution over the interval $[0,1]$

2. **Mutation**: in the mutation process, a member of the mating pool is randomly selected and its parameters are mutated by adding Gaussian noise with parameter-dependent standard deviations (see Table 2).

Parameter	Standard Deviation (σ)
R_s	0.01
X_s	0.01
X_m	0.33
R_{r1}	0.01
X_{r1}	0.01
R_{r2}	0.01
X_{r2}	0.01
R_c	6.67

Table 2: Standard deviations for mutation noise

The fitness of the next generation is then calculated and the process repeats itself for n_{gen} generations.

The default settings for the genetic algorithm implemented in **Moto** for motor parameter estimation are shown in Table 3. A flowchart of the genetic algorithm implemented in **Moto** is shown in Figure 9.

Setting	Setting Description	Default Value
n_{pop}	Population of each generation	20
n_{pool}	Number of members in the mating pool	15
n_e	Number of elite children	2
c_f	Crossover fraction	80%
n_{gen}	Maximum number of generations	30

Table 3: Default settings for genetic algorithm

3.5 Hybrid Algorithms

It was shown in [14] that linear restrictions imposed on R_s and X_{r2} have a significant influence on the convergence and error rates of descent algorithms. The parameters R_s and X_{r2} are also difficult to estimate accurately based solely on commonly available manufacturer data. Moreover, the selection of initial conditions can also affect the performance of descent algorithms.

On the other hand, natural optimisation algorithms can yield lower average error rates, but never low enough to qualify for convergence (as defined by a squared error of $< 1 \times 10^{-5}$). However, the performance of natural optimisation algorithms is unaffected by the choice of initial conditions.

Hybrid algorithms attempt to overcome the limitations of descent algorithms by applying a genetic algorithm to select R_s and X_{r2}. In other words, a baseline descent algorithm (e.g. NR, Damped NR, LM, etc) is run with fixed values for R_s and X_{r2}, which are in turn iteratively selected using a genetic algorithm in an outer loop. A

