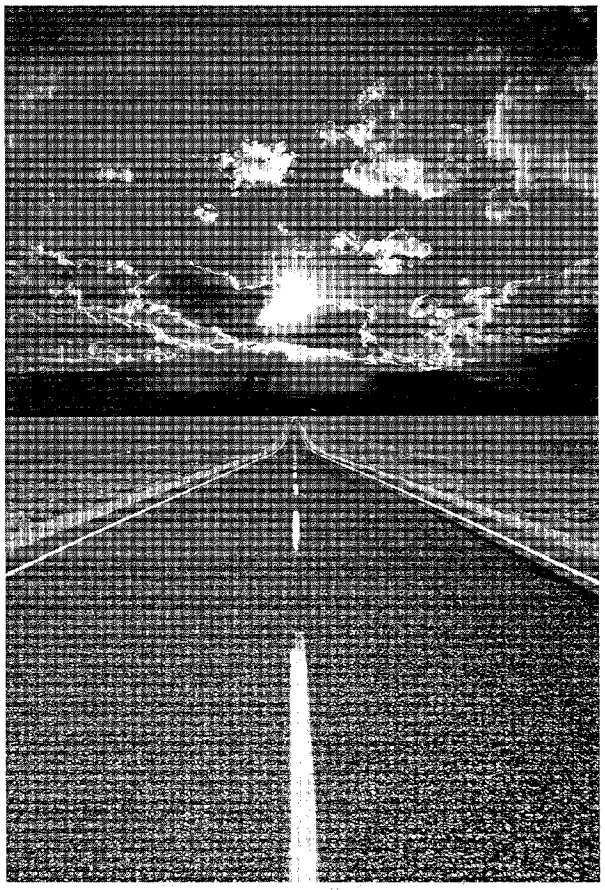
RBF Network Feedforward Compensation of Load Disturbance in Idle Speed Control



©Pete Turner/The Image Bank

Dimitry Gorinevsky and Lee A. Feldkamp

This article is devoted to the problem of idle speed control for an automotive engine. The control objective is to maintain optimal engine speed despite the changing engine load which acts as a disturbance. We study the problem using a phenomenological model of an idling automotive engine known as the Ford model. The system is highly nonlinear and includes delays in the control loop. The problem was previously considered by a number of authors, e.g., [1, 21, 17, 19], and can be considered as somewhat of a benchmark problem for the design of advanced powertrain controllers.

Idle speed control is an example of a disturbance rejection problem that contains several aspects commonly found in real-world control applications. An engine at idle is typically well away from its most favorable region of operation and exhibits significant nonlinearities. Control of such a system is complicated by time delays of both physical (the time between the induction and power strokes) and computational origin. In the model used here, there are two control variables, and these differ in both their range of effectiveness and in their temporal characteristics; spark advance is fast-acting but limited in its effect, while throttle has a large range but a slower effect which results both from the dynamics of filling the intake manifold and from the induction-power delay. The spark variable also has a maximum effective value, i.e., a value beyond which it has an effect opposite to that expected.

These features make the problem very complicated and justify development of nontraditional nonlinear controllers. In [21] a nonlinear fuzzy logic feedback controller was developed for a

This article was originally presented at the IEEE International Conference on Control Applications, Sept. 15-18, 1996, Dearbon, MI.
Gorinevsky is with the Department of Electrical Engineering, University of British Columbia, 2356 Main Mall, Vancouver, B.C., Canada V6T 1W5, email: gorin@ee.ubc.ca; and Measurex Devron Inc., North Vancouver, B.C., Canada V7J 3S4. Feldkamp is with Ford Research Laboratory, Ford Motor Company, MD 1170, Room 3135, SRL, P.O. Box 2053, Dearborn, MI 48121-2053, email: lfeldkam@ford.com.

0272-1708/96/S05.00©1996IEEE

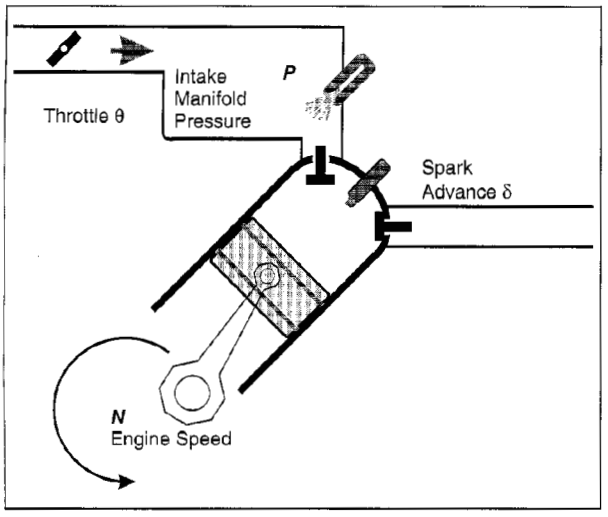


Fig. 1. Schematics of the engine model.

simplified engine model without delays. In [17], a nonlinear high-performance feedback controller based on a recurrent neural network was demonstrated for the model including delays.

In [17], disturbances in the form of changing torque load were regarded as unknown; the controller was required to react to the effect of such disturbances. (Though not discussed, providing disturbance information to the controller would have improved the performance considerably.) Furthermore, the procedure for the neural network training described in [17] is not intended to be employed for in-use controller adaptation; rather it would be used in the development process to produce a fixed-weight controller. In spite of the lack of explicit adaptation, the recurrent networks employed exhibit considerable robustness to plant variations. A method to exploit this tendency was presented in [3].

This article describes a nonlinear *adaptive* feedforward controller for compensation of external load disturbances in the idle speed control of an automotive engine. The controller is based on a Radial Basis Function (RBF) network approximation of certain input-output mappings describing the system. An underlying assumption used in the controller design is that the external engine load is known to the controller. In particular, that might be achieved by putting an appropriate torque sensor in the powertrain or using other available information.

In general, load disturbances of an idling engine, can be categorized in terms of whether, for a particular system design, they are (a) unanticipated, (b) anticipated but uncertain in magnitude or timing, or (c) anticipated and approximately known (or constant) in magnitude and timing. Clearly, a change of system design can convert some disturbances in category (a) to (b) or (c), and some in (b) to (c). Here are examples, taken from our own experimental work on a specific vehicle and reported in [18]:

(a)

- Power window activation
- Cigarette lighter (or other minor electrical load)
- Air conditioning turning off
- Rear window defroster
- Initial power steering activation, i.e., small steering wheel motion.

