Analog Fault Detection Using RBF Neural Networks

MAHDI YUSEFKHANI, JAFAR TANHA, NASER AAYAT
Department of Computer Engineering & IT
PAYAME NOOR University
4Rahe Lashgarak, MiniCity, Tehran
IRAN

Abstract: Micro Electro Mechanical Systems will soon usher in a new technological renaissance. Just as ICs brought the pocket calculator, PC, and video games, MEMS will provide a new set of products and markets. Learn about the state of the art, from inertial sensors to microfluidic devices.
Over the last few years, considerable effort has gone into the study of the failure mechanisms and reliability of MEMS. Although still very incomplete, our knowledge of the reliability issues relevant to MEMS is growing. One of the major problems in MEMS production is fault detection. After fault detection or fault diagnosis, we can use hardware or software methods to overcome it. Most of MEMS have nonlinear and complex models. So it is difficult or impossible to detect the faults by traditional methods, which are model-based. In this paper RBF neural network is used for fault detection in a RF MEMS.

Key-Words: - Fault detection, Neural Networks, MEMS, Analog Fault.

1 Introduction

Reliability of Micro Electro Mechanical Systems (MEMS) is a very young and fast-changing field. Fabrication of a MEMS System involves many new tools and methods, including design, testing, packaging and reliability issues. Especially the latter is often only the very last step that is considered in the development of new MEMS. The early phases are dominated by considerations of design, functionality and feasibility; not reliability [1].
One important reason for missing reliability data is that in view of the use of new materials and process, the material data, the know-how on failure modes, the means and the procedures to perform reliability tests and consequent failure analysis are often not present and unknown [2].
The traditional approaches to fault detection and diagnosis involve the limit checking of some variables or the application of redundant sensors. More advanced methods rely on the spectral analysis of signals emanating from the machinery or on the comparison of the actual plant behavior to that expected on the basis of a mathematical model. The latter approach includes methods which are more deterministically framed and those formulated on more of a statistical basis, and parameter estimation.
In methods based on mathematical models, the models obtained must be linear. To work with non-linear systems, it is necessary to select a point and obtain a linearized model around it [3].
In MEMS most of the parts are strictly non linear and finding a proper model is difficult or sometimes impossible. So using mathematical model for fault detection in MEMS has some drawbacks. The constraints of this kind of model have motivated the development of artificial intelligent approaches.
In this paper, we will use neural networks for fault detection in MEMS.

2 Fault detection methods

The work on fault diagnosis in the AI community initially focused on the expert system or knowledge-based approach where heuristics are applied to explicitly associate symptoms with fault hypothesis. The short coming of a pure expert system approach led to the development of model-based approaches based on qualitative models in the form of qualitative differential equations, signed digraphs, qualitative functional and structural models.
Other approaches assume the availability of process history based data which are then used to develop neural network approaches.
Neural networks mimic intelligence. The learning or training nodes of neural networks is different from that of traditional statistical methods.
The results of comparison between a Model Based Fault Detection method (MBFD) and a Neural Network fault detection and classification method is provided in Table1 [4].
As we can see, both of them have their strengths and weaknesses. In MEMS usually there is not a proper and accurate model. Additionally, our knowledge about faults, their sources and effects is not

complete. So usually there are novel or undetermined faults which are not considered in model. As a result neural network is a proper tool for fault detection and classification in MEMS.

Criterion	MBFD	NN
Novel faults	Poor	Fair
Robustness to noise	Fair	Good
Resolution	Fair	Good
Adaptability	Good	Fair
Range of application	Good	Bad

Table1- Comparison of MBFD and NN

Generally speaking, there are four types of neural networks:

-Back propagation Neural Network (BPNN)

-Probabilistic Neural Network (PNN)

-Self-Organizing Mapping (SOM)

-Radial Basis Function Neural Network (RBF)

There are some drawbacks to BPNN and SOM. The BPNN requires a large number of training patterns to let network learn the underlying mapping function. The second problem is that the accuracy of the training patterns should not be a measure of whether a model is good or not. BPNN has a low reliability with novel data.

