Delaunay-based Local Model Networks for Nonlinear System Identification

T. ULLRICH and H. TOLLE
Darmstadt University of Technology, Dept. Control Systems Theory & Robotics
Landgraf-Georg Strasse 4, D-64283 Darmstadt
E-Mail: thul@rt.e-technik.th-darmstadt.de

Keywords

Nonlinear Systems, Delaunay Networks, Automotive Applications

Abstract

This paper presents an efficient methodology for nonlinear system identification based on Delaunay networks. These networks [1] share the learning capabilities of artificial neural networks but they are computationally much more efficient. In the approach discussed here, the interpolation nodes of a Delaunay network are interpreted as *local linear models* of a nonlinear plant. Hence, standard parameter estimation techniques can be applied to train the network. Combined with a heuristic strategy for structural optimization, the proposed method constructs nonlinear models which can be implemented on low-cost microcontrollers and can meet strict realtime requirements. Thus, the method is a promising approach in fields of application where computer power is a limited resource. Two examples of automotive applications are discussed and exemplify the approach.

1 Introduction

Due to the increasing demand for environmental compatibility, automotive applications, particularly engine management systems, are an area where nonlinear model-based control and supervision techniques need to be applied. However, computer power and storage capacity are expensive resources in this field of application. The aim of this work therefore is the development of methods for automatic construction of nonlinear models which can then be implemented on low-cost hardware.

Delaunay networks [1] approximate a nonlinear function by means of a set of *interpolation nodes* and the *Delaunay triangulation* is used to define the network topology. Extending earlier work, a set of parameters of a locally valid *linear* model are assigned to each node. Hence, the approach is similar to the local model networks proposed in [2], [3] and [4] and to Takagi-Sugeno fuzzy systems [5]. From a control perspective, the method is an extended *gain-scheduling* technique.

The concept of local model networks is summarized in section 2.1 and section 2.2 reviews the Delaunay network approach. Section 3 then introduces a procedure for data-driven construction of Delaunay-based models (system identification). This procedure is applied to two automotive modelling problems in section 4.

2 Local Modelling

2.1 Local Model Networks

The idea of *Local Model Networks* (LMN) is to approximate a nonlinear system with a set of relatively simple (and typically linear[1]) models. Each local model is valid in a certain operating regime, i.e. a subdomain of the input space. Therefore, *validity functions* are assigned to the local models. For a given input vector, the responses of all local models are smoothly interpolated according to their respective degree of validity. A comprehensive discussion of this approach can be found in [2]. Similar ideas have been applied in [6] to interpolate local controllers.

A LMN that imitates the behaviour of a nonlinear multi-input single-output plant $y(\underline{\zeta})$ is defined as

$$y(\underline{\zeta}) \approx \widehat{y}(\underline{\zeta}) = \sum_{i=1}^{N} \widehat{f}_i(\underline{\zeta}) \cdot \Phi_i(\underline{\zeta}), \quad \underline{\zeta} \in I \subset I\!R^m. \quad (1)$$

Generally, $\underline{\zeta}$ may comprise different physical inputs as well as the dynamic history of these inputs.

The validity functions $\Phi_i(\underline{\zeta})$ assigned to the N local models $\widehat{f}_i$ are normalized in the sense that

$$\sum_{i=1}^{N} \Phi_i(\underline{\zeta}) = 1, \quad \forall \underline{\zeta} \in I \subset I\!R^m. \quad (2)$$

The models $\widehat{f}_i$ are again nonlinear functions in the general case. Local *linear* models are obtained, if each local model $\widehat{f}_i$ is a Taylor series expansion of $y(\underline{\zeta})$ around the model's *position* $\underline{\zeta}_i$ in the input space. In case of

[1] Throughout this paper, the term 'linear model' denotes a model which is linear with respect to its parameters.

zeroth-order expansions, i.e. $\widehat{f}_i(\underline{\zeta}) = \widehat{\Theta}_i \approx y(\underline{\zeta}_i)$, the LMN is equivalent to conventional basis function models such as RBF, B-Spline and CMAC networks. A *first-order* expansion yields the following model structure:

$$y(\underline{\zeta}) \approx \widehat{y}(\underline{\zeta}) = \sum_{i=1}^{N} \underbrace{[\widehat{\Theta}_{i0} + \underline{\widehat{\Theta}}_i^T \cdot (\underline{\zeta} - \underline{\zeta}_i)]}_{\widehat{f}_i} \cdot \Phi_i(\underline{\zeta}). \quad (3)$$

