# Modelling Nonlinear Characteristics using Hierarchical Delaunay-Networks

T. Ullrich and H. Tolle
Darmstadt University of Technology, Landgraf-Georg-Strasse 4, D-64283 Darmstadt
E-Mail: thul@rt.e-technik.th-darmstadt.de

**Abstract**

Real-time control of nonlinear plants requires efficient methods to approximate nonlinear characteristics such as process models or control strategies. Artificial Neural networks can be applied to represent such characteristics but often this approach does not meet constraints of limited computational resources. Automotive control systems are an example of application areas, where real-time processing based on relatively simple hardware is necessary. Delaunay-Networks are an efficient means for real-time function approximation. They consist of a number of interpolation nodes, the Delaunay triangulation of which is used to define finite elements in the input space. Within each finite element, a linear model of the represented function is applied. This approach yields short network response times but a relatively large storage capacity is required for the representation of the triangulation. This is particularly true if more than two variables comprise the network's input space. The paper at hand introduces the concept of *Hierarchical Delaunay-Networks* (HDN), an approach that allows for memory expense to be traded-off against response time. A simplified engine torque model is used to illustrate this method.

## 1 Introduction

When analytical modelling of nonlinear plants is intractable or too time-consuming, systems that *learn* from data are a promising approach. Artificial Neural Networks (ANN) are generally applicable to this kind of problems, however, in application areas where real-time operation on relatively simple hardware is necessary, the implementation of ANNs often is infeasible. *Interpolation networks* which are based on mathematical principles will be shown to be an alternative.

Interpolation networks consist of a set of *interpolation nodes* which are connected to define *finite elements* in the input space. Within each finite element, a simple, e.g. a *linear* model of the represented function is applied. Typically, lattice-like networks with hypercubes as finite elements are used [1]. However, the lattice-structure does not allow to adjust the distribution of the nodes to the local complexity of the underlying function. Thus, these models do not comply with the *principle of parsimony* which demands that the model should represent a given set of data with the fewest possible parameters (nodes). Parsimonious models are computationally inexpensive and yield a high generalisation quality. Overparameterised models, on the other hand, are prone to overfitting, i.e. to modelling measurement noise or the peculiarities of the given samples.

One approach to parsimonious interpolation networks is to allow the nodes to be arbitrarily distributed. The node density can then be adjusted to the local complexity of the represented function. Instead of hypercubes, it is then appropriate to use *simplices* as finite elements. A simplex in the $n$-dimensional input space is defined by $n+1$ nodes each of which comprises a *position* in the input space and an *attribute*, i.e. an estimate of the approximated function's value at the node position. Thus, simplices allow the network response to be computed by local *linear* interpolation.

Queries to a simplex-based interpolation network are processed in a two step procedure. At first, given a *query point* $\mathbf{x}$ the simplex which contains $\mathbf{x}$ is selected (see figure 1 left). The vertices of that simplex are called *active nodes*. These $n+1$ active nodes define a hyperplane which is used to compute the network response (figure 1 right).

The rest of this paper is organized as follows. In section 2 simplex-based networks with simplices obtained form the *Delaunay triangulation* of the nodes are introduced. These *Delaunay-Networks* (DN) have some desirable properties for function approximation. Further information on this approach can be found in [2] and [3]. Section 3 discusses the further development of this method to scalable *Hierarchical Delaunay-Networks* (HDN). The basic idea of HDN is to memorize the Delaunay triangulation of a subset of the nodes only. Queries to the network are processed by first selecting the simplex of the stored triangulation that contains the current query point (1st hierarchical level). Then a local reconstruction of the Delaunay triangulation is carried out in the vicinity of the current query point and a simplex for output interpolation is selected (2nd hierarchical level). This is achieved by applying *topological transitions* [4] and *containment tests* in an iterative manner. HDNs are *scalable*, i.e. their memory

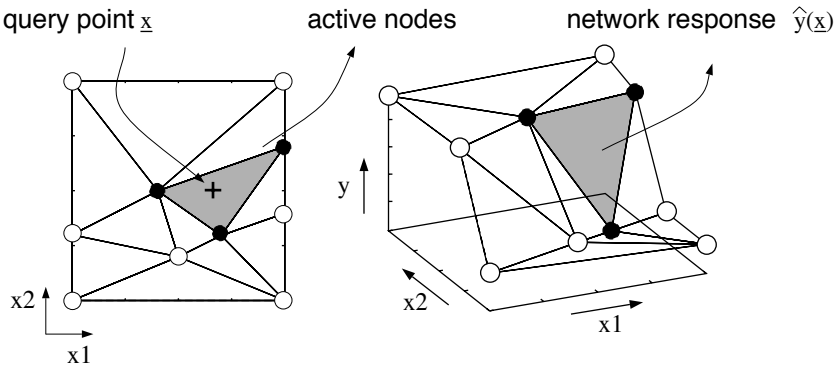



Figure 1: A simplex-based interpolation network with 2D input space.

requirements and response time can be adapted to comply with the limits of a given application:

1. **Trade-off between storage capacity and response time:** The higher the number of nodes that are included in the memorized triangulation is, the lesser topological transitions need to be carried out in the second level, yielding faster network response and higher memory expense.