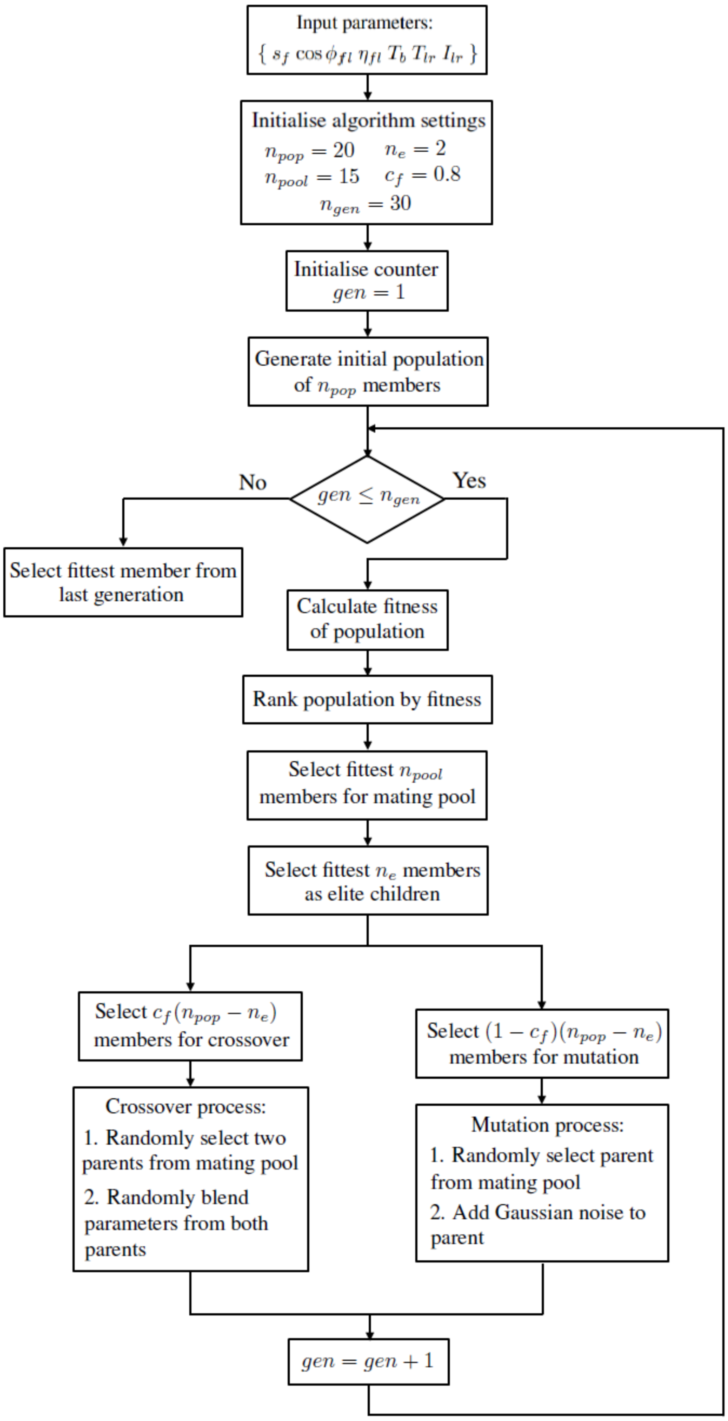

Input parameters:
$\{ s_f \cos\phi_{fl}\ \eta_{fl}\ T_b\ T_{lr}\ I_{lr} \}$
Initialise algorithm settings
$n_{pop} = 20$ $n_e = 2$
$n_{pool} = 15$ $c_f = 0.8$
$n_{gen} = 30$
Initialise counter
$gen = 1$
Generate initial population of n_{pop} members
No
$gen \leq n_{gen}$
Yes
Select fittest member from last generation
Calculate fitness of population
Rank population by fitness
Select fittest n_{pool} members for mating pool
Select fittest n_e members as elite children
Select $c_f(n_{pop} - n_e)$ members for crossover
Select $(1 - c_f)(n_{pop} - n_e)$ members for mutation
Crossover process:
1. Randomly select two parents from mating pool
2. Randomly blend parameters from both parents
Mutation process:
1. Randomly select parent from mating pool
2. Add Gaussian noise to parent
$gen = gen + 1$

Figure 9: Flowchart for genetic algorithm

flowchart of the proposed hybrid algorithm is shown in Figure 10. A more detailed description of the proposed algorithm follows.

An initial population of n_{pop} estimates for R_s and X_{r2} are randomly sampled from a uniform distribution with upper and lower limits as shown in Table 4. Each pair of estimates is referred to as a member of the population.

Parameter	Range of Initial Estimate (pu)	
	Lower Bound	Upper Bound
R_s	0	0.15
X_{r2}	0	0.15

Table 4: Range of initial parameter estimates

The descent algorithm is then run on each member of the population. The fitness of each member (in terms of the squared error $\mathbf{F}'\mathbf{F}$) is calculated and ranked. The lowest fitness members are discarded and the rest are retained to form the mating pool for the next generation (there are n_{pool} members in the mating pool).

The fittest n_e members in the mating pool are retained for the next generation as **elite children**. Of the remaining $n_{pop} - n_e$ children to be created for the next generation, $c_f\%$ will be produced by crossover and the rest $(1 - c_f\%)$ by mutation. The proportion c_f is called the **crossover fraction**.

1. **Crossover**: in the crossover process, two members of the mating pool are randomly selected and combined by taking a random blend of each member's parameters, e.g. the crossover of parameter R_s:

$$R_{s,child} = \alpha R_{s,parent1} + (1 - \alpha) R_{s,parent2} \tag{50}$$

   where α is a random variable selected from a uniform distribution over the interval $[0, 1]$

2. **Mutation**: in the mutation process, a member of the mating pool is randomly selected and its parameters are mutated by adding Gaussian noise with standard deviations of 0.01.

The descent algorithm is then run for the next generation of estimates for R_s and X_{r2}. The fitness is calculated and the process repeats itself for n_{gen} generations. If at any point during the process the descent algorithm converges, then the hybrid algorithm stops and selects the parameter estimates from the converged descent algorithm as the solution. Otherwise, the parameter estimates yielding the best fitness after n_{gen} generations are selected.

The recommended settings for the hybrid algorithm implemented in **Moto** are shown in Table 5.