SOM is known as a topological mapping algorithm, in which patterns with similar characteristics cluster together automatically. Output nodes will thus be ordered by competitive learning. The learning rate and neighbor size of SOM have to be optimally selected by experience, and a SOM net needs a large time to converge [5].

In this paper RBF is used for fault detection and classification in MEMS. Two different methods are used for learning. The first is Derivative-based optimization method and in the other method Kalman filtering is used for optimal selection of Radial Basis Functions.

3 Radial Basis Functions

There have been a number of popular choices for the *g(.)* function at the hidden layer of RBFs. The most common choice is a Gaussian function of the form:

$$g(u) = \exp(-u / b^2) \qquad (1)$$

Where, β is a real constant. Other hidden layer functions that have often been used are the multiquadric function:

$$g(u) = (u^2 + b^2)^{1/2} \qquad (2)$$

And the inverse multiquadric function:

$$g(u) = (u^2 + b^2)^{-1/2} \qquad (3)$$

Where β is a real constant. Scince RBF prototypes are generally interpreted as the centers of receptive fields, hidden layer functions should have the following properties:

1-The response at a hidden neuron is always positive.

2-The response at a hidden neuron becomes stronger as the input approaches the prototype.

3- The response at a hidden neuron becomes more sensitive to the input as the input approaches the prototype.

3.1 Derivative Based Optimization

The response of an RBF, with the hidden layer function, g(.) can be written as follows:

$$\hat{y} = \begin{bmatrix} w_{10} & w_{11} & ..... & w_{1c} \\ w_{20} & w_{21} & ..... & w_{2c} \\ \vdots & \vdots & & \vdots \\ w_{n0} & w_{n1} & ..... & w_{nc} \end{bmatrix} \begin{bmatrix} 1 \\ g(\|x-u_1\|^2) \\ \vdots \\ g(\|x-u_2\|^2) \end{bmatrix} \qquad (4)$$

We will use the following notation as short hand for the weight matrix on the right-hand side of Eq(5).

$$\begin{bmatrix} w_{10} & w_{11} & ..... & w_{1c} \\ w_{20} & w_{21} & ..... & w_{2c} \\ \vdots & \vdots & & \vdots \\ w_{n0} & w_{n1} & ..... & w_{nc} \end{bmatrix} = \begin{bmatrix} w_1^T \\ w_2^T \\ \vdots \\ w_n^T \end{bmatrix} = W \qquad (5)$$

If we are given a training set of M desired input-output responses {x$_i$,y$_i$}, (i=1,2,…,M), then we can augment M equations of the form of Eq(4) as follows:

$$[\hat{y}_1 \quad \hat{y}_2 \quad .... \quad \hat{y}_M] = W \begin{bmatrix} 1 & .... & 1 \\ g(\|x_1-u_1\|^2) & .... & g(\|x_M-u_1\|^2) \\ \vdots & .... & \vdots \\ g(\|x_1-u_c\|^2) & .... & g(\|x_M-u_c\|^2) \end{bmatrix} \qquad (6)$$

We will introduce the following notation for the matrix on the right hand side of Eq(6).

$$h_{0k} = 1 \quad (k = 1,...M) \qquad (7)$$

$$h_{jk} = g(\|x_k - u_j\|^2) \quad (k = 1,...M), (j = 1,...,c) \qquad (8)$$

In this case we can write the matrix on the right-hand side of Eq(6) as:

$$\begin{bmatrix} h_{01} & \cdots & h_{0M} \\ h_{11} & \cdots & h_{1M} \\ \vdots & \vdots & \vdots \\ h_{c1} & \cdots & h_{cM} \end{bmatrix} = [h_1 \quad .... \quad h_M] = H \qquad (9)$$