2. **Trade-off between maximum response time and model accuracy:** The maximum response time can be guaranteed not to exceed a certain limit by prescribing a maximum number of topological transitions *per query* . This means that the network response can be computed from a non-Delaunay simplex in some situations and this leads to a loss of model accuracy temporarily.

In section 4 the HDN approach is illustrated by means of a simplified engine torque model. In this example, the input space comprises only two variables, but the method can also be applied to multi-dimensional modelling problems. A three-dimensional example will be discussed at the conference.

# 2 Delaunay Networks

Given a set of $N$ nodes arbitrarily distributed in the $n$-dimensional input space, the definition of a set of simplices connecting $n+1$ nodes respectively, is not unique. At maximum, it is possible to construct as many as $s_{max} = \binom{N}{n+1}$ simplices. However, those simplices used in an interpolation network have to satisfy two constraints. Firstly, they must be non-intersecting to avoid ambiguity of the network response. Moreover, the union of all simplices must cover the network's entire input space in order to avoid *extrapolation*. This is desirable because extrapolation generally leads to unreliable responses and the implementation of extrapolation routines involves computational overhead.

A certain type of simplices which are particularly useful for function approximation are *Delaunay simplices* [5]. They are characterized by the empty-circle property:

**Definition 2.1** *A simplex $T_i$ consisting of $n+1$ nodes in $\mathbb{R}^n$ is a Delaunay simplex if and only if its embedding $n$-dimensional hypersphere does not contain any other node of the network.*

Figure 2 (left) illustrates this definition for a set of four nodes in $\mathbb{R}^2$. The circumcircles of the triangles in part a contain the fourth node, respectively[1]. The triangles in part b, however, posess the empty circle property and define the *Delaunay triangulation* of the given set of nodes.

Delaunay simplices can be shown to satisfy the two constraints; they are non-intersecting and cover the entire input domain provided that the node distribution is properly chosen[2]. Moreover, Delaunay simplices minimize the worst case approximation error that arises from the local linear modelling. This property is proven in [6], and can be motivated intuitively by means of a one-dimensional example: If a function $\hat{y}(x)$ with bounded second derivative $\hat{y}'' \leq 2 \cdot c$ is to be approximated by a linear model on a finite interval $[a, b]$, the hardest modelling problem is to represent a quadratic function $\hat{y} = c \cdot x^2 + b \cdot x + a$,

[1] In the 2D case, simplices are triangles and their embedding hyperspheres reduce to circumcircles.

[2] The convex hull of the set of nodes must cover the entire input domain and the nodes have to be in general position.

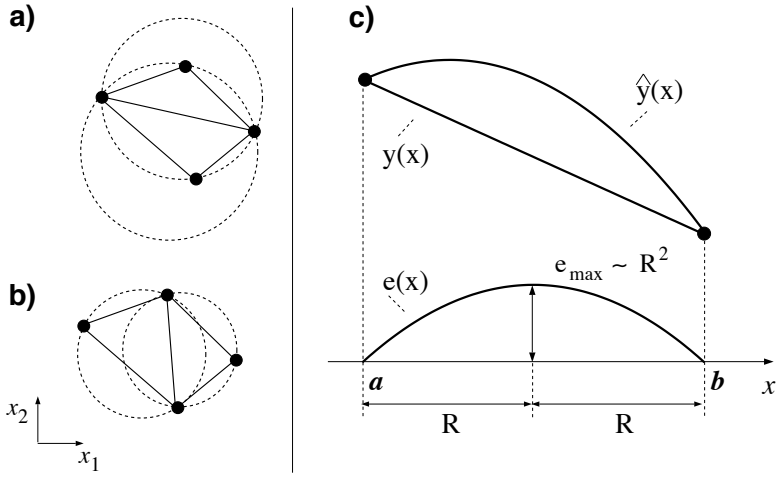



Figure 2: **a) and b)** The Delaunay definition. **c)** Minimization of the worst case approximation error.

since the second derivative is then equal to the maximum of $2 \cdot c$ on the entire interval. In this case the maximum approximation error occurs in the centre of the interval and is proportional to the square of its half width, see figure 2 c. In higher-dimensional cases, the linear model is applied within the convex hull of simplices. The diameter of the circumspheres of these simplices play the role of the interval width then. As a result of the empty circle property, the circumspheres of Delaunay simplices have minimum diameters. Hence, they minimize the maximum approximation error.