Setting	Setting Description	Default Value
n_{pop}	Population of each generation	15
n_{pool}	Number of members in the mating pool	10
n_e	Number of elite children	2
c_f	Crossover fraction	80%
n_{gen}	Maximum number of generations	10

Table 5: Recommended settings for hybrid algorithm

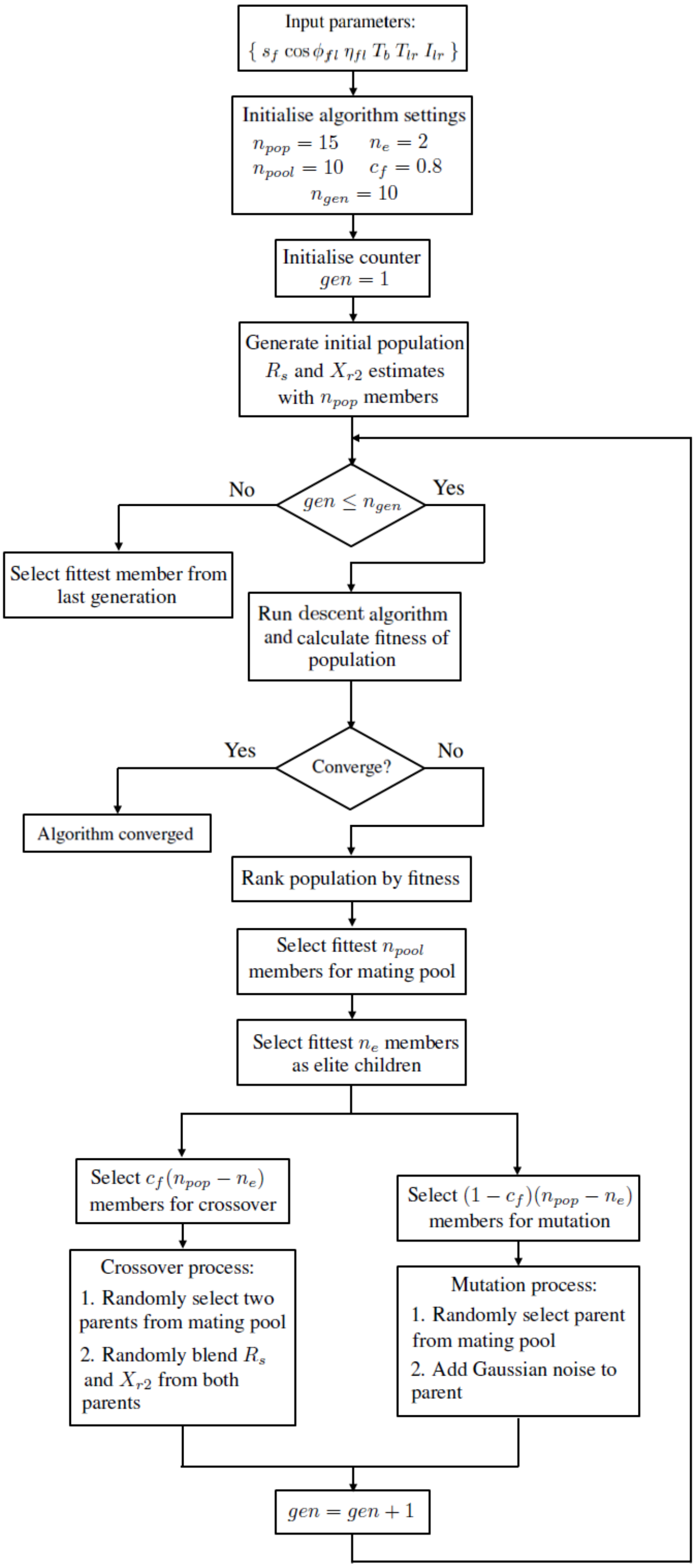
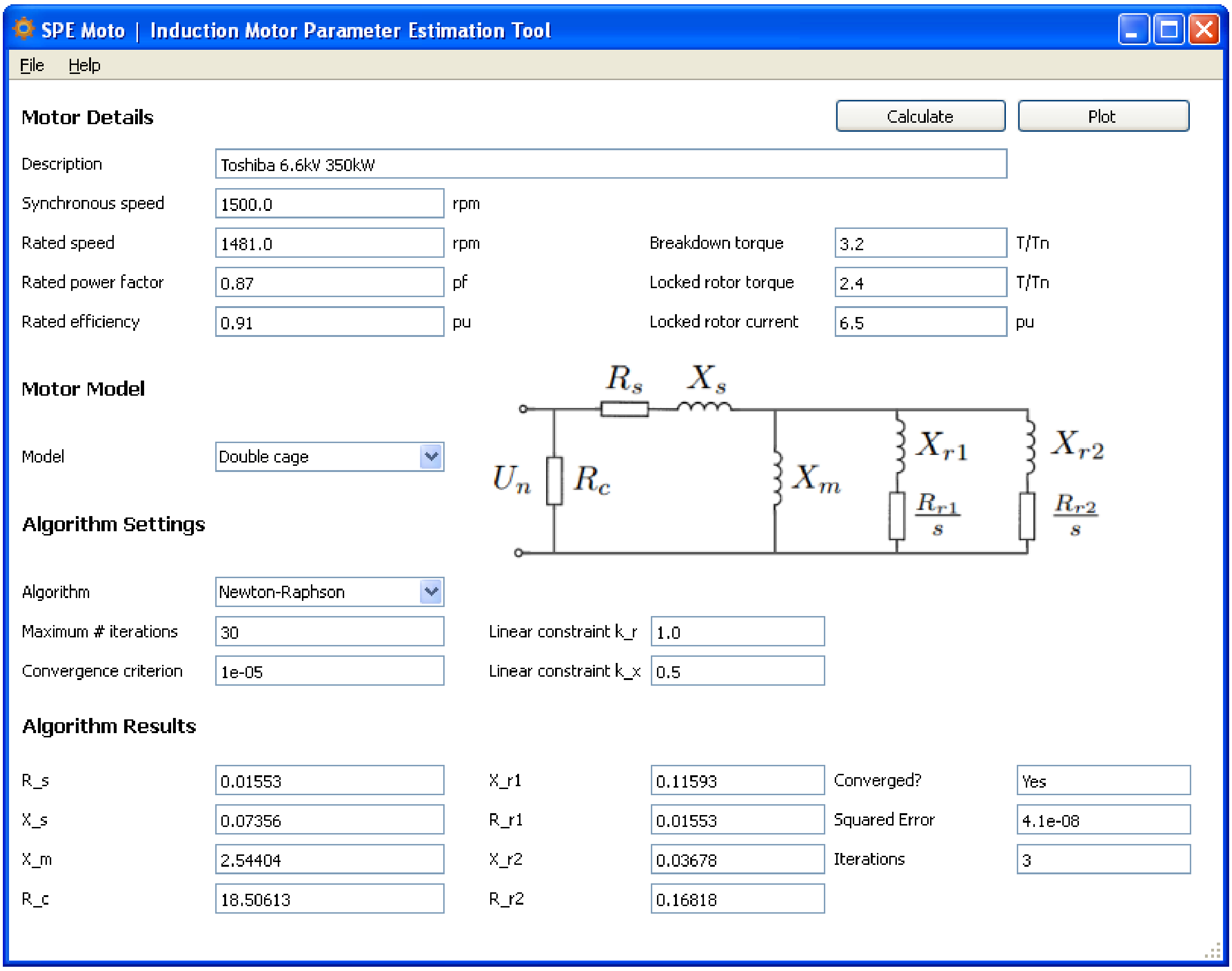

