Available online at www.sciencedirect.com
ScienceDirect
Control Engineering Practice 16 (2008) 787–797
CONTROL ENGINEERING PRACTICE
www.elsevier.com/locate/conengprac

Application of neural network in suppressing mechanical vibration of a permanent magnet linear motor

Hassan Yousefi[a,*], Markus Hirvonen[a], Heikki Handroos[a], Azita Soleymani[b]

[a]*Institute of Mechatronics and Virtual Engineering, Department of Mechanical Engineering, Lappeenranta University of Technology, P.O. Box 20, FIN-53851, Lappeenranta, Finland*
[b]*Department of Chemical Technology, Lappeenranta University of Technology, Finland*

Received 15 March 2006; accepted 10 August 2007
Available online 24 October 2007

Abstract

In this study, a recurrent neural network compensator for suppressing mechanical vibration in a permanent magnet linear synchronous motor (PMLSM) is studied. The linear motor is controlled by a conventional PI velocity controller, and the vibration of the flexible mechanism is suppressed by using a hybrid recurrent neural network. The differential evolution strategy and Kalman filter method are used to avoid the local minimum problem, and estimate the states of system, respectively. The proposed control method is firstly designed by using a nonlinear simulation model built in Matlab Simulink and then implemented in a practical test rig. The proposed method works satisfactorily and suppresses the vibration successfully.
© 2007 Elsevier Ltd. All rights reserved.

Keywords: Neural networks; Genetic algorithms; Velocity control; Linear motors; Kalman filters

1. Introduction

Neural networks have been studied in many areas from finance to technology for many years. The first innovations in this area were made in the early 1940s, and after that a number of different neural network structures and teaching algorithms have been developed (Haykin, 1994). Nowadays the development of computers and algorithms allows the application of neural networks to practical problems.

The linear motor is an old invention but it is only recently that, as a result of the development of permanent magnets and their decreased costs, permanent magnet (PM) linear motors have become a viable alternative to rotating motors fitted with linear transmissions. The linear motor simplifies the mechanical structure, eliminating the contact-type nonlinearities caused by backlash, friction, and compliance. Linear motors are used in applications where accurate positioning and fast dynamics are needed.

A great deal of attention is paid to the control of the linear motor itself (Otten, de Vries, van Amorengen, Rankers, & Gaal, 1997) where a neural network controller is implemented for a linear motor motion control. In that case the vibration of the tool mechanism, reel, gripper or any apparatus connected to the motor is not taken into account. This might reduce the capability of the machine system to carry out its assignment and impair the lifetime of the equipment. Nonetheless, it is usually more important to know how the load of the motor behaves. The load control of these motors has been less studied. The aim of the load controller is to drive the flexible load to a reference value in such a way that the load follows the reference value as accurately as possible, but without awkward vibration. One of the most traditional methods to suppress resonance in the electromechanical system is to allow only slow changes in the reference command. For example, different kinds of filters are used in a reference signal to suppress mechanical vibration. Dumetz, Vanden Hende, and Barre (2001) have studied bi-quad and low pass filters in a control loop but also as a reference filter. The closed loop filter makes it possible to compensate poles and zeros

*Corresponding author. Tel.: +3585 6212482; fax: +3585 6212499.
E-mail address: Yousefi@lut.fi (H. Yousefi).

0967-0661/$ - see front matter © 2007 Elsevier Ltd. All rights reserved.
doi:10.1016/j.conengprac.2007.08.003

of the transfer function from the motor side, and the reference filter compensates poles of the transfer function on the load side. Another widely used filter for vibration suppression is the Notch filter (Ellis & Lorenz, 2000). The drawbacks of the filtering are the low sensitivity to parameter variations, and the reduction of the dynamical properties of a servo system.

The state feedback control is another widely used method in vibration suppression. The method needs not only the values of the measured states, but also the information of unknown states. One possibility to get all of these values is to use an estimator. One example of this kind of control system is in Jun-Keun and Seung-Ki (1995), where a LQ-based speed controller with a Kalman filter in a two-mass motor drive system is studied. In the method, the motor is controlled by a simple PI controller and load acceleration can be measured or estimated and used as a compensation feedback. Kang and Sul (2000) and Lee, Kang, and, Sul (1999) have used this kind of a method successfully in the vibration control of elevators. Montanaro and Beale (1998) have combined the LQG method and acceleration compensation.

In the study, the idea of acceleration compensation of a flexible mechanism has been extended so that the compensation signal is produced by a neural network. The advantage of neural networks is their ability to pick up dependency even from noisy signals and the drawbacks are they increase the computational time and are considered as a black box in the establishment of the model, hiding any information between the input and the output.

Therefore, neural networks cannot be used in the systems where sufficient measurement data are not available. One possibility to reduce the calculation time is to teach the network in the simulation model, and after that finalise the teaching by using measurement data from a physical system. In the study by using the differential evolution, the final weights of neural network have been found. After the weights are finalised, they are transferred to the neural network compensator in the experimental setup. The states for the neural network compensator are estimated using the Kalman filter.

The paper is organised as follows. In Section 2, system model is presented. Section 3 presents controller design steps. The technical details of applying current controller, back propagation algorithm, differential evolution (DE) strategy, reference model and state estimation are presented in Section 3. Section 4 describes results for evaluating the reliability and performance of neural network. In Section 5 the conclusions are summarised.

2. System model

The modelling of the dynamics of the linear synchronous motor examined in this paper is based on the space-vector theory. The time-varying parameters are eliminated and all the variables are expressed on orthogonal or mutually