In this case we can write Eq(6) as

$$\hat{Y} = WH \qquad (10)$$

Now, if we want to use gradient descent to minimize the training error, we can define the error function:

$$E = \frac{1}{2}\left\| Y - \hat{Y} \right\|^2 \qquad (11)$$

It has been shown that gradient of error can be computed as

$$\frac{\partial E}{\partial w_i} = \sum_{k=1}^{M} (\hat{y}_{ik} - y_{ik}) h_k \qquad (i = 1,..., n) \qquad (12)$$

$$\frac{\partial E}{\partial v_j} = \sum_{k=1}^{M} 2g'(\left\| x_k - v_j \right\|^2)(x_k - v_j) \sum_{i=1}^{n} (y_{ik} - \hat{y}_{ik}) w_{ij} \qquad (13)$$

$$(j = 1,..., c)$$

Now we can optimize the RBF with respect to the rows of the weight matrix W and the prototype location v_j by iteratively computing the above partials and performing the following updates:

$$w_i = w_i - h\frac{\partial E}{\partial w_i} \qquad (i = 1,..., n) \qquad (14)$$

$$v_j = v_j - h\frac{\partial E}{\partial v_j} \qquad (j = 1,..., c) \qquad (15)$$

Where the η is the step size of the gradient descent method. This optimization stops when w_i and v_j reach local minima.

3.2 Using Kalman Filter for Optimization

Alternatively we can use kalman filtering to minimize the training error. Derivation of the extended kalman filter are widely available in the literature [6]. Consider a nonlinear finite dimensional discrete time system of the form

$$q_{k+1} = f(q_k) + w_k \qquad (16)$$

$$y_k = h(q_k) + v_k \qquad (17)$$

Where the vector θ_k is the state of the system at time k, w_k is the process noise, y_k is the observation vector, v_k is the observation noise, and *f(.)* and *h(.)* are nonlinear vector functions of the state. Assume that the initial state θ_0 and the noise sequence $\{v_k\}$ and $\{w_k\}$ are Gaussian and independent from each other with

$$E(q_0) = \bar{q}_0 \qquad (18)$$

$$E[(q_0 - \bar{q}_0)(q_0 - \bar{q}_0)^T] = P_0 \qquad (19)$$

$$E(w_k) = E(v_k) = 0 \qquad (20)$$

$$E(w_k w_l^T) = Qd_{kl} \qquad (21)$$

$$E(v_k v_l^T) = Rd_{kl} \qquad (22)$$

The problem addressed by the extended Kalman filter is to find an estimate $\hat{q}_{k+1}$ of q_{k+1} given y_j (j=0,1,…,k). If the nonlinearity in Eq(16,17) are sufficiently smooth, we can expand them around the state estimate $\hat{q}_k$ using Taylor series to obtain

$$f(q_k) = f(\hat{q}_k) + F_k \times (q_k - \hat{q}_k) + HOT \qquad (23)$$

$$h(q_k) = h(\hat{q}_k) + H_k^T \times (q_k - \hat{q}_k) + HOT \qquad (24)$$

Where HOT is higher order terms and

$$F_k = \left.\frac{\partial f(q)}{\partial q}\right|_{q=\hat{q}_k} \quad \text{and} \quad H_k^T = \left.\frac{\partial h(q)}{\partial q}\right|_{q=\hat{q}_k} \qquad (25)$$

Neglecting the higher order terms in Eq(23,24), the system in Eq(16,17) can be approximated as

$$q_{k+1} = F_k q_k + w_k + f_k \qquad (26)$$

$$y_k = H_k^T + v_k + j_k \qquad (27)$$

Where φ_k and ϕ_k are defined as

$$f_k = f(\hat{q}_k) - F_k\hat{q}_k \qquad (28)$$

$$j_k = h(\hat{q}_k) - H_k^T\hat{q}_k \qquad (29)$$

It can be shown that the desired estimate $\hat{q}_n$ can be obtained by the recursion [7]

$$\hat{q}_k = f(\hat{q}_{k-1}) + K_k[y_k - h(\hat{q}_{k-1})]; \qquad (30)$$

$$K_k = P_k H_k (R + H_k^T P_k H_k)^{-1}; \qquad (31)$$

$$P_{k+1} = F_k(P_k - K_k H_k^T P_k)F_k^T + Q. \qquad (32)$$

K_k is known as the kalman gain. In the case of a linear system, it can be shown that P_k is the covariance matrix of the state estimation error, and the state estimate $\hat{q}_{k+1}$ is optimal in the sense that it approaches the conditional mean $E[q_{k+1} | (y_0, y_1,..., y_k)]$ for large *K*.