Input parameters:
$\{\ s_f\ \cos\phi_{fl}\ \eta_{fl}\ T_b\ T_{lr}\ I_{lr}\ \}$
Initialise algorithm settings
$n_{pop} = 15$ $n_e = 2$
$n_{pool} = 10$ $c_f = 0.8$
$n_{gen} = 10$
Initialise counter
$gen = 1$
Generate initial population
R_s and X_{r2} estimates
with n_{pop} members
$gen \leq n_{gen}$
No
Yes
Select fittest member from last generation
Run descent algorithm and calculate fitness of population
Converge?
Yes
No
Algorithm converged
Rank population by fitness
Select fittest n_{pool} members for mating pool
Select fittest n_e members as elite children
Select $c_f(n_{pop} - n_e)$ members for crossover
Select $(1 - c_f)(n_{pop} - n_e)$ members for mutation
Crossover process:
1. Randomly select two parents from mating pool
2. Randomly blend R_s and X_{r2} from both parents
Mutation process:
1. Randomly select parent from mating pool
2. Add Gaussian noise to parent
$gen = gen + 1$

Figure 10: Flowchart for hybrid algorithm (with natural selection of R_s and X_{r2})

4 Using the Program

Moto is quite straightforward to use. Simply fill in the details of the motor at the top the main screen (see Figure 11), select the model and algorithm settings and then press the"Calculate" button. The resulting equivalent circuit parameters will appear in per unit values at the bottom of the screen. The torque-speed and current-speed curves implied by the estimated parameters can then be viewed by pressing the "Plot" button (see Figure 12).



SPE Moto | Induction Motor Parameter Estimation Tool

Motor Details

Description: Toshiba 6.6kV 350kW

Synchronous speed: 1500.0 rpm

Rated speed: 1481.0 rpm

Rated power factor: 0.87 pf

Rated efficiency: 0.91 pu

Breakdown torque: 3.2 T/Tn

Locked rotor torque: 2.4 T/Tn

Locked rotor current: 6.5 pu

Motor Model

Model: Double cage

Algorithm Settings

Algorithm: Newton-Raphson

Maximum # iterations: 30

Convergence criterion: 1e-05

Linear constraint k_r: 1.0

Linear constraint k_x: 0.5

Algorithm Results

R_s	0.01553	X_r1	0.11593	Converged?	Yes
X_s	0.07356	R_r1	0.01553	Squared Error	4.1e-08
X_m	2.54404	X_r2	0.03678	Iterations	3
R_c	18.50613	R_r2	0.16818		

Figure 11: **Moto** main screen

4.1 Algorithm Settings

Algorithm: select between the seven algorithms described in Section 3

Maximum # iterations: is applicable to descent and hybrid algorithms and is the maximum number of iterations that the algorithm will execute without reaching convergence.

Convergence criterion: is the squared error value that the algorithms use as the convergence point. An algorithm will stop if it produces an estimate with a squared error below this covergence criterion. By default, the convergence criterion is 1×10^{-5}.

Linear constraint k_r: is only applicable to descent algorithms. The linear constraint k_r is the constant factor used to calculate the stator resistance R_s (refer to Section 3.1.1).