This is equivalent to a single model the parameters of which depend on the input vector $\underline{\zeta}$. Thus, by combining constant values, (3) can also be written as

$$y(\underline{\zeta}) \approx \widehat{y}(\underline{\zeta}) = \widehat{\Theta}_0(\underline{\zeta}) + \underline{\widehat{\Theta}}^T(\underline{\zeta}) \cdot \underline{\zeta} \quad (4)$$

and any method for function approximation can be applied to represent the characteristics $\widehat{\Theta}_0(\underline{\zeta})$ and $\underline{\widehat{\Theta}}(\underline{\zeta})$.

If the input space is high-dimensional, the implementation cost of a network that represents these characteristics is high. However, in many applications a-priori knowledge is available and can be exploited to reduce the input space dimension. It is assumed that the designer can specify the input variables with respect to which the plant behaviour is strongly nonlinear. Only those inputs should be used for parameter scheduling in terms of equation (4). The input vector $\underline{\zeta}$ is thus decomposed into a *scheduling vector* $\underline{x}$ and an *information vector* $\underline{\psi}$ which is processed by a linear model. Equation (4) can then be written as

$$y(\underline{\zeta}) \approx \widehat{y}(\underline{\zeta}) = \begin{pmatrix} \widehat{\Theta}_0(\underline{x}) \\ \underline{\widehat{\Theta}}(\underline{x}) \end{pmatrix}^T \cdot \begin{pmatrix} 1 \\ \underline{\psi} \end{pmatrix} = \begin{pmatrix} \widehat{\Theta}_0(\underline{x}) \\ \underline{\widehat{\Theta}}(\underline{x}) \end{pmatrix}^T \cdot \underline{\psi}.$$

Although this qualitative knowledge is available in many engineering applications, it should be mentioned that there are methods to extract such information automatically [7, 8] as well.

2.2 Delaunay-Based Local Model Networks

The concept of local linear models has attracted many researchers' interest recently [3, 4, 7]. The main difference between these approaches is the method used to interpolate the local models. The method proposed here, is to use Delaunay networks for this task. This is motivated by the computational efficiency of the Delaunay network paradigm. Moreover, algorithms for automatic construction of such networks have been developed.

Delaunay networks consist of a set of N interpolation nodes. Each node comprises a position $\underline{p}_i$ in the network's input space (*scheduling space*) and a vector $\underline{\widehat{\Theta}}_i$ of p parameters. The distribution of the nodes in the scheduling space is adjusted to the 'local degree of nonlinearity' of the modelled nonlinear plant[2]. In

[2]The question how to construct such a distribution is dealt with in section 3.

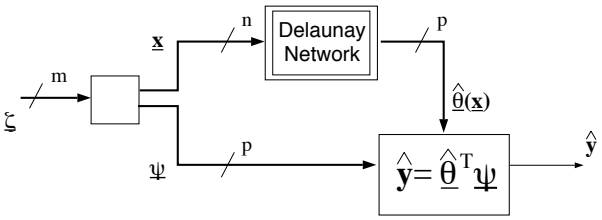

Figure 1: *Parameter scheduling based on Delaunay networks. The m-dimensional input vector $\underline{\zeta}$ is decomposed into a n-dimensional scheduling vector $\underline{x}$ and a p-dimensional information vector $\underline{\psi}$.*

those regions where the parameters vary strongly with respect to the scheduling variables, the node density is increased, whereas in regions where the plant shows only weak nonlinearities, a lower node density suffices.

The *Delaunay triangulation* is a graph that connects subsets of $n+1$ nodes in the n-dimensional scheduling space so that they define non-intersecting *simplices* (triangles if $n = 2$). These simplices possess the *empty circle property*: The n-dimensional circumsphere of a Delaunay simplex contains no other node of the network [9]. It is this empty circle property that leads to high accuracy when linear interpolation within the simplices is applied [10].