## 2.1 Containment Tests

To process a query to a Delaunay network, the simplex the convex hull of which contains the query point $\mathbf{x}$ needs to be determined (see figure 1). This problem can be solved efficiently by means of *Barycentric Coordinates* (BC). By definition [7], the BC of a query point $\mathbf{x}$ with respect to a simplex $T$, are $n+1$ weights which move $T$'s center into $x$. Formally, this is expressed by the set of linear equations

$$C(T) \cdot \mathbf{b} = \mathbf{x}, \tag{1}$$

where $C(T)$ denotes a matrix of the (cartesian) coordinates of $T$'s vertices and $\mathbf{b}$ is the vector of BC, $\mathbf{b} = (b_1, \ldots, b_{n+1})^T$. An important property of the BC is, that their signs describe on which side of $T$'s edges the query point $\mathbf{x}$ is located. If $\mathbf{x}$ is located inside $T$'s convex hull, $x \in CH(T)$, all $n+1$ BC are positive, whereas if $\mathbf{x}$ is located beyond one of $T$'s edges[3], the respective BC is negative. BC provide an efficient means for selecting the active nodes, since they can be computed easily for low-dimensional problems by calculating determinants (i.e. solving (1) by applying Cramer's rule).

# 3 Hierarchical Delaunay Networks

The proposed methods allow the implementation of interpolation networks which yield very short response times; results obtained on a T805 processor were reported in [3]. However, the disadvantages of DN are the relatively large strorage capacity needed for the representation of the triangulation and the fact, that inserting or deleting nodes is a computationally complex task[4]. Node insertion and deletion is essential, however, when a DN is constructed automatically on the basis of a set of training data.

*Hierarchical Delaunay Networks* (HDN) cope with these difficulties by memorizing the triangulation for a subset of $\widehat{N} \leq N$ nodes only. Hence, memory requirements can be reduced. The Delaunay triangulation for this subset is computed off-line. Additional nodes can be inserted and deleted again without the need to update the topological database. This results in a considerable speed-up of network construction procedures.

[3] In the $n$-dimensional case, the edges are $(n-1)$-dimensional planes.

[4] If an additional node is inserted, some simplices are likely to loose the empty-circle property. In case of node deletion, those simplices which had the deleted node as a vertex, are destroyed. In either case, the triangulation needs to be recomputed and the database needs to be updated.

### 3.1 Topological Transitions

To compute the response of a HDN, it is necessary to reconstruct the Delaunay triangulation locally, i.e. in the vicinity of the current query point $\mathbf{x}$. Starting with that simplex of the memorized triangulation that contains $\mathbf{x}$, the triangulation is iteratively refined under consideration of all $N$ nodes. These refinements are carried out by applying *topological transitions* [4] and *containment tests* (see section 2.1).

Given the simplex $T$ that contains the current query point $\mathbf{x}$, the node list is searched for a node inside the circumsphere of $T$. If no such node can be found, $T$ is a global Delaunay simplex and can be used for computing the network response. Otherwise, the local Delaunay triangulation for the set of $(n+2)$ nodes[5] is computed. From $n+2$ nodes, $\binom{n+2}{n+1} = n+2$ simplices can be constructed. One of theses candidates is identical to $T$ and hence needs no further investigation. The remaining $n+1$ simplices $T_i$ are defined as follows:

$$T_i = T \backslash \{p_i\} \cup \{p\}\,. \tag{2}$$

Here, $p$ denotes the node detected in $T$'s circumsphere, $p_i, \quad i = 1, \ldots, n+1$ are the vertices of $T$. The $i$-th candidate simplex $T_i$ is obtained if the $i$-th vertex of $T$ is replaced by $p$. The task to be solved now, is to determine that candidate $T_j$ with the following properties:

1. $T_j$ is a *local* Delaunay simplex, i.e. its circumsphere does not contain the remaining node.
2. $T_j$ contains the query point $\mathbf{x}$.

The first requirement is met, if the $i$-th barycentric coordinate of the location of the node $p$ with respect to the simplex $T$ is positive. This result is quoted here without proof, the reader is referred to [8] for a comprehensive discussion. The second criterion can be checked by the containment test explained in section 2.1. After the candidate $T_j$ which posesses both properties, is found, the procedure is repeated with $T_j$ taking $T$'s place. This iteration is continued until one of the following conditions hold true:

1. $T$ is a global Delaunay simplex, i.e. there is no node within its circumsphere.
2. A prescribed maximum number of $t_{max}$ iterations has been performed.