(b)

- Large power steering activation. A sensor flags steering wheel motion beyond a certain point, but magnitude of load can vary considerably.

(c)

- Air conditioning turning on. A flag from engine control unit anticipates activation of the air conditioning, but some uncertainty in magnitude and timing of the load change exists.
- Shifts between NEUTRAL and DRIVE

In this article we do not distiguish between the above categories of the load disturbance and assume more prior information than is typically provided in current production vehicles. We demonstrate how this information on the external engine load may be used to advantage by a nonlinear feedforward controller.

The proposed controller uses an RBF network for computing the required feedforward control action. The control action (control program) is a sequence of the control inputs needed to ensure a smooth transition of the engine into a new regime as the external load changes. Such a program ensuring optimal quality of the transient processes by necessity depends on the engine state at the time when the transition starts, as well as the new, changed, engine load. The RBF network used in the controller approximates the dependence of the feedforward program on the engine state and the disturbance.

The network has a large number of outputs defining individual steps of the feedforward control sequence. In other words, the nonlinear mapping that is approximated by the RBF network represents a *nonlinear vector field*. It is known that RBF networks are especially well suited for approximation of vector fields compared to other architectures [5, 13].

The proposed controller is based on the optimization and approximation algorithms that are discussed in more detail in the papers [7, 8, 11, 12, 9] in application to a few other control problems. It acts in an open-loop manner between the changes of the engine load disturbance and can be considered as a nonlinear sampled-data (generalized sample-hold) controller. It is necessary to emphasize that the presented simulation is but an example of the proposed approach application. The same approach using neural network computation of the optimal feedforward control program might be applied to other problems of powertrain control.

Problem Statement

Control Problem

Let us consider a Ford dynamic model of an idling automotive engine as studied in [1, 17, 21]. The model is schematically shown in Fig. 1 and it describes a 1.6 litre, 4-cylinder fuel-injected engine.

The system has a very nonlinear dynamics with two states: P, the intake manifold pressure, and N, the engine speed (in rpm). There are two inputs to the system: the throttle angle θ and the spark advance δ.

The model used in this article, for convenience, lumps the idle speed control valve in with the overall engine throttle. In practice, the driver-controlled throttle is usually closed at idle, and air flow is controlled by a bypass air valve. For example, in the test vehicle used in [18], control signal was a duty cycle for the valve.

This control does have substantial authority, easily enough to propel the vehicle at low speed. The dynamics of the valve are benign and are neglected.

The model used in simulation follows the paper [17] and, in contrast to the model considered in [21], includes the delay in the control loop. For the manifold pressure P expressed in kPa and the engine speed N expressed in rpm, the model is given by two differential equations

$$\dot{P} = 42.40(\dot{m}_{ai} - \dot{m}_{ao}),$$
$$\dot{N} = 56.26(T_i - T_L). \tag{1}$$

The right-hand-side expressions in (1) are defined by the input and state variables θ, δ, P, N, and the external disturbance torque d is as follows:

$$\dot{m}_{ai} = \left(1 + 0.907\theta(t-\tau)\right) + 0.09988^2(t-\tau)g(P),$$
$$g(P) = \begin{cases} 1 & \text{for } P < 50.6625 \\ 0.0197\left(101.325P - P^2\right)^{1/2} & \text{for } P \geq 50.6625 \end{cases}$$
$$\dot{m}_{ao} = -0.5968 \cdot 10^{-3} N - 0.1336P + 0.5341 \cdot 10^{-3} NP - 1.757 \cdot 10^{-6} NP^2,$$
$$m_{ao} = \dot{m}_{ao} / (120N),$$
$$T_i = -39.22 + 3.25024 \cdot 10^5 m_{ao} - 0.0112\delta^2(t-\tau_\delta) + 0.635\delta(t-\tau_\delta),$$
$$+\left(0.0216 + 0.675 \cdot 10^{-3}\delta(t-\tau_\delta)\right)N(2\pi/60) - 0.102 \cdot 10^{-3} N^2 (2\pi/60)^2,$$
$$T_L = (N/263.17)^2 + d,$$
$$\tau_\delta = \tau + 15/N, \tag{2}$$

where the throttle angle θ and spark advance δ are in degrees, with ranges 5 to 25 and 10 to 45 respectively; the external engine load torque disturbance d is in Nm with the range of 0 to 61. The expressions (2) were obtained phenomenologically by fitting the experimental data. The variables $\dot{m}_{ai}$ and $\dot{m}_{ao}$ refer to the mass air flows into and out of the manifold, m_{ao} is the air mass in the cylinder, T_i is the torque developed by the engine, and T_L is the load torque.

Dependencies of the variables on the time t are not shown explicitly in (2), except for the control inputs θ and δ that are delayed. The delay of the digital controller is τ = 20 ms, and τ_δ is a deadtime (in sec) incorporating the controller delay τ and a quarter engine revolution delay needed for the throttle change to take effect on the cylinder pressure. More discussion of the engine dynamics model (1), (2) can be found in the paper [17].

The external load d of the engine acts as a disturbance. Herein, it is assumed that the load is constant most of the time, but may change values step-wise at some instants. It is further assumed that the load change is immediately known to the controller. The control goal is to ensure that once the load changes, the control inputs are changed in such a way that after a short transient process the engine settles in a regime with the desired speed N = 750 rpm. The transient processes should be smooth, fast, and not exhibit excessive overshoot. Stabilization of the manifold pressure P is not required.

The layout of the control system is shown in Fig. 2. The feedforward controller we design is essentially a sampled-data

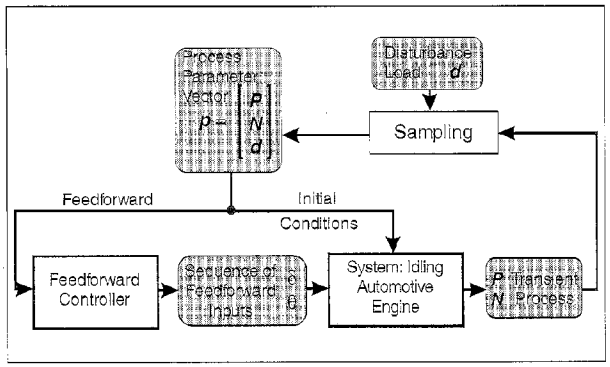


Fig. 2. Control problem arrangement.

controller with asynchronous sampling that is activated at the instant of the load change. The transient process occurring after the engine load change, depends both on the new load d and on the initial conditions P and N at the time of the load change. We collect the data d, P, and N into the information vector p,

$$p = [P\ N\ d]^T \tag{3}$$

As shown in Fig. 2, the vector p is input to the controller. The controller output is a sequence of the control inputs θ and δ. The output sequence should be chosen to compensate for the effect of the disturbance and provide for good quality of the transient process, such as little oveshoot for a moderate control effort. A more formal input/output problem statement and performance index is are formulated in the next subsection.