The *output computation* procedure of Delaunay networks as proposed in [1] comprises two steps. At first, the Delaunay simplex that contains the current scheduling vector $\underline{x}$ is searched for. The vertices of this simplex are the *active nodes* for the input $\underline{x}$ and contribute to the network response which is computed in the second step. The search for the active nodes is guided by the signs of *barycentric coordinates*. Since the scheduling variables are continuous over time, this is a very efficient method that leads to short response times. This search algorithm is described in [1] and [11] in greater detail. Formally, the selection of the active nodes yields a vector of N interpolation weights (N denotes the total number of nodes in the network). However, this weight vector $\underline{\widetilde{b}}(\underline{x})$ contains non-zero elements only at those positions that correspond to the active nodes.

After the $n+1$ active nodes for the input $\underline{x}$ have been determined, the network response is computed by linearly interpolating the *attributes* assigned to these nodes. In case of local model networks as discussed here, the attributes assigned to each node comprise a set of p parameters (p equals the dimension of the information vector $\underline{\psi}$, cf figure 1).

Figure 2 illustrates the selection of the active nodes and the calculation of their respective interpolation weights. In the depicted example, the input vector $\underline{x}$ lies within the triangle defined by the nodes 3, 4 and 6. Hence, only these nodes contribute to the network response. Accordingly, only the third, fourth and

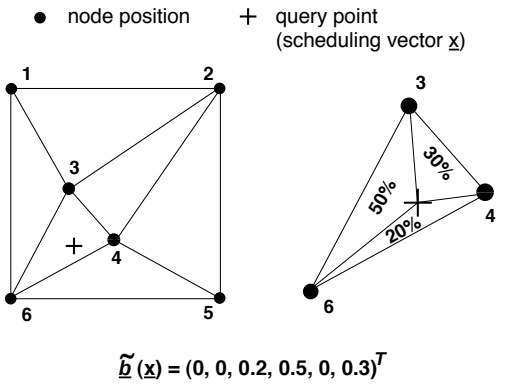

Figure 2: *Selection of active nodes and calculation of the respective interpolation weights.*

sixth element of the weight vector $\underline{\tilde{b}}(\underline{x})$ are non-zero. The interpolation weights that correspond to the active nodes are given by the respective barycentric coordinates. Geometrically, these weights are the relative areas of the 'subsimplices' defined by n of the $n+1$ active nodes and the input vector $\underline{x}$ (cf figure 2 right).

In the following, $\underline{\widehat{\Theta}}_i$ denotes the parameter set assigned to the i-th node, $i = 1, \ldots, N$. $\widehat{\Theta}_{ij}$ is the j-th parameter assigned to the i-th node, $j = 1, \ldots, p$. $\widehat{\mathbf{\Theta}} = (\widehat{\Theta}_{ij})$ denotes the network's parameter matrix. The model response $\widehat{y}(\underline{x}, \underline{\psi})$ is then given by

$$\underbrace{\widehat{y}(\underline{x}, \underline{\psi})}_{\text{scalar}} = \underbrace{\underline{\widehat{\Theta}}^T(\underline{x})}_{(1\times p)} \cdot \underbrace{\underline{\psi}}_{(p\times 1)} = \underbrace{\underline{\tilde{b}}^T(\underline{x})}_{(1\times N)} \cdot \underbrace{\widehat{\mathbf{\Theta}}}_{(N\times p)} \cdot \underbrace{\underline{\psi}}_{(p\times 1)}. \quad (5)$$

3 Model Construction

Building a parameter-scheduling model comprises two steps:

1. Definition of scheduling variables and dynamic order of the linear model.
2. Automatic design of the Delaunay network.

The first problem is assumed to be solved either by exploiting expert knowledge or by applying automatic input selection techniques. Besides, the input selection problem can also be tackled interactively due to the efficiency of the model construction methodology discussed in the following. Typical modelling problems require only little CPU-time (some seconds to some minutes, depending on the complexity of the problem and the size of the dataset). Hence, various ($\underline{x}$, $\underline{\psi}$)-decompositions and different dynamic orders can easily be assessed.