In either case, the currently existing simplex $T$ is used for output interpolation. It can be guaranteed that $T$ contains the query point $\mathbf{x}$. If the procedure was stopped because of the prescribed maximum number of iterations, $T$ might not be a global Delaunay simplex; in section 4 the impact of the limit $t_{max}$ on the worst case response time as well as on the model accuracy is investigated. Figure 3 summarizes the algorithm.

```
Find node p∈CS(T)
if (no such node exists) OR (max no. of iterations reached)
  compute network response using simplex T
else
    for i=1, ..., n+1
      construct candidate T_i according to eq. (2)
      if (p_i∉CS(T_i)) AND (x∈CH(T_i))
        set T = T_i and start from beginning
```

**Symbols:**

| | |
|---|---|
| **x** | query point |
| T | simplex containing **x** |
| CH(T) | convex hull of T |
| CS(T) | circumsphere of T |

Figure 3: Algorithm for the iterative local reconstruction of the Delaunay triangulation. The procedure starts with a simplex $T$ which is part of the memorized triangulation and contains the query point $\mathbf{x}$.

### 3.2 Selection of Nodes for Off-line Triangulation

The HDN approach requires the selection of an appropriate subset of $\widehat{N} \leq N$ nodes and the computation of the Delaunay triangulation for this subset. $\widehat{N}$ serves as a trade-off parameter for response time and memory expense. Thus, it has to be determined according to the application-specific requirements and resources. After specifying *how many* nodes are included in the memorized triangulation, it is necessary

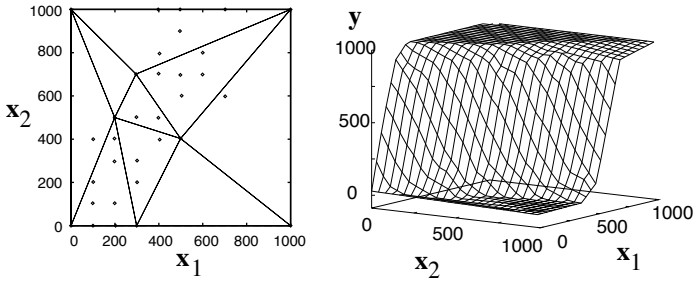
[5] n+1 nodes are the vertices of $T$, the additional node is located inside $T$'s circumsphere.

to determine *which* nodes should be included. An appropriate selection can be done automatically, if the fact is considered that incremental construction techniques typically lead to *fractal growth* of the network: The node distribution after a few node insertions qualitatively is similar to that of the final network. Hence, it is recommended to select the *first* $\widehat{N}$ nodes that were inserted into the network during its construction[6]. This choice assures that the circumspheres of all simplices of the memorized triangulation contain approximately the same number of nodes and therefore the number of topological transitions will be relatively independent from the positions of the query points.

# 4 Results

To illustrate the approach and to investigate its performance, we constructed a HDN model of the nonlinear bivariate function

$$y(x_1, x_2) = 1000 \cdot \left[1 + e^{(2 - \frac{1}{50}x_1 + \frac{1}{100}x_2)}\right]^{-1}. \tag{3}$$




This equation simulates the steady-state torque characteristics of a combustion engine as a function of the throttle valve position and the engine speed. The figure illustrates a HDN model of this function with $N = 32$ nodes. In the depicted example, $\widehat{N} = 8$ nodes are part of the memorized triangulation. The right diagram shows the interpolated surface, i.e. the model's input/output relationship.

In figure 4 the impact of the number of nodes $\widehat{N}$ which are included in the memorized triangulation on the memory expense and response time is illustrated. The more nodes are included, the higher is the required storage capacity and the lower are the network's response times, since fewer topological transitions need to be carried out. The timings were obtained on a T805 microprocessor running at 30 MHz. The query points used in this simulation were derived from real-world signals, measured in a test vehicle. Apparently, there is a large difference between the *average* and the *maximum* response time. This is due to the fact, that the query point moves slowly in stationary operating modes. Once a global Delaunay simplex is reconstructed, it can be used in successive query cycles. However, in transient conditions, the Delaunay triangulation needs to be reconstructed in another area of the input space. This requires a large number of topological transitions and leads to the maximum response time.