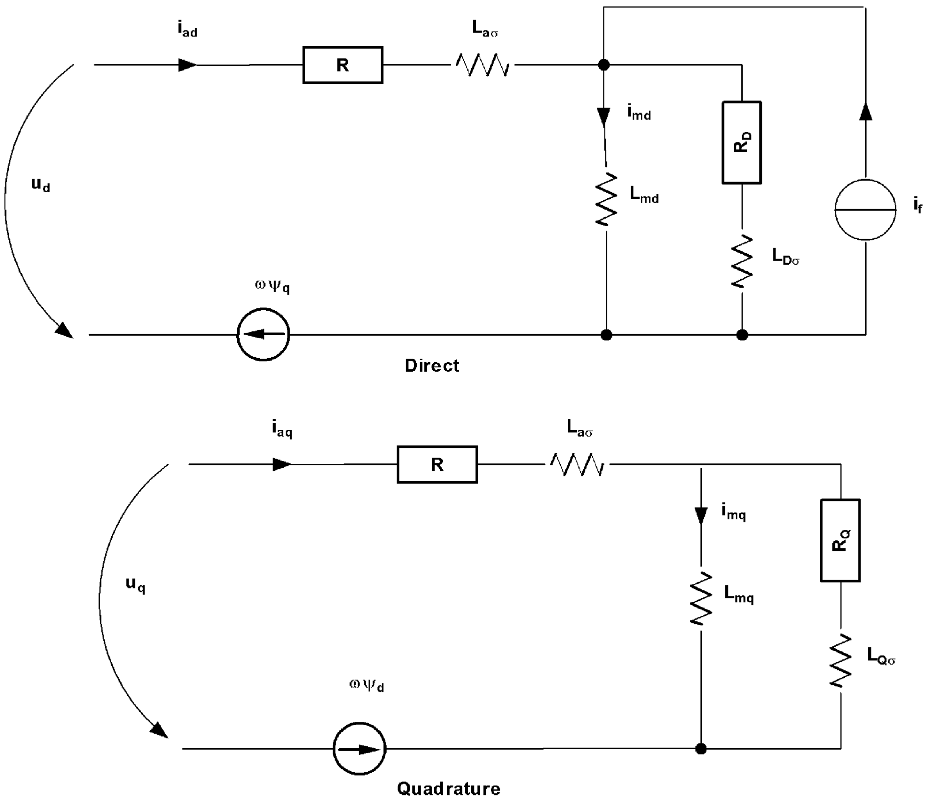

Fig. 1. Two axial model of the linear synchronous motor.

decoupled direct and quadrature axes, which move at a synchronous speed of ω_s. The d- and q-axis equivalent to the circuit of the permanent magnet linear synchronous motor (PMLSM) are shown in Fig. 1, and the corresponding equations are (1) and (2), respectively.

The voltage equations for the synchronous machines are

$$u_d = Ri_{ad} + \frac{d\psi_d}{dt} - \omega_s\psi_q, \tag{1}$$

$$u_q = Ri_{aq} + \frac{d\psi_q}{dt} + \omega_s\psi_d, \tag{2}$$

where u_d and u_q are the d- and q-axis components of the terminal voltage, i_{ad} and i_{aq} the d- and q-axis components of the armature current, R is the armature winding resistance and ψ_d, ψ_q are the d- and q-axis flux linkage components of the armature windings. The synchronous speed can be expressed as $\omega_s = \pi v_s/\tau$, where v_s is the linear synchronous velocity, i.e. the velocity in which the motor moves, and τ the pole pitch, i.e. the length of the pole pair of permanent magnets. Although the physical system does not contain a damper, which in PMLSM usually takes the form of an aluminium cover on the PMs, virtual damping must be included in the model due to eddy currents. The voltage equations of the short-circuited damper winding are

$$0 = R_D i_D + \frac{d\psi_D}{dt}, \tag{3}$$

$$0 = R_Q i_Q + \frac{d\psi_Q}{dt}, \tag{4}$$

where R_D and R_Q are the d- and q-axis components of the damper winding resistance and i_D and i_Q the d- and q-axis components of the damper winding current. The armature and damper winding flux linkages in the above equations are

$$\psi_d = L_{ad}i_{ad} + L_{md}i_D + \psi_{pm}, \tag{5}$$

$$\psi_q = L_{aq}i_{aq} + L_{mq}i_Q, \tag{6}$$

$$\psi_D = L_{md}i_{ad} + L_D i_D + \psi_{pm}, \tag{7}$$

$$\psi_Q = L_{mq}i_{aq} + L_Q i_Q, \tag{8}$$

where L_{ad} and L_{aq} are the d- and q-axis components of the armature self-inductance, L_D and L_Q the d- and q-axis components of the damper winding inductance, L_{md} and L_{mq} the d- and q-axis components of the magnetising inductance and ψ_{pm} the flux linkage per phase of the permanent magnet. By solving the flux linkage differential equations from (1) to (4) and substituting the current equations from (5) to (8) into these equations, the equations for the simulation model of the linear motor can be derived. The electromagnetic thrust of a PMLSM is

$$F_{dx} = \frac{p_e}{v_s} = \frac{3p}{2}\frac{\pi}{\tau}(\psi_d i_{aq} - \psi_q i_{ad}), \tag{9}$$

where p_e is the electromechanical power and p represents the number of pole-pairs.

2.1. Non-idealities of PMLSM

The force ripple ripple in an iron-core PM linear motor has been studied by Zhao and Tan (2005). The force ripple of the PMLSM is larger than that of rotary motors because of the finite length of the stator or mover and the wide slot opening. In the PMLSM, the thrust ripple is caused mainly by the detent force generated between the PMs and the armature. This type of force can be divided into two components: tooth and core-type detent force. The core-type detent force can be efficiently reduced by optimising the length of the moving part or smooth forming the edges of the mover and the tooth-type detent force can be reduced by skewing the magnets and chamfering the edges of the teeth (Hyun, Jung, Shim, & Yoon, 1999; Inoue & Sato, 2000; Jung & Jung, 2002).

The ripple of the detent force produces both vibration and noise and reduces controllability (Chun, Jung, & Jung, 2000). The force ripple is dominant at low velocities and accelerations. At higher velocities, the cogging force is relatively small and the influence of dynamic effects (acceleration and deceleration) is more dominant (Otten et al., 1997). The force ripple can be described by sinusoidal functions of the load position, x, with a period of φ and amplitude of A_r, i.e.

$$F_{\text{ripple}} = K_s A_{r1}\sin(\varphi_1 x)[A_{r1} + A_{r2}\sin(\varphi_2 x)], \tag{10}$$

where K_s is a scaling factor. In Fig. 2, the results of the simulation are compared with those measured in the reference system.

The model also takes into account the effect of friction. Friction is highly nonlinear and may result in steady-state errors, limit cycles and poor performance (Olsson, Åström, de Wit, Gäfvert, & Olsson, 1998). The friction model took into account the Coulomb (static) and viscous (dynamic) components

$$F_\mu = \text{sign}(v)[F_{\text{coulomb}} + \text{abs}(v)F_{\text{viscous}}], \tag{11}$$

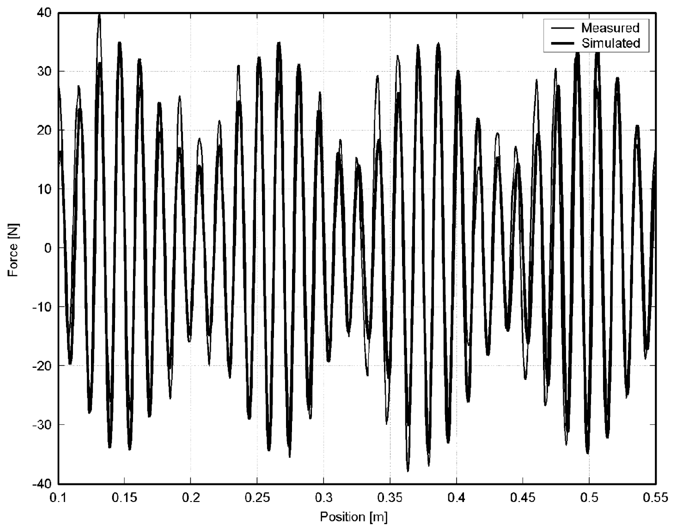

Fig. 2. Comparison of measured and simulated detent forces.