The design of the Delaunay network again comprises two problems. A node *distribution* that accounts for the local complexity of the modelled plant needs to be determined. Furthermore, the *parameters* assigned to the nodes have to be tuned so that the modelling error is minimized.

The construction of an appropriate node distribution is a complex combinatorial optimization problem and therefore heuristic search strategies need to be applied. Three such strategies that construct a Delaunay network incrementally are compared in [12]. Section 3.2 discusses one of these methods which is particularly adequate for local linear modelling of *dynamical* systems. Tuning the elements of the parameter matrix $\widehat{\mathbf{\Theta}}$, however, is a standard problem of linear parameter estimation. This is outlined in section 3.1.

3.1 Parameter estimation

This section deals with the question how to tune the $N \cdot p$ elements of a Delaunay network's parameter matrix $\widehat{\mathbf{\Theta}}$ given a set of M samples, comprising the inputs $\underline{x}_i$ and $\underline{\psi}_i$ as well as the desired responses $y_i^d, \quad i = 1, \ldots, M$.

Given the network's node distribution, this estimation problem is *linear* with respect to the adjustable parameters provided that a series-parallel model structure is applied [13]. Therefore, the parameters can be determined by linear least squares estimation. Note that the series-parallel structure (i.e. one-step-ahead prediction) is only assumed during model construction. Hereafter, the model can be run in parallel with the plant as well, i.e. perform long-term predictions.

There are two different methods of parameter estimation for LMNs. The first is to estimate all $N \cdot p$ parameters simultaneously. This is referred to as *global least squares tuning*. The second possibility is to tune the parameters assigned to the N nodes separately. This *local least squares tuning* method decomposes the original estimation problem which is of size $N \cdot p$ into N estimation problems of size p. The respective advantages of global and local tuning are compared in [4]. Here, the global tuning method is applied since this leads generally to higher model accuracy.

3.2 Structural optimization

In [12] three heuristic strategies for the construction of an appropriate node distribution are discussed. These methods work *incrementally*, i.e. the construction procedure is started with a small initial network into which nodes are inserted on the basis of heuristic criteria. The procedure is stopped when a user-defined criterion is met. The following sections deal with *initialization*, *node insertion* and the choice of the *stopping criterion*.

3.2.1 The inital network

The initial network typically contains only 2^n nodes which are the vertices of the embedding cube of the n-dimensional scheduling domain (see figure 2 where the nodes 1, 2, 5 and 6 are the vertices of the embedding square of the two-dimensional scheduling domain). It is hereby guaranteed that the network response is always computed by *inter*polation whereas error-prone *extra*polations are avoided.

3.2.2 Iterative Node Insertion

The aforementioned heuristic strategies for node insertion [12] assess the *local accuracy* of the network. Additional nodes are inserted in those areas of the scheduling space where large modelling errors are observed. When modelling densely sampled dynamic systems, the amount of measurement data is usually large. Therefore, the so-called *local-weighted-error*-strategy from [12] is advantageous and applied here since its computational complexity depends only weakly on the size of the dataset.

The network's $N \cdot p$ parameters are re-estimated after each node insertion. The two operations node insertion and parameter estimation are repeated iteratively until a user-defined stopping criterion is met.

3.2.3 The stopping criterion

The specification of the stopping criterion should be guided by three objectives:

(1) Model accuracy
(2) Computational resources
(3) Avoidance of overfitting

Thus, a user-defined minimum accuracy (objective 1) or a maximum number of nodes (objective 2 with respect to limited storage capacity of the target system) might be used to stop the construction process. On the other hand, it is essential to prevent overfitting, i.e. the modelling of measurement noise. Since overfitting leads to low reliability and poor generalization capabilities (see e.g. [4, 7]), the construction procedure must be stopped before overfitting occurs. If the amount of measurement data is large enough, the dataset should be split into a *training set* and a *test set*. The training set is used for parameter tuning (section 3.1) and node insertion (section 3.2.2), whereas the test set is only used to monitor the model's generalization capabilities. The construction procedure should be stopped when further node insertions do not improve the model accuracy on the test set.