We can apply a similar technique to the training of RBF networks. In general we can view the optimization of the weight matrix *W* and the prototypes v_j as a weighted least-squares minimization problem, where the error vector is the difference between the RBF outputs and the target values for those outputs. Consider the RBF network with *m* inputs, *c* prototypes, and *n* outputs. We use *y* to denote the target vector for the RBF outputs, and $h(\hat{q}_k)$ to denote the actual outputs at the *k*th iteration of the optimization algorithm.

$$Y=[y_{11} \dots y_{1M} \dots y_{n1} \dots y_{nM}]^T \qquad (33)$$

$$h(\hat{q}_k) = [\hat{y}_{11} \dots \hat{y}_{1M} \dots \hat{y}_{n1} \dots \hat{y}_{nM}]_k^T \qquad (34)$$

N is the dimension of the RBF output and M is the number of training samples. The state of the nonlinear system can then be represented as

$$\theta=[w_1^T \dots w_n^T \; v_1^T \; \dots v_c^T \;]^T \qquad (35)$$

The vector θ thus consist of all $(n(c+1)+mc)$ of the RBF parameter arranged in a linear array. The nonlinear system model to which the Kalman filter can be applied is

$$.\theta_{k+1}= \theta_k \, , \qquad (36)$$

$$Y_k=h(\theta_k) \qquad (37)$$

Where $h(\theta_k)$ is the RBF network's nonlinear mapping between its parameters and its output. In order to execute a stable kalman filter algorithm, we need to add some artificial process noise and measurement noise to the system model. So we rewrite Eq(36,37) as

$$\theta_{k+1}= \theta_k + w_k \qquad (38)$$

$$Y_k=h(\theta_k)+v_k \qquad (39)$$

Where w_k and v_k are artificially added noise processes. Now we can apply the kalman recursion of Eqations. *f(.)* is the identity mapping and y_k is the target output of the RBF network. $h(\hat{q}_k)$ is the actual output of the RBF network given the RBF parameters at the *k*th iteration of the kalman recursion. F_k is the identity matrix (a constant even though it is written as a function of *k*). The Q and R matrices are tuning parameters which can be considered as the covariance matrices of the artificial noise processes w_k and v_k , respectively. It can be shown that the partial derivative of the RBF output with respect to the RBF network parameters is given by:

$$H_k = \begin{bmatrix} H_w \\ H_v \end{bmatrix} \qquad (40)$$

Where H_w and H_v are given by:

$$H_w = \begin{bmatrix} H & 0 & \dots & 0 \\ 0 & H & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & \dots & 0 & H \end{bmatrix} \qquad (41)$$

$$H_v = \begin{bmatrix} -w_{11}g'_{11}2(x_1-v_1) & \dots & -w_{11}g'_{m1}2(x_m-v_1) & \dots & -w_{n1}g'_{11}2(x_1-v_1) & \dots & -w_{n1}g'_{m1}2(x_m-v_1) \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ -w_{1c}g'_{1c}2(x_1-v_c) & \dots & -w_{1c}g'_{mc}2(x_m-v_c) & \dots & -w_{nc}g'_{1c}2(x_1-v_c) & \dots & -w_{nc}g'_{mc}2(x_m-v_c) \end{bmatrix} \qquad (42)$$

Where *H* (with no subscript) is the $(c+1)_* M$ matrix given in Eq(9), w_{ij} is the element in the *i*th row and *j*th column of the *W* weight matrix, $g'_{ij} = g'(\left\| x_i - v_j \right\|^2)$, x_i is the *i*th input vector, and v_j is the *j*th prototype vector. H_w is an $n(c+1)_*nM$ matrix, H_v is an mc_*nM matrix, and H_k is an $[n(c+1)+mc]_*nM$ matrix. Now that we have the H_k matrix, we can executethe recursion of Eq(30,31,32), thus using the extended kalman filter in order to determine the weightmatrix *W* and the prototypes v_j.