A feedforward that ensures the desired transient processes is sought as a zero-order sample hold function (see Fig. 3). The feedforward control is initiated at time t_d of the load disturbance change. After changing a few values, the control inputs are kept constant till the next change of the engine load occurs. We consider the timing of the control input change—i.e., durations of the zero-order holds of the control values—to be predefined. The schedule of changing the feedforward control values could be chosen subject to the achievable sampling period of the digital controller, information processing and cylinder firing delays, and transient process optimization requirements. With this schedule given, for each transient process the feedforward control inputs are defined as functions of time by a sequence of their sampled values θ_j and δ_j. Let us collect these sampled values of the control in a vector denoted by U,

$$U = \left[(\theta_1 - \theta_*)\ldots(\theta_{n_u} - \theta_*)(\delta_1 - \delta_*)\ldots(\delta_{n_u} - \delta_*)\right]^T \in \Re^{N_U}, \tag{4}$$

where $N_U = 2n_u$, and θ_* and δ_* are nominal values on the throttle and spark advance, respectively. It is assumed that the last values in the control sequence, θ_{n_u} and δ_{n_u}, are applied at time $t_d + T_c$ and these values of the throttle and spark advance are maintained till the time of the next load disturbance change. The duration T_c of the feedforward control transient can be determined empirically to be about the desired duration of the transient process.

After each application of disturbance (change of the engine load), the transient process in the system can be described

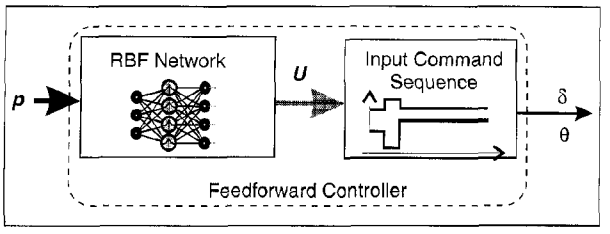


Fig. 3. Conceptual design of the controller.

through a vector Y of the sampled deviation of the system variables from the steady state,

$$Y = \left[\Delta P_1 \ldots \Delta P_{n_y} \Delta N_1 \ldots \Delta N_{n_y}\right]^T \in \Re^{N_Y},$$
$$\Delta P_j = P(t_d + j\tau_y) - P_*, \quad \Delta N_j = N(t_d + j\tau_y) - N_*, \qquad (5)$$

where $N_Y = 2n_y$, n_y is the number of input samples, τ_y is the output sampling period, t_d is time of the load change and beginning of the transient process, P_* = 35 kPa, and N_* = 750 rpm are the nominal values of the engine state variables. We further call Y the *output vector*.

Let us denote by T_f the duration of the output sampling interval in (5), $T_f = (n_y + 1)\tau$, and $T_f > T_c$. When designing the sampled-data feedforward controller, we assume that the interval between the changes of the load disturbance d is always greater than T_f.

In addition to the feedforward control sequence defined by the vector U, the transient process described by the output vector Y depends on the system state variables P and N at the time of the load change, as well as on the new, changed value of the load d. We put these values together into the *information* vector $p = [P\ N\ d]^T$. For the output vector, we can write

$$Y = f(U, p); \quad Y \in \Re^{N_Y}, \quad U \in \Re^{N_U}, \quad p \in \mathcal{P} \subset \Re^{N_p}, \qquad (6)$$

where $f(\cdot, \cdot)$ is a smooth mapping, since the right-hand sides of the system dynamics equations are smooth. In (6) we assume that the information vector $p = [P\ N\ d]^T$ belongs to a compact domain $\mathcal{P}$. This domain $\mathcal{P}$ could be defined from the practical bounds on the system variables P, N, and d.

Optimization Problem

Ideally, the feedforward control should provide for small deviations in the transient process, thus, U should be chosen so that components of the vector Y are small. This should be achieved by means of possibly small control effort. Thus, the control objective could be described as minimization of a quadratic performance index of the form

$$J(Y, U) = Y^T Q\, Y + U^T R\, U, \qquad (7)$$

where $Q \in \Re^{N_Y, N_Y}$ and $R \in \Re^{N_U, N_U}$ are positive definite weighting matrices.

For a fixed p, the problem (6), (7) is a classical nonlinear least square minimization problem. Such problems are typically solved using iterative minimization methods, for instance, the Levenberg-Marquardt method. An optimal vector $U = U_*$ can be found as a result of this minimization. The minimization could be performed numerically by using a computer model of the engine embedded in the mapping (6). In this article, however, we choose a different approach and perform the optimization on-line, with the system in the iteration loop. In that case, each evaluation of the function (6) corresponds to an input/output experiment with the system.

In the simplest case, performing the minimization on-line, with the system in the loop, can be explained as follows. At each minimization step, initial conditions of the system P and N should be reproduced and the same disturbance d should be applied. At each minimization step, a feedforward control sequence U (4) would be applied to the system and the transient process described by the vector Y (5) obtained. A version of the standard Levenberg-Marquardt algorithm applicable to such on-line optimization is described later.

Unfortunately, the above outlined on-line optimization approach would not completely solve the stated practical control problem for two reasons. First, the optimal input U_* will be different for each value of the information vector p. Second, it would not be practical to demand that several mimimization iterations would be done with reproduced initial conditions and disturbance (that is, p). In doing the optimization on-line we have to count on the vector p changing from step to step of the minimization in a largely uncontrollable way. For the idle speed control problem stated above, the last component of the information vector is a new disturbance value, while the first two (P and N) are the steady-state values defined by the preceding disturbance load. The changing information vector p effectively plays a role of the disturbance in the control (optimization) process defined by (6), (7).

The control problem we consider in this article is to find an entire *mapping* $U_*(p)$, i.e., the entire dependence of the optimization problem solution U_* on p, rather than a single value of U_* for given p. The mapping $U_*(p)$ will then define a control law for the disturbance compensation. This control law should be used on-line in real time; therefore we are looking for a computationally efficient way for finding $U_*(p)$.

Controller Architecture

An efficient way to compute $U_*(p)$ is to use a neural network or other approximation technique, which will define the mapping $U_*(p)$ through a limited number of approximation parameters (network weights). In this article, we use a Radial Basis Function (RBF) network for the purpose of such approximation.