4 Automotive Applications

4.1 Engine torque modelling

This section exemplifies the proposed methods. At first, the identification of dynamic engine torque characteristics is investigated. To generate data for model construction and validation, a *simulation* of a combustion engine was used. Details of this simulation can be found in [11].

The variables with respect to which a combustion engine shows distinct nonlinear behaviour are the throttle valve position α_{thr} and the engine speed n_{eng}. This knowledge motivates the choice of α_{thr} and n_{eng} as scheduling variables. Moreover, the engine torque dynamics are assumed to be of first order with respect to throttle valve excitations. Hence, the model structure, i.e. the decomposition into $\underline{x}$ and $\underline{\psi}$, is given by

$$\begin{aligned} \underline{x}(k) &= \left(\begin{array}{cc} \alpha_{thr}(k) & n_{eng}(k) \end{array} \right)^T \\ \underline{\psi}(k) &= \left(\begin{array}{ccc} 1 & \alpha(k-1) & T(k-1) \end{array} \right)^T \end{aligned}$$

where T denotes the torque and k is the discrete time. The scheduling space is two-dimensional in this application and three parameters are assigned to each node ($n = 2,\ p = 3$, cf figure 1). One of the three parameters $\widehat{\Theta}_i$ (which corresponds to the constant '1' in the vector $\underline{\psi}$) is the local bias, whereas the other two parameters of each node determine the local gain and the local system time constant. Using a throttle valve profile acquired in a test vehicle and the aforementioned engine simulation, a set of 7000 samples were generated. 75% of these samples were used as training data, the remaining 25% served as the test set.

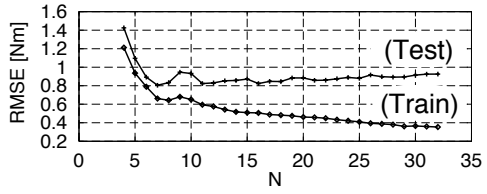

Figure 3: *Engine torque modelling: One-step-ahead prediction error (root mean squared error) on the training and test set during network construction.*

Figure 3 shows the one-step-ahead prediction errors on the two datasets during network construction, i.e. plotted versus the number of nodes N. This diagram suggests that the insertion procedure should be stopped at $N = 11$ to prevent overfitting. The construction of a model with 11 nodes requires 1.08 minutes CPU time on an Ultra-Sparc-140 workstation and it yields a one-step-ahead prediction error of 0.595 Nm on the training set and 0.823 Nm on the test set (root mean squared errors). Figure 4 shows the distribution of the 11 nodes in the scheduling space and their Delaunay triangulation.

Whilst one-step-ahead prediction is assumed during network construction, the resulting model can be run in parallel with the plant as well, i.e. perform long-term predictions. In this case the root mean squared errors are 1.481 Nm and 1.907 Nm on the training and test set. The maximum response time of the model is 0.179 msec (on a T805 microprocessor running at 30 MHz).

4.2 Turbocharger modelling

As a second example of an automotive application, this section discusses the identification of the charging pressure characteristics of a Diesel engine exhaust turbocharger. In contrast to the first example, a real-world dataset was used here. For a detailed description of the turbocharger modelling problem the reader is referred to [3].

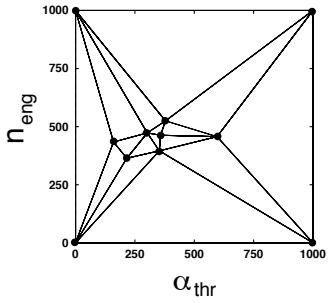

Figure 4: *Engine torque modelling: Distribution and Delaunay Triangulation of $N = 11$ nodes in the two-dimensional scheduling space. The scheduling inputs α_{thr} and n_{eng} are normalized to the range [0, 1000].*

Again, expert knowledge is available and motivates the choice of the injection rate m_b and the engine speed n_{eng} for parameter scheduling. The dynamic characteristics of the charging pressure p are assumed to be of second order with respect to the injection rate and engine speed. Hence, the scheduling space is two-dimensional and 7 parameters are assigned to each node ($n = 2,\ p = 7$ cf figure 1).