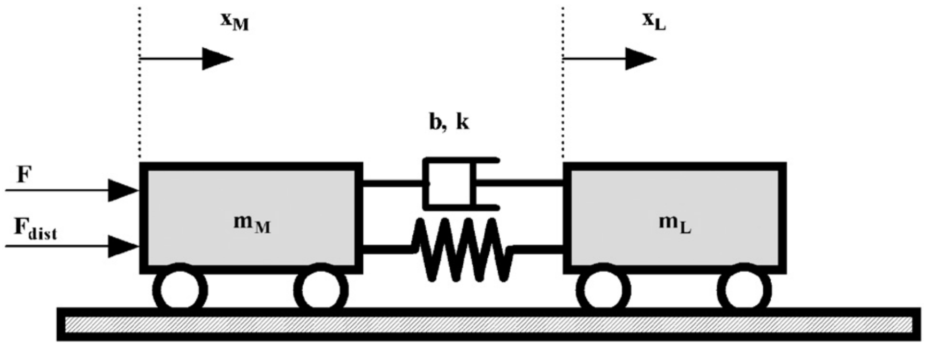

Fig. 3. Two-mass model of the system.

where v is the velocity of the motor. The main disadvantage when using highly nonlinear models for simulations or control purposes is high discontinuity at near-zero speed, which gives rise to problems such as numerical chatter. The total disturbance force equation can be described using the equations of detent force and friction force; i.e. the disturbance force F_{dist} is

$$F_{\text{dist}} = F_{\text{ripple}} + F_{\mu}, \tag{12}$$

This resultant disturbance force component is added to the electromotive force to influence the dynamical behaviour of the linear motor system.

The effect of load variation was also taken into consideration. The mechanism in this study is made according to the ACC Benchmark Problem (Wie & Bernstein, 1990), i.e. a two-mass model, Fig. 3. The ACC Benchmark Problem consists of two masses attached by a single spring, moving on a friction-free horizontal surface. The control input is the force applied to the first mass, and the output is the position or velocity of the second mass.

The motion equations of this system are

$$\begin{aligned} m_{\text{M}}\ddot{x}_{\text{M}} &= k\Delta x + b\Delta\dot{x} + F_{\text{T}}, \\ m_{\text{L}}\ddot{x}_{\text{L}} &= -k\Delta x - b\Delta\dot{x}, \end{aligned} \tag{13}$$

where k is the spring constant, b is the damping factor, m_{M} and m_{L} are motor and load masses, F_{T} is the summation of the controller force, F, and the disturbance force, F_{dist}, x_{M} and x_{L} are motor and load positions and Δx and $\Delta\dot{x}$ are compression of the spring and the velocity difference between masses, respectively.

3. Controller design

3.1. Current controller of the linear motor

The current control of the system is implemented in the form of vector control. Vector control is based on the space vector theory of electrical machines and, therefore, can easily be implemented in the motor model that is also based on the space vector theory. Generally in the vector control theory, the force and flux components are analysed separately from the motor currents using the mathematical model of the machine, and control algorithms control these components separately. In the vector control used in this study, the direct axis current i_{ad} is set to zero ($i_{\text{ad}} = 0$) assuming that it does not influence the generation of force; i.e. Eq. (9) transforms to

$$F_{\text{d}x} = \frac{3p}{2}\frac{\pi}{\tau}(\psi_{\text{d}} i_{\text{aq}}). \tag{14}$$

This means that angle θ, between the armature current and q-axis, always remains at 0° and that the thrust is proportional to the armature current $i_{\text{a}} = i_{\text{aq}}$.

3.2. The neural network design

In this study, the aim of the controller is velocity tracking of a flexible load. Classical approaches, such as P or PI regulators, do not provide satisfactory results. For this reason, a hybrid recurrent neural network controller is used to improve the response of the system. The aim of a neural network compensator is to increase the damping of the system. Compensation is achieved by cancellation of the vibration applying alternative force (against the ripple force). The structure of the hybrid controller is presented in Fig. 4.

One of the earliest recurrent neural networks was the Jordan network. The Jordan networks use input of the past output like a memory for a neural network. In the Jordan network, the activation values of the output units are feedback into the input layer through a set of extra input units called the *state units*. The numbers of state units are equal to the number of network outputs. The connections between the output and state units have a fixed weight of +1; learning takes place only in the connections between input and hidden units as well as hidden and output units.

In this study, the back propagation algorithm with a momentum term to online update the weights and biases of a neural network was used (Rumelhart, Hinton, & Williams, 1986). The back propagation algorithm is a learning scheme in which the error is back propagated layer by layer and used to update the weights. The algorithm is a gradient descent method that minimises the error between the desired outputs and the actual outputs calculated by the multi-layer perceptron (MLP).

The back propagation training process requires that the activation functions be bounded, differentiable functions.

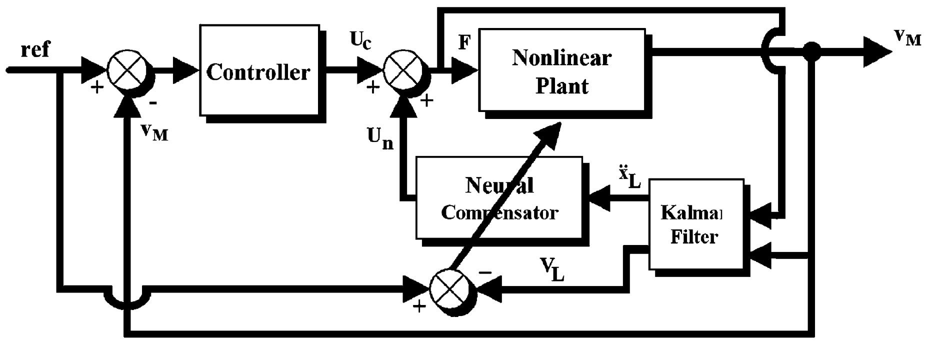

Fig. 4. Proposed neural network hybrid controller.