The controller we design is illustrated in Fig. 3. The controller approximates the nonlinear vector-valued mapping $U(p)$ with help of an RBF network. Thus, the controller design consists in choosing the matrix $\overline{U}$ of the RBF network weights to achieve an optimal quality of the transient processes. Following the neural network terminology, we can call the optimization process of tuning the RBF network weights the training process. If this training is performed on-line, the controller is an adaptive one.

For the system in question, the information vector p is changing from step to step of the process in search of the optimal approximation for U_*. We obtain an approximation of $U_*(p)$ as a result of an on-line iterative *nonlinear least square parametric optimization* procedure proposed in [8]. This procedure can be considered as an adaptive controller for compensation of the

disturbance p in the system (6). The controller is an adaptive one, as it assumes the system (6) to be initially unknown and estimates certain optimization gains in the course of the iterations. The procedure of [8] can be considered as an extension of the Levenberg-Marquardt method and it updates the weights of the RBF network approximation for $U_*(p)$ as shown in Fig. 4. At each step of the parametric optimization process, the controller computes a vector U given the vector p. Then, the feedforward defined by U is applied to the system and, based on the obtained vector Y, the controller updates the available approximation of $U_*(p)$. The parametric optimization procedure of [8] is explained in the next section for completeness.

On-Line Parametric Nonlinear Least Square Optimization

This section formulates the algorithm for on-line parametric nonlinear least square optimization, which we use to solve the formulated problem. This optimization algorithm, formulated and studied analytically in [8], can be considered (and applied) as an adaptive algorithm for nonlinear system control. The derivation below uses a somewhat more general notation than [8]. This derivation mostly follows [8] and is included for completeness.

In the beginning of this section we briefly discuss a classical Levenberg-Marquardt algorithm which could be applicable to on-line optimization of the control input in the absence of the dependence on the information vector p in (6). This algorithm is a basis for the main algorithm applied in this article which is introduced at the end of this section.

On-Line Optimization of Input U for Fixed p

Let us first consider the information vector p to be fixed. In that case, U_* can be found as a solution of a standard nonlinear least square minimization problem (7), where $Y = f(U)$ (we have removed the second argument in (6)). This problem is well known and studied and a few iterative algorithms exist for finding its solution numerically. Let us briefly consider a modification of the Levenberg-Marquardt algorithm for nonlinear least square optimization, which has very well recommended itself for solution of such problems. This algorithm is a basis for the parametric optimization algorithm of [8] that we apply in this article and understanding it is necessary to understand our main algorithm.

Let us first assume that the gradient matrix $G = \dfrac{\partial f}{\partial U}$ of the mapping $Y = f(U)$ can be obtained for a given argument U.

Let $U^{(n)}$ and $Y^{(n)} = f(U^{(n)})$ be input and output vector obtained at iteration n. The Levenberg-Marquardt method iterates the minimizing input guess as (see [2])

$$U^{(n-1)} = U^{(n)} - \left(R + I_{N_u}\mu_n + G^{(n)^T} Q G^{(n)} \right)^{-1} \left(R U^{(n)} + G^{(n)^T} Q Y^{(n)} \right), \tag{8}$$

where $\mu_n \geq 0$ is a step length parameter, I_{N_U} is a $N_U \times N_U$ unity matrix, Q and R result from (7), and $G^{(n)} = G(U^{(n)})$.

The motivation for the update (8) is as follows. Let us consider an affine model of the mapping $Y = f(U)$ of the form

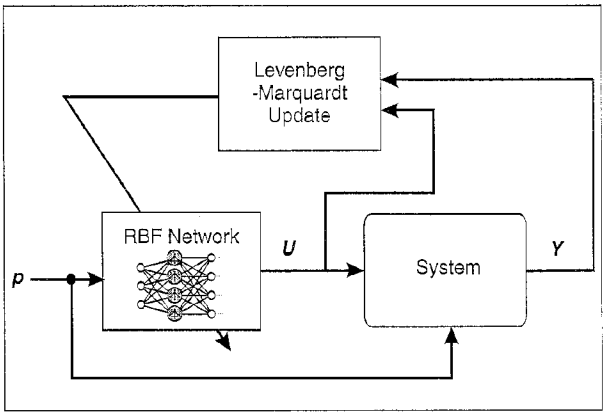


Fig. 4. On-line training of the network (adaptive control).

$$\hat{Y} = Y^{(n)} + G^{(n)}\left(U - U^{(n)}\right); \tag{9}$$

Let us also demand that the minimization step length does not exceed a value $l_n > 0$, so that

$$U^{(n+1)} = U^{(n)} + s^{(n)} \; ; \; \| s^{(n)} \| \leq l_n \tag{10}$$

By solving (7) and (10) with respect to $U^{(n+1)}$ and using the Lagrange multiplier method to satisfy constraint (10) we come to (8). The Lagrange multiplier μ_n is nonnegative and can be computed once G and allowed step length l_n are given. Clearly, dependence of μ_n on l_n is monotone nonincreasing. In practice, instead of computing μ_n from l_n, usually μ_n itself is made a parameter of choice.

For the problem in hand, the gradient G is not available and we can only estimate values of the function $Y = f(U)$ itself. In that case, a natural approach is to employ a secant estimation of the gradient. To this end, one can consider an affine model (9) of the mapping $Y = f(U)$ and try to update estimates of parameters of this model from the available input/output measurements.

The most commonly used method for estimating the gradient is *Broyden secant update*. Let $G^{(n)}$ be an estimate of the gradient at the step n. Denote by $s^{(n)}$ the variation of input, and by $w^{(n)}$ the corresponding variation of the output at the previous minimization step. For a small step length $\| s^{(n)} \|$, the gradient should satisfy

$$G s^{(n)} = w^{(n)} \, , \; s^{(n)} = U^{(n)} - U^{(n-1)}, \; w^{(n)} = Y^{(n)} - Y^{(n-1)} \tag{11}$$

The Broyden update rule can be considered as an application of the projection estimation algorithm [6], which is very popular in adaptive control and signal processing applications. The Broyden update is used in conjunction with the input update (8) and has the form

$$\hat{G}^{(k+1)} = \hat{G}^{(k)} + \left(w^{(k)} - \hat{G}^{(k)} s^{(k)} \right) s^{(k)^T} \Big/ \left(c^2 + \left\| s^{(k)} \right\|^2 \right), \tag{12}$$

where $c > 0$ is a scalar parameter used to avoid division by zero.