The training set for network construction comprises 1300 samples which cover all relevant operating conditions of the turbocharger and excite its dynamics. Two separate test sets (collected in urban and interstate traffic) are available to assess the model's generalization capabilities. Given this data, a model consisting of $N = 10$ nodes appeared to be appropriate (total number of parameters: $N \cdot p = 70$). If more than 10 nodes are inserted, the performance with respect to the test sets starts to deteriorate (overfitting).

The model yielded a one-step-ahead prediction error of 0.0112 bar on the training, 0.0157 bar on the urban test and 0.0148 bar on the interstate test set. The long-term prediction errors are 0.0201 bar, 0.0373 bar and 0.0309 bar, respectively. The maximum response time obtained on a T805 microprocessor is 0.295 msec.

5 Conclusion

This paper proposed Delaunay-based interpolation networks for local linear modelling. A significant advantage of this approach is its computational efficiency. The model construction algorithm discussed in section 3 exploits a-priori knowledge about the relevant input variables and the plant's degree of nonlinearity with respect to the different inputs. According to this knowledge, the input vector is decomposed into a scheduling vector and an information vector. Only the scheduling vector is directly processed by the Delaunay network and as a result, the network's input space can be low-dimensional. Two scheduling variables were sufficient in the automotive examples discussed in section 4. However, Delaunay networks with three or four inputs can also be implemented efficiently so that plants with even higher complexity can as well be identified with the approach presented here. With maximum response times of 0.179 msec and 0.295 msec respectively, the models can actually be applied in critical real-time systems.

Acknowledgements

This work is sponsored by the German National Science Foundation (DFG). Moreover, the first author is grateful to O. Nelles (Dept. of Control Systems Technology) for discussions on local modelling and for providing the turbocharger data. Last but not least, the support given by K. Clarkson whose `hull` program was used for the off-line computation of the Delaunay triangulations, is gratefully acknowledged.

References

[1] T. Ullrich and H. Tolle. Delaunay networks for modelling of non-linear processes. In *IASTED/ISMM International Conference Modelling and Simulation*, pages 319–322, Pittsburgh, USA, April 1996.

[2] T. Johansen and B. Foss. Constructing NARMAX models using ARMAX models. *International Journal of Control*, 58:1125–1153, 1993.

[3] O. Nelles, S. Sinsel, and R. Isermann. Local basis function networks for identification of a turbocharger. In *IEE UKACC Control*, Exeter, UK, 1996.

[4] R. Murray-Smith and K. Hunt. Local model architectures for nonlinear modelling and control. In *ESPRIT PROJECT 8039: NACT*. Daimler-Benz AG/University of Glasgow, 1996.

[5] T. Takagi and M. Sugeno. Fuzzy identification of systems and its application to modeling and control. *IEEE Transactions on Systems, Man and Cybernetics*, (1):116–132, 1985.

[6] E. Ersü and S. Wienand. An associative memory based learning control scheme with PI-controller for SISO nonlinear processes. In *IFAC Symposium on Microcomputer Applications in Process Control*, Istanbul, Turkey, July 1986.

[7] J. S. Jang. Input selection for anfis learning. In *IEEE Conference on Fuzzy Systems*, New Orleans, 1996.

[8] M. Brown, K.M. Bossley, and C. J. Harris. Neurofuzzy algorithms for model identification: Structure and parameter determination. In *IMACS/IEEE-SMC Multiconference CESA - Symp. Control, Optimization and Supervision*, volume 2, pages 1061–1066, Lille, France, 1996.

[9] C. L. Lawson. *Software for C^1 Surface Interpolation*, pages 161–194. Academic Press, 1977.

[10] S. M. Omohundro. The delaunay triangulation and function learning. Technical Report 90-001, International Computer Science Institute, Berkeley, California, 1989.

[11] T. Ullrich. Delaunay networks for modelling of nonlinear processes. *submitted to IASTED International Journal of Modelling and Simulation (Paper No. 1995-151)*.

[12] M. Brown and T. Ullrich. Comparison of node insertion algorithms for delaunay networks. In *Proceedings of IMACS 2nd Mathmod*, pages 775–780, Vienna, 1997.

[13] K. S. Narendra and K. Parthasarathy. Identification and control of dynamical systems using neural networks. *IEEE Transactions on Neural Networks*, 1(1):4–27, 1990.