One of the most commonly used functions satisfying these requirements is the hyperbolic tangent function as follows:

$$f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}. \tag{15}$$

The learning procedure requires only that the change in weights and biases are proportional to $\partial E_p/\partial w$. True gradient descent requires that infinitesimal steps be taken. The constant of proportionality is the learning rate in the procedure. The higher the learning rate the greater the value changes in the weights of the neural network. One way to increase the learning rate without leading to oscillation is to modify the general delta rule to include a momentum term. This can be accomplished by the following rule:

$$\Delta w_{ji}(n+1) = \eta(\delta_{pj} o_{pi}) + \alpha \Delta w_{ji}(n), \tag{16}$$

where n indicates the presentation number. The parameter of η is the learning rate, and α is a constant which determines the effect of past changes on the current direction of movement in weight space, and $\Delta w(n)$ is the change in the weight in step n.

The training procedure can be determined from Fig. 4. When calculating the error, there is no target for the output of the neural network, so the error of the neural network cannot be updated from its output directly. To solve the problem, the error was defined based on the control target and it was defined as the difference between the reference velocity and the load velocity. Updating of the neural network is based on the extended back propagation method. The structure of updating is the same as the back propagation, but it has only one extra term (k_c).

The extra term is defined as follows:

$$k_c = \frac{a_{\text{load}}}{\dot{U}_n}, \tag{17}$$

where k_c is the extra gain for online updating of the neural network. There are two parameters in order to find the amount of k_c. The first parameter, a_{load}, is estimated by the Kalman filter directly, and the second parameter is calculated using the following equation:

$$\dot{U}_n = \frac{U_n(n) - U_n(n-1)}{T_s}, \tag{18}$$

where T_s is the sampling time. The k_c is bounded, because the term of $(U_n(n) - U_n(n-1))$ cannot be zero during online training. The term is the difference between the two successive outputs of the neural network. Following, the only two probable cases where the term will be zero are discussed.

The first case occurs when the neural network is saturated and the output of the neural network does not change with varying inputs. The saturation problem happens when the initial weights of the neural network are not selected correctly (local minimum) or the extent of the learning rate for online training is too great. These two conditions did not occur in this study, because the saturation problem was avoided. The DE algorithm is used to find the best values for the weights (global minimum) and during online training; the learning rate must be sufficient enough.

The second case happens when the system approaches to steady state condition (the error and the term of $(U_n(n) - U_n(n-1))$ approach zero). In this case the online updating of weights is stopped, because in a practical application of a neural network, the online update only happens when the error is bigger than the tolerance chosen by the designer (the neural network is trained until the error reaches a level of tolerance), so as is discussed, the term of $k_c$is bounded and it can be used in online training.

The new online updating for the neural network as the compensator can be defined as

$$\Delta w_{ji}(n+1) = \eta(\delta_{pj} o_{pi}) k_c + \alpha k_c \Delta w_{ji}(n). \tag{19}$$

Neural network parameters are updated online by learning rate and momentum factors. As shown in Fig. 5, the proposed recurrent neural network has three hidden, one input and one output layers. The inputs of the neural network are two normalised accelerations of load mass ($a_L(k)$, $a_L(k-1)$) and the state unit, $u_n(k-1)$. The absolute value of network output is equal or less than one, so the network gain, G_n is used to get the proper compensation. Several kinds of structures with one or two hidden layers were tested. The proposed structure emulates the acceleration feedback and improves the dynamic behaviour of the system.

Choosing the weights is important in the back propagation algorithm. To avoid the local minimum problem, the DE algorithm is used to find the offline weights (Storn & Price, 1995).

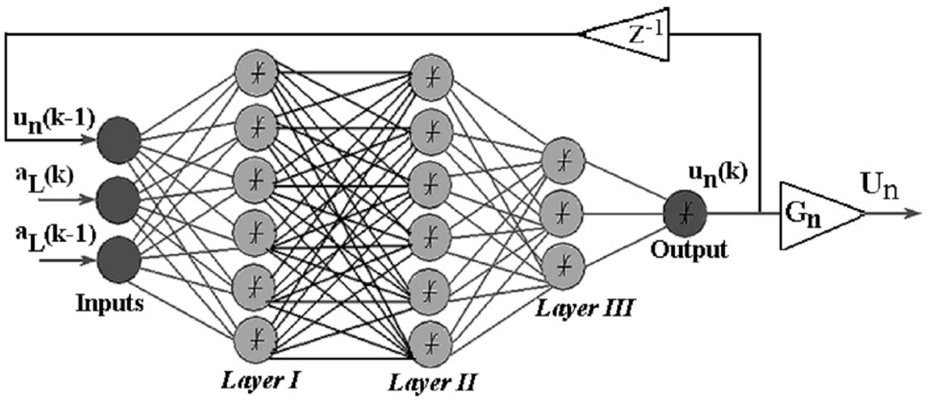

Fig. 5. The structure of the proposed recurrent neural network.

3.3. Differential evolution (DE) strategy

Global optimisation methods are under continuous development, and lately, genetic algorithms and evolution strategies have been found to be promising stochastic optimisation methods (Ilonen, Kamarainen, & Lampinen, 2003). DE is a simple and powerful population based, direct-search algorithm for globally optimising functions defined especially on functions with real-valued parameters. The schematic procedure of a class of DE is as follows (Corne, Dorigo, & Glover, 1999):

The DE inputs areas are as follows:

$$D, G_{\max}, NP \geqslant 4, F_s \in (0, 1+), CR \in [0,1], x^{(lo)}, x^{(hi)}, \tag{20}$$

where D is the number of parameters, NP is the population size, F_s is the scale factor, CR is the crossover control constant, $G_{\max}$ is the maximum number of generation, and hi, lo are upper and lower initial parameter bounds, respectively.