Approximating Parametric Optimum

Under reasonable assumptions (studied in [8]), the mapping $U_*(p)$ is smooth. Therefore, this mapping could be approximated with the help of an expansion with respect to certain basis functions $\{h_j(p) : \mathcal{P} \mapsto \Re\}$ as

$$U_*(p) = \sum_{j=1}^{N_a} U_{*j} h_j(p) + \delta U(p), \quad \|\delta U(p)\| \le \delta_U, \tag{13}$$

where $U_{*j} \in \Re^{N_U}$ are the expansion weights, $p \in \mathcal{P}$, and $\delta U(p) \in \Re^{N_U}$ is the approximation error. For $U_*(\cdot)$ with a uniformly bounded derivative on $\mathcal{P}$, the approximation error bound δ_U can be made arbitrarily small provided the expansion order N_a is sufficiently large. For a polynomial expansion, this is ensured by a multidimensional version of the Weierstrass theorem. In this article, we use another type of expansion, for which this is valid: a radial basis function expansion.

Let us consider expansion (13) with the basis functions $h_j(p)$ of the form $h_j(p) = h(p\text{-}Q(j))$, where the function $h(r)$ depends on the radius $\| r \|$, and $Q^{(j)} \in \Re^{N_P}$ are given vectors. Such expansions are known under the name of RBF networks. One can find further details and references in [4, 14, 15, 16]. Most commonly used radial functions are Gaussian, $h(p) = \exp\left(-\|p\|^2 / r_0^2\right)$, and reverse Hardy multiquadrics, $h(p) = \left(1 + \|p\|^2 / r_0^2\right)^{-1/2}$. For these functions, the expansion (13) is local, i.e., the radial functions decay away from their centers. This is very advantageous for many applications. Parameter r_0 in these functions has a meaning of the function width and is chosen to be of the order of the mean distance between the RBF centers in the domain $\mathcal{P}$. Usually, the radial function width parameter r_0 is chosen to be about an average distance between the neighboring node centers.

To find the parametric optimum (13), we neglect the expansion residual δU in (13). In that case, the parametric optimum is defined by a set of the RBF network weights U_{*j}, which we further collect in a single weight matrix of the form

$$\overline{U}_* = \left[U_{*1} \ldots U_{*N_a}\right] \in \Re^{N_U, N_a} \tag{14}$$

There are a few advantages in using RBF network approximation in the described problem, compared to other parametric approximation schemes, in particular, a multilayered sigmoidal neural network. First, an RBF network with fixed centers is linear in its weights, which makes an on-line training of such network a special case of classical adaptive control with linearly parametrized nonlinearities. Second, for the vector-valued output, the number of the nonlinear computations in the RBF network approximation is independent of the output dimension [5, 13]. Thus, RBF networks are especially convenient for approximation of the vector fields, such as $U(p)$.

Affine RBF Network Model of the System Mapping

The on-line parametric optimization algorithm of [8] we derive can be considered as an extension of the Levenberg-Marquardt algorithm. Similarly to a standard derivation of the Levenberg-Marquardt algorithm given earlier, our derivation here will be based on an affine model (9) of the mapping (6). Such affine model can be written in the form

$$\hat{Y} = \hat{G}(p)U + \hat{Z}(p), \tag{15}$$

where $\hat{G}(p)$ and $\hat{Z}(p)$ are (smooth) matrix and vector functions of p. For a fixed information vector p, the model (15) is affine in U, at the same time the model depends on p in a nonlinear way.

Let us introduce functions:

$$Y_*(p) = f\left(U_*(p), p\right), \quad G_*(p) = \left.\frac{\partial f(U,p)}{\partial U}\right|_{U=U_*(p)},$$
$$Z_*(p) = Y_*(p) - G_*(p)U_*(p) \tag{16}$$

Similarly to (13), let us use RBF networks of the same form for approximation of the functions $Z_*(p)$, and $G_*(p)$

$$Z_*(p) = \sum_{j=1}^{N_a} Z_{*j} h_j(p) + \delta Z(p), \quad \|\delta Z(p)\| \le \delta_Z, \quad p \in \mathcal{P}, \tag{17}$$

$$G_*(p) = \sum_{j=1}^{N_a} G_{*j} h_j(p) + \delta G(p), \quad \|\delta G(p)\| \le \delta_G, \quad p \in \mathcal{P}, \tag{18}$$

where $Z_{*j} \in \Re^{N_Y}$ and $G_{*j} \in \Re^{N_Y, N_U}$ are the expansion weights. The functions $Z_*(\cdot)$ and $G_*(\cdot)$ have derivatives, which are uniformly bounded on $\mathcal{P}$ provided that the derivatives of $U_*(\cdot)$ are bounded. Hence, δ_Z and δ_G can be made small for high-order N_a of the expansions (17), (18).

We are now in position to explain the basic algorithm of [8] for the parametric NLS (Nonlinear Least Square) minimization, which is the main algorithm applied in this article. When deriving the algorithm, we assume that the approximation errors δ_U, δ_Z, and δ_G are zero. These errors are taken into account in the algorithm convergence analysis in [8]. In the absence of the approximation error, (16) can be represented in the linear regression form

$$U_*(p) = \overline{U}_* \Phi(p), \quad \overline{U}_* = \left[U_{*1} \ldots U_{*N_a}\right] \in \Re^{N_U, N_a}, \tag{19}$$

$$\Phi(p) = \left[h_1(p) \ldots h_{N_a}(p)\right]^T \in \Re^{N_a} \tag{20}$$

Similarly to (20), we can present (17) and (18) in the form of regressions linear in the weights Z_{*j} and G_{*j}. With these regression representations (17) and (18) in mind, the model of the form (15) can be represented as the following regression:

$$\hat{Y} = \Theta \overline{\Phi}(p, U), \quad \Theta \in \Re^{N_Y, N_a (N_U + 1)}, \quad \overline{\Phi}(p, U) \in \Re^{N_a (N_U + 1)}, \tag{21}$$

$$\overline{\Phi}(p, U) = \Phi(p) \otimes W, \quad W = \left[c \;\; U^T\right]^T, \tag{22}$$

where $\otimes$ denotes the Kronecker (direct) product of matrices and $c > 0$ is a scalar scaling parameter. For a fixed parameter p, the model (21) has the form (15). By substituting Θ into (21), one obtains an affine model of the form (15), where

$$\hat{Z}(p) = \sum_{j=1}^{N_a} Z_j h_j(p), \quad \hat{G}(p) = \sum_{j=1}^{N_a} G_j h_j(p) \tag{23}$$

We assume that for $\Theta = \Theta_*$, the affine model (21) gives exactly the linearization of the mapping (6) in the optimum (17), (18). In other words,

$$\Theta_* = \left[Z_{*1}/c \;\; G_{*1} \ldots Z_{*N_a}/c \;\; G_{*N_a} \right] \tag{24}$$

Basic Algorithm

The parametric NLS optimization algorithm of [8] iteratively updates a guess of the vector $\overline{U}_*$. This algorithm is a generalization of the classical Levenberg-Marquardt algorithm that is widely used for a standard (not parametric) NLS minimization.