4 Solution Results

EM3DS is a MEMS simulator software, which has been used for fault simulation in RF MEMS. 20 faults and one fault free pattern have been simulated in a RF low pass filter MEMS. These 20 faults consist of both digital and analog faults. Changing substrate resistance, magnetic and electric properties, short and open, disconnection, connection between separate parts and some other faults have been simulated by software. The S parameters are calculated and used for training and testing RBF neural network.

We have used a two dimension data as input to RBF neural network. First of all, two RBF networks with Gaussian and multiquadric function are used. The training algorithm is gradient descent. The result has been shown in Tables 2, 3.

	Detected as Fault	Detected as Fault free	Correct fault detection percent
40 Faulty Pattern	25	15	%62.5
10 Fault free Pattern	4	6	%60
Total 50			%62

Table2- Multiquadric function and gradient descent learning

	Detected as Fault	Detected as Fault free	Correct fault detection percent
40 Faulty Pattern	26	14	%65
10 Fault free Pattern	4	6	%60
Total 50			%64

Table3- Gaussian function and gradient descent learning

In second method, the same RBF networks are used but for learning, kalman filter has been used. Kalman filter helps the network to find optimum parameters. The results for the same patterns have been shown in Table4, 5.

	Detected as Fault	Detected as Fault free	Correct fault detection percent
40 Faulty Pattern	29	11	%72.5
10 Fault free Pattern	4	6	%60
Total 50			%70

Table4- Multiquadric function and Kalman filter learning

	Detected as Fault	Detected as Fault free	Correct fault detection percent
40 Faulty Pattern	31	9	%77.5
10 Fault free Pattern	3	7	%70
Total 50			%76

Table5- Gaussian function and Kalman filter learning

4 Conclusion

Fault detection in MEMS is an important issue in MEMS production and maintenance. Most of the faults occure in microscopic dimension and we have not enough knowledge about it. Finding a proper model is usually difficult or even impossible. In this paper neural networks are proposed for fault detection. BPNN and Kohonen neural networks have some drawbacks. There are always novel faults in MEMS, which have not learnt to neural network, already. BPNN and Kohonen networks can't find these faults properly and correct fault detection percent is about 50-60.
In this paper, RBF neural networks have been used for fault detection. The results in Tables2,3,4,5 show that Gaussian function with kalman filter learning algorithm has the best result. The results are better than BPNN and Kohonen neural networks.
Further research could focus on the application of kalman filter training to RBF network with alternative forms of the generator functions. In addition, the convergence of the kalman filter could be further improved by more intelligently initializing the training process.

References:

[1] B. Murari, Integrating Nonelectronic Components Into Electronic Microsystems, *IEEE Micro*, Vol.23, No.3, May/June 2003, pp. 36-44

[2] R.Muller, U.Wagner, Reliability of MEMS-a methodical approach, *Proc. 11th European symposium on reliability of electron devices, failure physics and analysis*, 2001, pp.1657-62.

[3] W.Liu, An Extended Kalman Filter and Neural Network Cascade Fault Diagnosis Strategy for Glutamic Acid Fermentatiion Process, *Journal of Artificial Intelligence in Engineering, ELSEVIER*, Vol.13, 1999, pp.131-140.

[4] R.Rengaswamy, D.Mylaraswamy, A Comparison of Model-Based and Neural Network-Based Diagnostic Methods, *Engineering Application of Artificial Intelligence, PERGAMON*, Vol.14, 2001, pp. 805-818

[5] Z.Yang, M.Zwolinski, Applying a Robust Heteroscedastic Probabilistic Neural Network to Analog Fault Detection and Classification, *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, Vol.19, No.1, 2000, pp. 142-151.

[6] B.Anderson, J.Moore, *Optimal Filtering*, Prentice-Hall, Englewood Cliffs, NJ, 1979.

[7] Dan Simon, Training radial basis neural networks with the extended Kalman filter, *Neurocomputing Journal, ELSEVIER*, Vol.48, Oct. 2002, pp.455-475.