Initialize:

$$\begin{cases} \forall i \leqslant NP \wedge \forall j \leqslant D: \\ x_{j,i,G=o} = x_{j,i}^{(lo)} + \text{rand}[0,1](x_j^{(hi)} - x_j^{(lo)}) \end{cases} \tag{21}$$

```
while $G > G_{\max}$
    $\forall i \leqslant NP$ :
        Mutate and recombine :
            $r_1, r_2, r_3 \in \{1, 2, \ldots, NP\}$,
            randomly selected $(r_1 \neq r_2 \neq r_3 \neq i)$
            $j_r$ randomly selected once each $i$
            $\forall j \leqslant D$,
            if $(\text{rand}_j[0,1) < CR \vee (j = j_r))$
            $u_{j,i,G+1} = x_{j,r_3,G} + F_s.(x_{j,r_1,G} - x_{j,r_2,G})$,
            otherwise
            $u_{j,i,G+1} = x_{j,i,G}$
        Select :
            $\vec{x}_{i,G+1} = \begin{cases} \vec{u}_{i,G+1} & \text{if } F_{\text{Co}}(u_{i,G+1}) \leqslant F_{\text{Co}}(x_{i,G}) \\ \vec{x}_{i,G} & \text{otherwise} \end{cases}$
    $G = G + 1$
```
(22)

Table 1
Parameters for differential evolution

Symbol	Parameter	Value
D	Number of parameters	91
NP	Number of population	120
F_s	Scale factor	0.8
R	Crossover control constant	0.7

Here, the number of initial weights for each neural network is 91 (for all layers except the output layer, a bias is considered). The selected parameters for this strategy are in Table 1.

The cost of the system is considered as a summation of absolute errors for one period (one second with a one millisecond sampling time) so the cost is a summation of absolute errors for one thousand points. The initial upper and lower bounds are [1,−1], respectively. Note that generally the number of the population (NP) is five times the number of parameters (D). Here, because of a memory saturation problem, the maximum NP was 120. The results show that DE can find the global minimum cost for the system. Depending on the expected value of the cost, the DE will find the proper weights. The cost of the system is defined as follows:

$$F_{\text{Co}}(x_{i,G}) = \sum_{k=1}^{K=1000} |e(k)|, \tag{23}$$

where

$$|e(k)| = |V_{\text{ref}}(k) - V_{\text{load}}(k)|. \tag{24}$$

The reference velocity V_{ref} in Eq. (24) is from the reference model introduced in Eq. (25) and V_{load} is the estimated load velocity. The minimum cost for the system was 5.6 and the related weights are used as initial values of weights for the feed forward neural network with a back propagation algorithm. Fig. 6 is the plot of the cost when the DE algorithm is used to search for the global minimum of the weights. Fig. 7 shows the amount of one of the network sample weights (first weight of the first layer) versus the iterations. It is clear that the weights approach

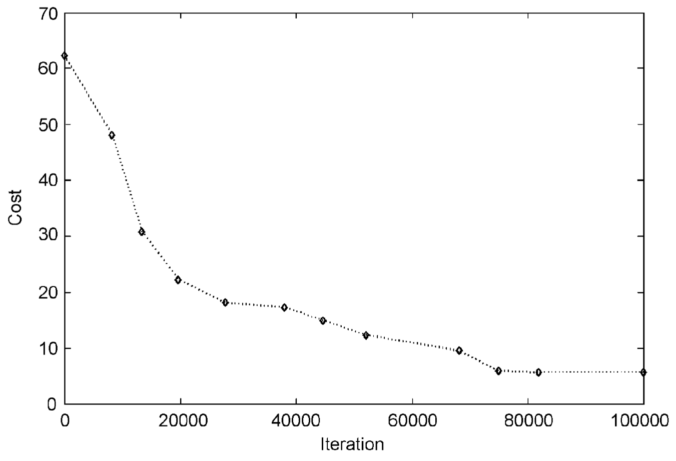

Fig. 6. Cost of the system against the number of its iterations.

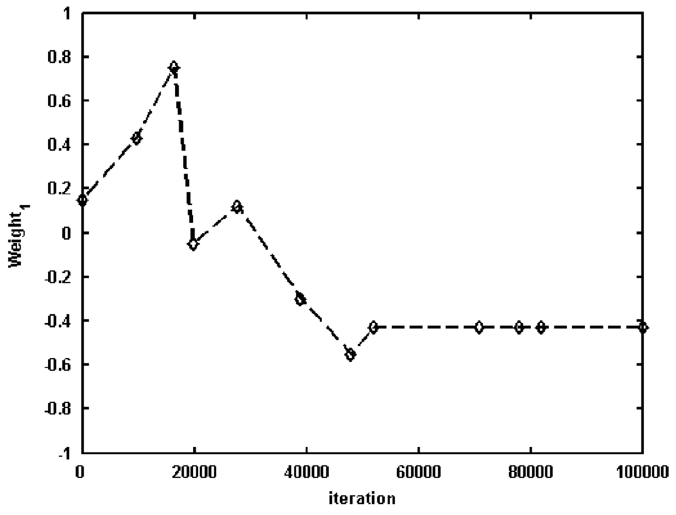

Fig. 7. A sample weight connecting the input layer to the first hidden layer of network verses the number of iterations.

(−0.43) after 52,000 iterations and then it remains at the final value. Therefore, the results show that the weights that converge to the specific amounts and network in the condition of offline working are stable.

3.4. Reference model

The desired linear, second order reference model was selected to run parallel with the nonlinear system. There are different kinds of methods that can be used to find a reference model, such as the ITAE or the Bessel transfer functions. In this study, the Bessel function is used (Franklin, Powell, & Emami-Naeini, 1994). The natural frequency of the system was chosen so that the response of the system is as fast as possible. The chosen damping ratio provides the minimum overshoot of the reference model. The natural frequency ω_n of this model is set to equal 100 rad/s with a damping ratio ζ of 1.732. The reference model is

$$G_{\mathrm{ref}}(s) = \frac{\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2}. \tag{25}$$

3.5. State estimation

States for the controller were estimated using the Kalman filter. In deriving the Kalman filter equations, the objective is to find an equation that computes a posterior state estimate $\hat{\mathbf{x}}_k$ as a linear combination of a prior estimate $\hat{\mathbf{x}}_k^-$ and a weighted difference between a measurement $\mathbf{y}_k$ and a measurement prediction $\mathbf{H}\hat{\mathbf{x}}_k^-$ (Leleux, Claps, Chen, Tittel, & Harman, 2001). The linear difference equations for the discrete Kalman Filter are

$$\mathbf{x}_{k+1} = \mathbf{\Phi}\mathbf{x}_k + \mathbf{\Gamma}\mathbf{u}_k + \mathbf{w}_k, \tag{26}$$

$$\mathbf{y}_k = \mathbf{H}\mathbf{x}_k + \mathbf{v}_k, \tag{27}$$

where $\mathbf{\Phi}$, $\mathbf{\Gamma}$, and $\mathbf{H}$ are discretized state, input, and output matrices, respectively, $\mathbf{w}_k$ is a random process that accounts for the noise present due to mismodelling, and $\mathbf{v}_k$ is a random noise process accounting for measurement noise.