Our goal is to build an approximation of the form (19) to the optimal input mapping $U_*(p)$. We assume that a sequence of the information vectors $\left\{p^{(k)}\right\}_{k=1}^{\infty}$ is given. Let $\overline{U}^{(k)}$ be the value of the input parameter matrix in (19) at the step k. Then, in accordance with (19), the input vector at step k is $U^{(k)} = \overline{U}^{(k)}\Phi\left(p^{(k)}\right)$. Let us denote by

$$U^{(k+1|k)} = \overline{U}^{(k)}\Phi\left(p^{(k+1)}\right) \tag{25}$$

the input U, which would be obtained at step $k+1$ if the matrix $\overline{U}^{(k)}$ is not updated. Let us denote the output vector at step k by $Y^{(k)} = f(U^{(k)}, p^{(k)})$.

Let us demand that, similarly to the standard Levenberg-Marquardt method, the minimization step should be bounded as

$$U^{(k+1)} = U^{(k+1|k)} + s^{(k)}, \; \| s^{(k)} \| \le l_k, \tag{26}$$

The output of the model (21) at the step $k+1$ is

$$\begin{aligned} \hat{Y}^{(k+1)} &= \Theta\overline{\Phi}\left(p^{(k+1)}, U^{(k+1)}\right) \\ &= \Theta\overline{\Phi}\left(p^{(k+1)}, U^{(k+1|k)}\right) + \Theta\left(\Phi\left(p^{(k+1)}\right) \otimes W_s^{(k)}\right) \\ &\equiv \hat{Y}^{(k+1|k)} + \hat{G}\left(P(k+1)\right)s^{(k)}, \end{aligned} \tag{27}$$

where $W_s^{(k)} = \begin{bmatrix} 0 & s^{(k)^T} \end{bmatrix}^T$ and $\Theta\overline{\Phi}\left(p^{(k+1)}, U^{(k+1|k)}\right)$.

By substituting $\hat{Y}^{(k+1)}$ in (27) as Y and $U^{(k+1)}$ in (25) as U into (7), and optimizing the minimization step $s^{(k)}$ subject to the constraint $\| s^{(k)} \| \le l_k$, we obtain

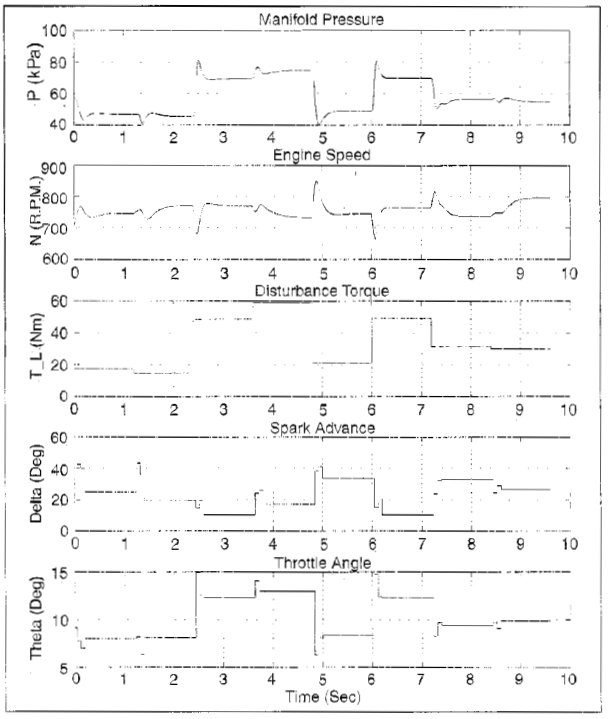


Fig. 5. System response to the disturbance with the trained feedforward controller: simulation results.

$$s^{(k)} = -\left(I_{N_U}\mu_k + \hat{D}^{(k+1)}\right)^{-1}\left(\hat{G}\left(p^{(k-1)}\right)^T Q\hat{Y}^{(k+1|k)} + RU^{(k+1|k)}\right) \tag{28}$$

$$\hat{D}^{(k+1)} = R + \hat{G}\left(p^{(k+1)}\right)^T Q\hat{G}\left(p^{(k+1)}\right) \tag{29}$$

where μ_k is a Lagrange multiplier that is introduced to comply with the step boundedness condition $\| s_k \| \le l_k$. As for the classical Levenberg-Marquardt method explained earlier, the dependence of μ_k on l_k is monotone nonincreasing and instead of first empirically choosing l_k and then computing the Lagrange multiplier, it is advisable to make μ_k itself a parameter of choice.

Note that according to (19) and (26)

$$s^{(k)} = \left(\overline{U}^{(k-1)} - \overline{U}^{(k)}\right)\Phi\left(p^{(k+1)}\right) \tag{30}$$

By finding a least square solution to (30) for the $\overline{U}^{(k-1)} - \overline{U}^{(k)}$ and substituting (28) for $s(k)$, we obtain a step of the proposed basic parametric NLS optimization method:

$$\overline{U}^{(k+1)} = \overline{U}^{(k)} - \left(I_{N_U}\mu_k + \hat{D}^{(k-1)}\right)^{-1}\left(\hat{G}\left(p^{(k+1)}\right)^T Q\hat{Y}^{(k+1|k)} + RU^{(k+1|k)}\right)\frac{\Phi\left(p^{(k+1)}\right)^T}{\left\|\Phi\left(p^{(k+1)}\right)\right\|^2} \tag{31}$$

where $\hat{G}(p)$ is defined by (23); $\hat{D}^{(k)}$, by (29); $\hat{U}^{(k+1|k)}$, by (25), and (20); and $\hat{Y}^{(k+1|k)}$ is defined in accordance with (21) as $\hat{Y}^{(k+1|k)} = \Theta\overline{\Phi}\left(p^{(k)}, U^{(k+1|k)}\right)$.