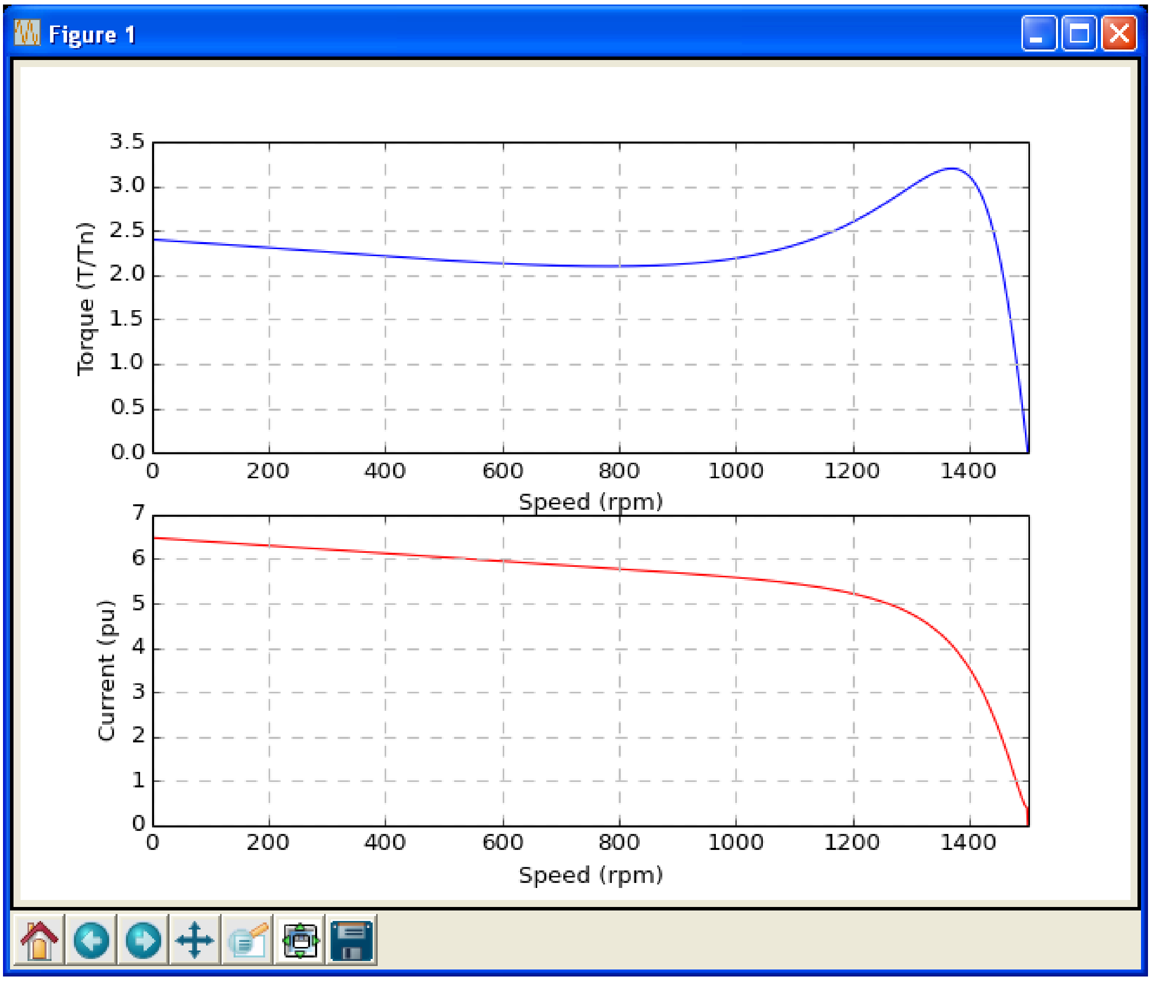


Figure 12: **Moto** plotting function

Linear constraint k_x: is only applicable to descent algorithms. The linear constraint k_x is the constant factor used to calculate the outer cage rotor reactance X_r2 (refer to Section 3.1.1).

Maximum # generations: is applicable to genetic and hybrid algorithms and is the maximum number of generations that the algorithm will evolve (refer to Section 3.4).

Members in population: is applicable to genetic and hybrid algorithms and is the number of members in each generation of estimates (refer to Section 3.4).

Members in mating pool: is applicable to genetic and hybrid algorithms and is the number of members that are retained for mating in each generation (refer to Section 3.4).

Elite children: is applicable to genetic and hybrid algorithms and is the number of elite children that will be spawned for each generation (refer to Section 3.4).

Crossover fraction: is applicable to genetic and hybrid algorithms and is the number of children that will be created through the crossover process (refer to Section 3.4).

4.2 Algorithm Results

R_s: is the stator resistance (in per unit)

X_s: is the stator reactance (in per unit)

X_m: is the magnetising reactance (in per unit)

R_c: is the core loss resistance (in per unit)

X_r1: is the inner cage rotor reactance (in per unit)

R_r1: is the inner cage rotor resistance (in per unit)

X_r2: is the outer cage rotor reactance (in per unit)

R_r2: is the outer cage rotor resistance (in per unit)

Converged: is a Yes / No status indicating whether the algorithm has converged.

Squared Error: is the resulting squared error of the algorithm

Iterations: is the number of iterations (or generations) that the algorithm executed

4.3 Saving and Loading

Moto can save and load otor data using the "Save as..." and "Open File..." items respectively in the File menu. The motor details and algorithm settings are saved in the **Moto** data files. Note that algorithm results are not saved and need to be re-calculated.

4.4 Plotting

The torque-speed and current-speed curves of the motor can be shown by pressing the "Plot" button. The resulting figure can be panned, zoomed and saved as an image using the toolbar at the bottom of the dialog box.