The first step in calculation estimations of the real-time Kalman filter is to initialise the process by suitable conjectures for the estimate state vector $\hat{\mathbf{x}}_k$ and the estimation error covariance $\mathbf{P}_k$. This procedure is called the time update:

$$\hat{\mathbf{x}}_k^- = \mathbf{\Phi}\hat{\mathbf{x}}_{k-1} + \mathbf{\Gamma}\mathbf{u}_{k-1}, \tag{28}$$

$$\mathbf{P}_k^- = \mathbf{\Phi}\mathbf{P}_{k-1}\mathbf{\Phi}^{\mathrm{T}} + \mathbf{Q}. \tag{29}$$

The next step is to calculate the Kalman gain and update the state estimation with measurements. This procedure is called the measurement update:

$$\mathbf{K}_k = \mathbf{P}_k^-\mathbf{H}^{\mathrm{T}}(\mathbf{H}\mathbf{P}_k^-\mathbf{H}^{\mathrm{T}} + \mathbf{R})^{-1}, \tag{30}$$

$$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k^- + \mathbf{K}_k(\mathbf{y}_k - \mathbf{H}\hat{\mathbf{x}}_k^-), \tag{31}$$

$$\mathbf{P}_k = (\mathbf{I} - \mathbf{K}_k\mathbf{H})\mathbf{P}_k^-, \tag{32}$$

where $\mathbf{I}$ is an $n \times n$ identity matrix. The estimated error covariance $\mathbf{P}_k$ is independent of the measurement. It could even be pre-calculated before any measurements are made. This is a direct consequence of the underlying assumption that the noise covariance matrices $\mathbf{Q}$ and $\mathbf{R}$ might be measured prior to operation of the filter (Stansfield, 2001).

For the Kalman filter, a linear state-space model of the mechanical system (13) is derived. The friction and other nonlinearities are assumed to be system noise, which the Kalman filter handles as a random process. The estimated states of the system are the velocity of the motor v_{M}, velocity of the load v_{L}, and spring force F_{s}, i.e. the state vector is

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} v_{\mathrm{M}} \\ v_{\mathrm{L}} \\ F_{\mathrm{s}} \end{bmatrix}. \tag{33}$$

Table 2
Mechanical parameters

Symbol	Parameter	Value
m_M	Motor mass	20 kg
m_L	Load mass	4 kg
K	Spring constant	13700 N m^{-1}

The state matrix **A**, input matrix **B** and output matrix **C** are described as

$$\mathbf{A} = \begin{bmatrix} 0 & 0 & -\dfrac{1}{m_M} \\ 0 & 0 & \dfrac{1}{m_L} \\ k & -k & 0 \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} \dfrac{1}{m_M} \\ 0 \\ 0 \end{bmatrix}, \quad \mathbf{C} = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}, \tag{34}$$

where k is the spring constant. Damping constant b is assumed to be negligible in respect of system dynamics. The control input u in the state space model is the motor thrust, F_T. The process noise covariance **Q** in this application is

$$\mathbf{Q} = \begin{bmatrix} 100 & 0 & 0 \\ 0 & 10 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{35}$$

and the measurement covariance $R = 0.01$ is scalar due to one input for the Kalman filter. In Table 2, mechanical parameters in the state space model are presented.

The acceleration estimation $\hat{\ddot{x}}_L$ used in the controller is defined from the estimated spring force $\hat{F}_s$ by dividing it by a mass of the load m_L, i.e. the acceleration estimation is

$$\hat{\ddot{x}}_L = \frac{\hat{F}_s}{m_L}. \tag{36}$$

4. Results

4.1. Verification of the simulation model

The simulation model was implemented and analysed in the MatLab/Simulink® software. The PWM inverter is modelled as an ideal voltage source and common Simulink blocks are used for the model. The time step of the integrator in the analysis was 10 μs, except for the velocity controller, which had a time step of 1 ms. The parameters used in the simulation are introduced in Table 3.

The simulation results were compared with those measured in the physical linear motor. Fig. 8 shows the schematic diagram of motor. The motor is a commercial three-phase linear synchronous motor application with a rated force of 675 N. The moving part (the mover) consists of a slotted armature, while the surface permanent magnets are mounted along the whole length of the path (the stator). The permanent magnets are slightly skewed (1.7°) in relation to the normal. Skewing the PMs reduces the detent force (Gieras & Piech, 2001). The moving part is set up on an aluminium base with four recirculating roller bearing blocks on steel rails. The position of the linear motor was measured using an optical linear encoder with a resolution of approximately 1 μm.

Table 3
Linear motor data

Symbol	Parameter	Value
F_N	Nominal force	675 N
A_N	Nominal current	7.2 A
K_m	Motor constant	94 N A^{-1}
R_P	Electric resistance	4.8 Ω
L_P	Winding inductance	20 mH
V_n	Maximum speed at nominal thrust	2.1 m s^{-1}
τ_M	Pole pitch (180°)	15 mm
P_N	Nominal power	3910 W
L	Stroke	1000 mm
Kp	P gain of vel. controller	10000 Ns m^{-1}
Ki	I gain of vel. controller	0.01 N m^{-1}

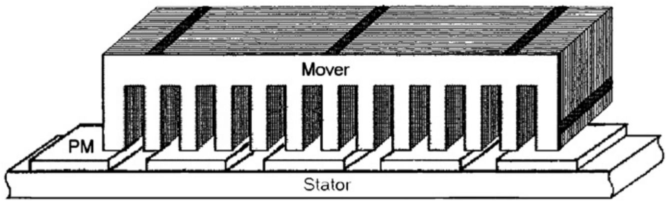

Fig. 8. The principle of PMLSM.

The physical linear motor application was driven in such a way that the PI velocity controller was implemented in Simulink to gain the desired force reference. The derived algorithm was transferred to C code for the dSPACE© digital signal processor (DSP) to use in real time. From the DS1103 card the force command F^* was fed into the drive of the linear motor. The computational time step for the velocity controller was 1 ms, while the current controller cycle was 31.25 μs.

The measured and simulated velocity responses and force generating quadrature currents are compared in Fig. 9. The sine function is used as a reference velocity.