The affine model (15) can be written in the regression form (27). Therefore, at each step of our parametric NLS algorithm, we can apply the projection update to estimation of the parameter matrix Θ. This update has the form

$$\hat{\Theta}^{(k+1)} = \hat{\Theta}^{(k)} + a^{(k)}\left(Y^{(k)} - \hat{\Theta}^{(k)}\overline{\Phi}^{(k)}\right)\overline{\Phi}^{(k)T} \Big/ \left\|\overline{\Phi}^{(k)}\right\|^2, \quad (32)$$

where $\overline{\Phi}^{(k)} = \overline{\Phi}\left(p^{(k)}, U^{(k)}\right)$ and could be considered as a generalization of the Broyden update (12). In (32), $a^{(k)}$ is a scalar deadzone parameter that is introduced in the usual way to compensate for the influence of the mismodeling error. The deadzone parameter $a^{(k)}$ is zero if the prediction error $Y^{(k)} - \hat{\Theta}^{(k)}\overline{\Phi}^{(k)}$ is within the mismodeling bounds, and unity otherwise.

The choice of the deadzone parameter $a^{(k)}$ in (32), as well as a proof of the algorithm convergence, are considered in [8]. To ensure convergence of the estimation algorithm, a small self-excitation signal is added to the computed control $U^{(k)}$ before it is applied to the system. This self-excitation is needed to make the regressor vector sequence in (32) persistently exciting as discussed in [7, 8].

Algorithm Convergence

Equations (20), (22), (31), and (32) constititute the basic algorithm for on-line parametric NLS optimization first proposed in [8]. An analysis of the algorithm convergence is presented in [8]. Since the proposed adaptive algorithm is, essentially, a nonlinear optimization algorithm, it is only possible to prove *local* convergence of the algorithm. The locality condition here means that the initial approximation to the nonlinear feedforward control mapping (19), (20), should be sufficiently close to the optimum. The domain of the algorithm convergence theoretically ensured in [8] depends on the degree of the system nonlinearity (second derivative bound).

The local convergence result of [8] demonstrates that the advocated algorithm is not self-contradictory. However, its practical usefulness depends on its performance in the particular application and is studied in the next section by means of simulation.

Simulation Results

In simulation, both feedforward controller outputs (throttle and spark advance) in each transient process are assumed to be a sequence of three steps. At each step, the control outputs are constant. The first two steps have duration of 80 ms and the third value of the control variables is maintained till the next transient starts. Thus, the dimension of the RBF network output vector U is six. To obtain the vector Y in (5), variables P and N were sampled $n_y = 25$ times each with an interval of 40 ms, starting 70 ms after the change of the disturbance. Thus, the dimension of the vector Y is 50.

For training of the RBF network, the extended Levenberg-Marquardt algorithm for on-line parametric optimization described in the previous section is used. The algorithm minimizes a mean square deviation of the engine speed N from its desired steady state value of 750 rpm in the transient process. The minimization (RBF network training) is performed while the engine load undergoes step-wise changes, i.e., during the system operation as shown in Fig. 4. In Fig. 4, the information vector p at each step is defined by the system steady state at the previous step, which can be monitored through the components of the output vector Y at the previous step. Thus, the closed-loop system has two feedback loops: the sampled feedback loop shown in Fig. 2, and the adaptation feedback loop for tuning the RBF network weights as shown in Fig. 4. The algorithm used can be considered as a nonlinear adaptive algorithm of the feedforward control.

For the simulation, the performance index (7) was chosen so that

$$Y^T QY = \sum_{j=1}^{n_y}\left(\Delta N_j\right)^2$$

$$U^T RU = 10^{-7}\sum_{j=1}^{n_y}\delta_j^2 + 2\cdot 10^{-4}\sum_{j=1}^{n_y}\left(\delta_j - \delta_{j-1}\right)^2 + 6\cdot 10^{-7}\sum_{j=1}^{n_y}\theta_j^2$$

where $\delta_0 = 0$. This choice of the performance index closely approximates one made in [17] and was empirically found to provide good quality of the transient processes. (See [17] for more discussion.)

The simulations use an RBF network with 45 nodes allocated on a regular grid in the network input space (information vector p) with five gradations in the disturbance d and three gradations in each P and N. The range of the input variables covered by the grid was $0 \le d \le 60$ Nm for the disturbance load; 22.5 kPa $\le$ P $\le$ 97.5 kPa, for the manifold pressure; and 500 rpm $\le$ N $\le$ 2,000 rpm for the engine speed. After 3,000 of the simulated transient processes a good quality feedforward compensation was achieved.

Though typical load disturbance in idle speed engine control are 10-20 Nm, the wider load range assumed in our simulation is meant to represent combinations of individual loads, e.g., simultaneous air conditioner compressor engagement and power steering lockup, or air conditioner engagement plus a NEUTRAL to DRIVE shift. Experience shows that in vehicle studies such combination loads are fully capable of stalling a poorly controlled engine.

The simulation results for a few transient processes with a trained RBF network are shown in Fig. 5. The transient process quality is quite good despite the large amplitude of the disturbance load change. Maximal transient process amplitude does not exceed 100 rpm and the maximal steady state error does not exceed 20 rpm. This quality is achieved with an RBF network of a relatively small size and can be further improved by increasing the size of the network to make the approximation of the mapping $U(p)$ more accurate. For comparison, for the purely feedback controller designed in [17], the maximal transient process error exceeds 300 rpm. One should, however, understand that unlike [17], the algorithm proposed in this article uses information on the disturbances.

High quality of the transient processes was obtained despite the 40 ms delay in the control loop. Such a delay normally makes design of feedback controller for the problem in hand much more

difficult. At the same time, for the proposed approach, the delay does not make the controller performance much worse, nor the controller design more complicated. One reason for this is use of a feedforward controller; hence, the delay does not influence feedback loop stability. Another reason of the high efficiency is that the input dimension of the RBF network computing the feedforward control is that of the information vector depends only on the number of the variables that define a steady state of the system, which is much smaller than that of the state vector for the system with delays.

Conclusions

The problem of feedforward compensation of load disturbances in idle speed control of automotive engine by using an adaptive RBF network approximation has been studied. The conclusions learned from this study are as follows.