5 Troubleshooting

5.1 Algorithm Not Converging

Not all motors have a convergent solution. In some cases, the motor's performance cannot be characterised by a single cage or double cage model with constant parameters. In such cases, the best approximation is desired and this is application-dependent. For example, if a motor starting study is required, then a more accurate representation of the motor's starting characteristics is more important than high accuracy at low slip values. Thus the motor parameters can be tuned to result in better starting performance while perhaps sacrificing accuracy at motor breakdown.

In other cases, a valid solution exists, but the selected algorithm is not reaching it. Here, the squared error value is an important guide to check whether changes in algorithms and settings are having a positve or negative effect.

Based on our experience, we recommend the following general workflow for solving induction motor parameter estimation problems using double cage models:

1. As an initial attempt, use the conventional Newton-Raphson algorithm with fixed linear constraints $k_r = 1$ and $k_x = 0.5$
2. If there is no convergence, use the damped NR algorithm with fixed linear constraints $k_r = 1$ and $k_x = 0.5$
3. If there is no convergence, try the LM algorithm with fixed linear constraints $k_r = 1$ and $k_x = 0.5$
4. If there is no convergence, try the hybrid DNR-GA algorithm
5. If there is no convergence, try the hybrid LM-GA algorithm
6. Finally, if there is still no convergence, use the genetic algorithm with 50 to 100 generations to give a solution with an adequately low squared error value

5.2 Bugs

If you find a bug in the software, please send us a mesasge at contact@sigmapower.com.au.

6 References

[1] I. Boldea and S. Nasar. *The Induction Machine Handbook.* CRC Press, 2002.

[2] M. Haque. Determination of nema design induction motor parameters from manufacturer data. *IEEE Transactions on Energy Conversion*, 23(4), 2008.

[3] R. L. Haupt and S. E. Haupt. *Practical Genetic Algorithms.* John Wiley and Sons, 1998.

[4] B. K. Johnson and J. R. Willis. Tailoring induction motor analytical models to fir known performance characteristics and satisfy particular study needs. *IEEE Transactions on Power Systems*, 6(3), 1991.

[5] K. Levenberg. A method for the solution of certain non-linear problems in least squares. *The Quarterly of Applied Mathematics*, 2, 1944.

[6] D. Lindenmeyer, H. W. Dommel, A. Moshref, and P. Kundur. An induction motor parameter estimation method. *Electrical Power and Energy Systems*, 23:251–262, 2001.

[7] D.W. Marquardt. An algorithm for least-squares estimation of non-linear parameters. *Journal of the Society for Industrial and Applied Mathematics*, 11(2), 1963.

[8] P. Nangsue, P. Pillay, and S. E. Conry. Evolutionary algorithms for induction motor parameter determination. *IEEE Transactions on Energy Conversion*, 14(3), 1999.

[9] R Nolan, P Pillay, and T Haque. Application of genetic algorithms to motor parameter determination. In *Proceedings of 1994 IEEE Industry Applications Society Annual Meeting*, pages 47–54, 1994.

[10] J. Pedra. Estimation of induction motor double-cage model parameters from manufacturer data. *IEEE Transactions on Energy Conversion*, 19(2), 2004.

[11] J. Pedra. Estimation of typical squirrel-cage induction motor parameters for dynamic performance simulation. *IEEE Proceedings on Generation, Transmission and Distribution*, 153(2), 2006.

[12] J. Pedra. On the determination of induction motor parameters from manufacturer data for electromagnetic transient programs. *IEEE Transactions on Power Systems*, 23(4), 2008.

[13] G. Rogers and D. Shirmohammadi. Induction machine modelling for electromagnetic transient program. *IEEE Transactions on Energy Conversion*, EC-2(4), 1987.

[14] J Susanto and S Islam. Estimation of Induction Motor Parameters Using Hybrid Algorithms for Power System Dynamic Studies. In *Australasian Universities Power Engineering Conference, 2013. AUPEC '13*, number October, 2013.

[15] S. S. Waters and R. D. Willoughby. Modeling induction motors for system studies. *IEEE Transactions on Industry Applications*, 1A-19(5), 1983.

[16] H.H. Weatherford and C.W. Brice. Estimation of induction motor parameters by a genetic algorithm. In *Conference Record of the 2003 Annual Pulp and Paper Industry Technical Conference, 2003.*, pages 21–28, 2003.