4.2. Neural network compensator

The merits of the neural network compensator are its flexibility, intelligence and acceptable vibration suppression. The neural network compensator is tested in several test runs of the nonlinear simulation model and the laboratory setup. Fig. 10 represents the convergence of the load in the simulation model towards zero after an initial displacement of 0.02 m in the case of free vibration

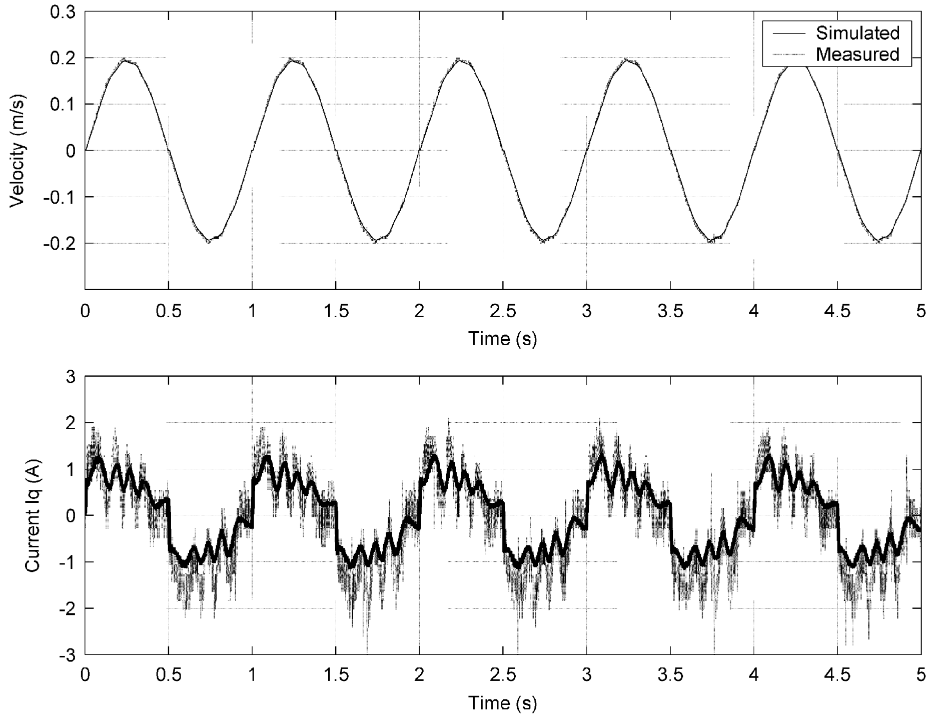

Fig. 9. A comparison of the measured and simulated velocity responses and quadrature currents in the case of a sine velocity reference.

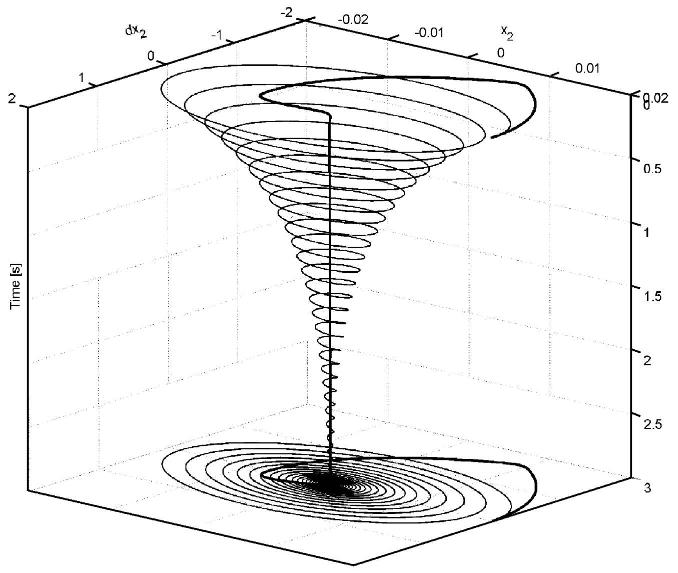

Fig. 10. Phase-plane figure of the load in the case of free vibration (thin), and a neural network-based controller (thick) when the initial displacement is 0.02 m.

of the load, and when the hybrid controller is used. This representation is called the phase-plane plot of the system, and it is usually used to analyse the stability of nonlinear systems. Parameters x_2 and dx_2 are load displacement and velocity, respectively. Figs. 11 and 12 show a comparison of the velocity responses in the case of PI controller with proposed compensator and without any compensator, respectively. The velocity of the load follows the reference command accurately; even the system stiffness is relatively loose. In this case, the reference has been a pulse function, the amplitude of which is $0.2\,\mathrm{m\,s^{-1}}$ and the frequency 1 Hz. The small ripple in the response in Fig. 11 is due to a small inaccuracy of the state estimator and linear PI motor control.

5. Conclusions

In the study, a recurrent neural network hybrid controller for a PMLSM is introduced and successfully implemented in the physical linear motor application. The motor velocity is controlled by a conventional PI controller and the vibration of the load is suppressed by using the recurrent neural network compensator. The DE algorithm proved to be an efficient tool for finding initial weights for the recurrent neural network. More traditional teaching methods might have a local minimum problem due to their gradient-based calculation, i.e. they frequently lead to local optimums. DE is not based on the gradient method, and therefore, the local minimum problem is avoided. The problem of DE is its speed in approaching the global minimum, and therefore, it is useful only in offline training. The combination of a neural network hybrid controller and a state estimator reduces the vibration of the load considerably, and the proposed controller is perceived to be stable in all conditions.