- The completed simulations shows that the proposed approach works well in the idle speed control problem.
- The considered results are obtained with a purely *feedforward* control scheme. In practice, this scheme should be used together with an appropriate feedback controller, e.g., one considered in [17]. Such a combination will both provide for the compensation of unknown disturbances and improve transient process quality compared to purely feedback control.
- For application of the approach, the engine load disturbance should be known. This can be achieved either by the direct measurement of the load, or by the use of an appropriate nonlinear state observer.
- The proposed approach uses a relatively simple and accurate neural network architecture, which can be trained on-line, adaptively. Furthermore, the complexity of the proposed neural network controller should not be influenced by the introduction of the delays in the system model.
- In addition to the idle speed control problem, the considered approach might be also useful in other powertrain control problems where compensation of known disturbances is required. For instance, it might be used in compensation of the engine load change caused by the gear shift.
- The proposed approach could be extended for the case of load disturbances which differ from constant on a certain interval. Such extensions are currently being studied.

References

[1] J.A. Cook and B.K. Powell, "Modeling of an internal combustion engine for control analysis," *IEEE Control Systems Magazine*, vol. 13, no. 3, pp. 62-68, 1993.

[2] J.E. Dennis and R.B. Schnabel, *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*, Prentice Hall, Englewood Cliffs, 1983.

[3] L.A. Feldkamp and G.V. Puskorius, "Training of Robust Neurocontrollers," *33rd IEEE CDC*, Lake Buena Vista, FL, December 1994.

[4] R. Franke, "Scattered Data Interpolation: Test of Some Methods," *Math. of Computation*, vol. 38, no.157, pp.181-200, 1982.

[5] F. Girosi, M. Jones, and T. Poggio, *Priors, Stabilizers, and Basis Functions: From Regularization to Radial, Tensor and Additive Splines*, AI Memo No. 1430 (C.B.C.L. Paper No. 75), Artificial Intelligence Laboratory, MIT, June 1993.

[6] G.C. Goodwin and K.S. Sin, *Adaptive Filtering, Prediction and Control*. Prentice-Hall, Englewood Cliffs, NJ, 1984.

[7] D.M. Gorinevsky, "On the Persistency of Excitation in Radial Basis Function Network Identification of Nonlinear Systems," *IEEE Tr. on Neural Networks*, vol. 6, no. 5, pp. 1237—1244, 1995.

[8] D.M. Gorinevsky, "An Algorithm for On-Line Parametric Nonlinear Least Square Optimization," *33rd IEEE CDC*, Lake Buena Vista, FL, December 1994.

[9] D. Gorinevsky and G. Vukovich, "Control of Flexible Spacecraft Using Nonlinear Approximation of Input Shape Dependence on Reorientation Maneuver Parameters," *13th World Congress of IFAC*, San Francisco, CA, June 1996.

[10] D.M. Gorinevsky, "An Application of On-Line Parametric Optimization to Task-Level Learning Control," *American Control Conf.*, Seattle, WA, June 1995.

[11] D.M. Gorinevsky, "Sampled-Data Indirect Adaptive Control of Bioreactor Using Affine Radial Basis Function Network Architecture," *Tr. ASME. J. Dynam. Syst. Meas. and Control* (to appear).

[12] D.M. Gorinevsky, "Adaptive Learning of Feedforward Control Using Radial Basis Function Network Approximation," *1993 IEEE Intern. Symp. on Intelligent Control*, Chicago, IL, pp. 762-767, August 1993.

[13] D.M. Gorinevsky and T.H. Connolly, "Comparison of Inverse Manipulator Kinematics Approximations from Scattered Input-Output Data Using ANN-like Methods," *American Control Conf.*, San-Francisco, CA, pp. 751-755, June 1993.

[14] T. Poggio and F. Girosi, "Networks for Approximation and Learning," *Proceedings of the IEEE*, vol.78, no.9, pp. 1481-1497, 1990.

[15] M.J.D. Powell, "Radial Basis Functions for Multivariable Interpolation: A Review," J.C. Mason and M.G. Cox, eds., *Algorithms for Approximation*, pp. 143-168, Clarendon Press, Oxford, 1987.

[16] M.J.D. Powell, "The Theory of Radial Basis Function Approximation in 1990,"W. Light, ed., *Advances in Numerical Analysis*, vol. 2, pp. 102-205, Clarendon Press, Oxford, 1992.

[17] G.V. Puskorius and L.A. Feldkamp, "Neurocontrol of Nonlinear Dynamical Systems with Kalman Filter-Trained Recurrent Networks," *IEEE Tr. on Neural Networks*, vol. 5, no. 2, pp. 279-297, 1994.

[18] G.V.Puskoris, L.A. Feldkamp, and L.I.Davis, Jr., "Dynamic Neural Network Methods Applied to On-Vehicle Idle Speed Control," *International Conference on Neural Networks*, Washington, D.C., June 1996, pp. 238-243, 1996.

[19] D. Shim, J. Park, P.P. Khargonekar, and W. Ribbens, "Engine Idle Speed Control," American Control Conference, Seattle, WA, pp. 2582-2586, June 1995.

[20] D. Torfs, D.M. Gorinevsky, and A.A. Goldenberg, "Experiments in Feedforward Shaping Control of Direct-Drive Robot Using RBF Network," *IEEE World Congress on Computational Intelligence*, Orlando, FL, June 1994.

[21] G. Vachtsevanos, S.S. Farinwata, and D.K. Pirovolu, "Fuzzy Logic Control of an Automotive Engine," *IEEE Cont. Syst. Magazine*, vol. 13, no. 3, pp. 62-68, 1993.

Dimitry Gorinevsky (M'91) received the M.S. diploma (1982) from the Moscow Institute of Physics and Technology and the Ph.D. (1986) from Moscow State University. He is currently an adjunct professor in the Department of Electrical Engineering at the University of British Columbia and a senior control engineer with Measurex Devron Inc., North Vancouver, B.C., developing advanced process control systems for the paper industry. He worked in process control, automotive, robotics, satellite control, and biomechanics. He has authored and coauthored

more than 70 technical papers and a book, and has six patents received or pending. Dr. Gorinevsky was awarded the Alexander von Humboldt International Research Fellowship in 1991 and is listed in 1993/1994 Who's Who in the World. He is a member of several professional organizations and a registered Professional Engineer in Ontario.

Lee A. Feldkamp received the B.S.E. degree in electrical engineering (1964) and the M.S. (1965) and Ph.D. (1969) degrees in nuclear engineering, all from the University of Michigan. He is currently a senior staff technical

specialist with the Vehicle Electronic Systems Department of Ford Research Laboratory. He has authored one book, more than 90 papers and reports, and was a co-recipient, with Gintaras V. Puskorius, of the 1995 *IEEE Transactions on Neural Networks* Outstanding Paper Award. Dr. Feldkamp is a member of the IEEE, and is a member of the Board of Governors of the International Neural Networks Society.