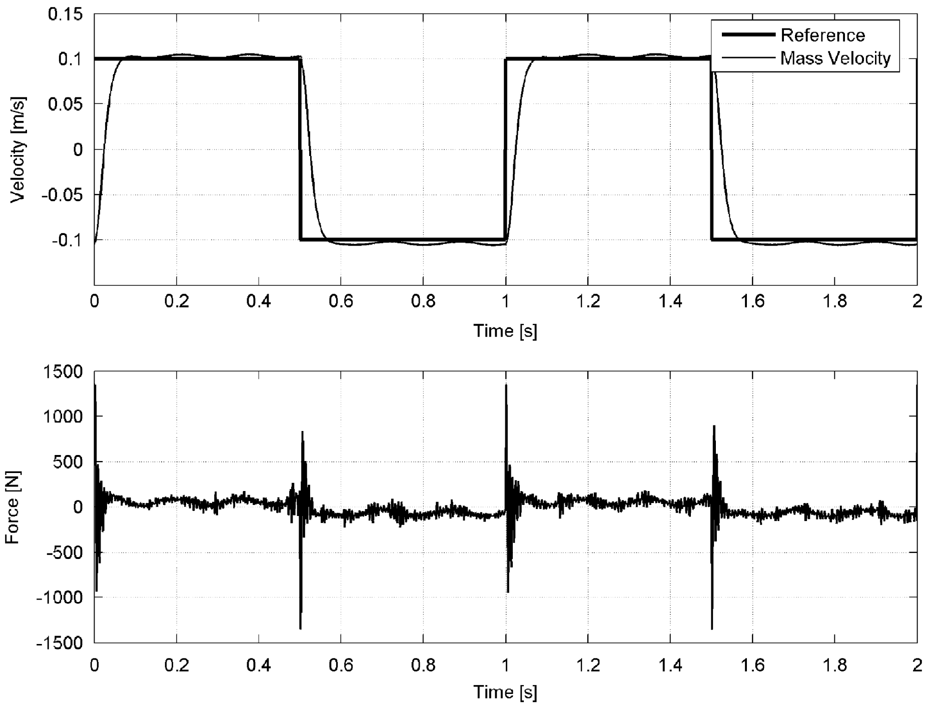

Fig. 11. Measured velocity of the load, and the driving force of the motor in the case of a neural network compensator and under a step excitation.

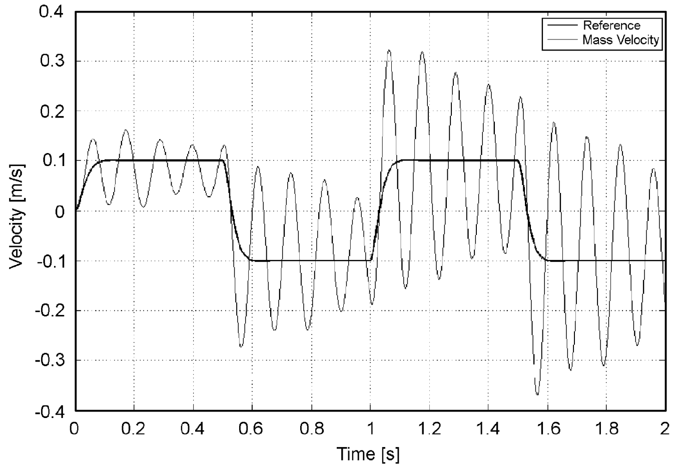

Fig. 12. Simulation of mass load velocity using PI controller without any compensator.

References

Chun, J. S., Jung, H. K., & Jung, J. S. (2000). Analysis of the end-effect of permanent magnet linear synchronous motor energized by partially exited primary. *ICEM, International Conference on Electrical Machines*, *1*, 333–337.

Corne, D., Dorigo, M., & Glover, F. (1999). *New ideas in optimization*. New York: McGraw-Hill.

Dumetz, E., Vanden Hende, F., & Barre, P. J. (2001). Resonant load control method application to high-speed machine tool with linear motor. *Proceedings of the Emerging Technologies and Factory Automation*, *2*, 23–31.

Ellis, G., & Lorenz, R. D. (2000). Resonant load control methods for industrial servo drives. *Proceedings of the IEEE Industry Application Society Annual Meeting, Rome, Italy*, *3*, 1438–1445.

Franklin, G., Powell, J., & Emami-Naeini, A. (1994). *Feedback control of dynamic systems*. USA: Addison-Wesley.

Gieras, J. F., & Piech, Z. J. (2001). *Linear synchronous motors, transportation and automation systems*. Boca Raton, USA: CRC.

Haykin, S. (1994). *Neural networks: A comprehensive foundation* (p. 696). USA: Macmillan College Publishing Company.

Hyun, D. S., Jung, I. S., Shim, J. H., & Yoon, S. B. (1999). Analysis of forces in a short primary type permanent magnet linear synchronous motor. *IEEE Transactions on Energy Conversion*, *14*(4), 1265–1270.

Ilonen, J., Kamarainen, J. K., & Lampinen, J. (2003). Differential evolution training algorithm for feed-forward neural networks. *Neural Proceeding Letters*, *7*(1), 93–105.

Inoue, M., & Sato, K. (2000). An approach to a suitable stator length for minimizing the detent force of permanent magnet linear synchronous motors. *IEEE Transactions on Magnetics*, *36*(4), 1890–1893.

Jung, S. Y., & Jung, H. K. (2002). Reduction of force ripples in permanent magnet linear synchronous motor. In *Proceedings of International Conference on Electrical Machines*, (pp. 452) Brugge, Belgium.

Jun-Keun, J., & Seung-Ki, S. (1995). Kalman filter and LQ-based speed controller for torsional vibration suppression in a 2-mass motor drive system. *IEEE Transactions on Industrial Electronics*, *42*(6), 564–571.

Kang, J. K., & Sul, S. K. (2000). Vertical-vibration control of elevator using estimated car acceleration feedback compensation. *IEEE Transactions on Industrial Electronics*, *47*(1), 91–99.

Lee, Y. M., Kang, J. K., & Sul, S. K. (1999). Acceleration feedback control strategy for improving riding quality of elevator system. *Proceedings of IEEE IAS, Phoenix*, *2*, 1375–1379.

Leleux, D. P., Claps, R., Chen, W., Tittel, F. K., & Harman, T. L. (2001). Application of Kalman filtering to real-time trace gas consentration measurements. *Applied Physics B: Lasers and Optics*, 85–93.

Montanaro, J., & Beale, G. (1998). Feedback control for canceling mechanical vibrations. *Proceedings of the IECON*, *3*, 1433–1438.

Olsson, H., Åström, K. J., de Wit, C. C., Gäfvert, M., & Olsson, P. L. (1998). Friction models and friction compensation. *European Journal of Control*, *4*(3), 176–195.

Otten, G., de Vries, T. J. A., van Amorengen, J., Rankers, A. M., & Gaal, E. W. (1997). Linear motor motion control using a learning feedforward controller. *IEEE/ASME Transactions on Mechatronics*, *2*(3), 179–187.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning internal representations: By error propagation. *Parallel Distributed Processing Explorations in the Microstructures of Cognition*, *1*, 318–362.

Stansfield, E. V. (2001). Introduction to kalman filters. *UK Adaptive Signal Processing Club Discussion Meeting*, pp. 1–7.

Storn, R., & Price, K. (1995). Differential evolution-a simple and efficient adaptive scheme for global optimization over continuous spaces. *Technical Report TR-95-012*, Berkeley, CA.

Wie, B., & Bernstein, D. S. (1990). A benchmark problem for robust control design. In *Proceedings of the American Control Conference*, (pp. 961–962) San Diego, CA, USA.

Zhao, S., & Tan, K. K. (2005). Adaptive feed forward compensation of force ripples in linear motors. *Control Engineering Practice*, *13*(9), 1081–1092.