In the simulations discussed so far, the number of topological transitions was not limited, i.e. the local reconstruction algorithm came up with a global Delaunay simplex for each query point. It is worthwile investigating, how a limited number of topological transitions per query affects the network's performance (see figure 4). The maximum response time decreases linearly with the reduced number $t_{max}$ of topological transitions. In some situations, the network response is computed using a non-Delaunay simplex then. Since Delaunay simplices yield the minimum worst case approximation error, the network responses obtained with a certain limit $t_{max}$ are compared to the responses with unbounded $t_{max}$. The lower diagram in figure 4 (right) shows the *maximum absolute* deviation. It is important to note, that this maximum error occurs only *temporarily*, since the iterative refinement of the simplices is continued in successive query cycles.

Figure 4 (right) suggests that $t_{max}$ can be set to approximately 75% of the value needed for reconstructing global Delaunay simplices in every query cycle without significant loss of accuracy. In the example discussed here, at maximum 13 transitions are necessary; setting $t_{max}$ to 10 yields only minor errors but reduces the maximum response time by 229 $\mu$sec.

# 5 Summary

Finite element based interpolation networks are a computationally efficient alternative to artificial neural networks for real-time modelling and control. When simplices are used as finite elements, the node

[6] Algorithms for data-driven construction of Delaunay networks are discussed in the accompanying paper by Brown and Ullrich.

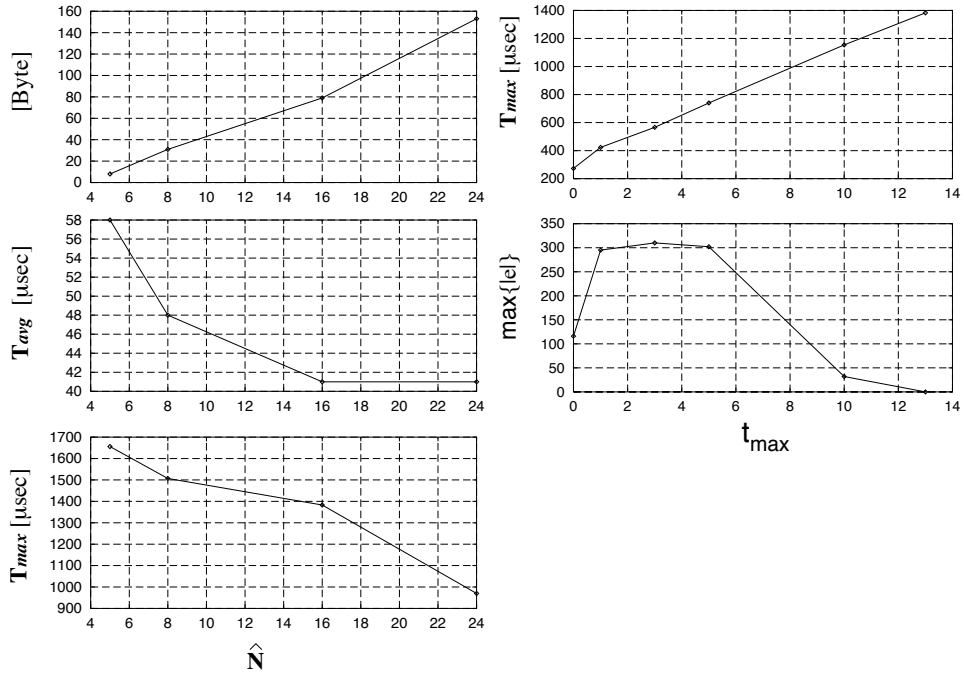



Figure 4: **Left**: Impact of the parameter $\hat{N}$ on memory expense and average ($T_{avg}$) and maximum ($T_{max}$) response time. **Right**: Impact of the parameter $t_{max}$ on the maximum response time and model accuracy.

distribution can be adjusted to the local complexity of the underlying function. Hence, parsimonious models can be built. The Delaunay triangulation is the appropriate structure for such networks since it minimizes the worst case approximation error. The further development of this concept to hierarchical Delaunay networks was described in the paper. The HDN approach allows for memory expense to be traded-off against the network's response time and thus the model can be adapted to meet application-specific constraints.

## Acknowledgements

This work is sponsored by the German National Science Foundation (DFG) under grant To 75/22-1. Furthermore, the author is grateful to Dr. T. Roos, Federal Institute of Technology at Zurich, for the helpful discussions on computational geometry.

# References

[1] M. Brown and C. Harris. *Neurofuzzy Adaptive Modelling and Control*. Prentice Hall, Hemel-Hempstead, 1994.

[2] S. M. Omohundro. Geometric learning algorithmns. Technical Report 89-041, International Computer Science Institute, Berkeley, California, USA, 1989.

[3] 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.

[4] L. Guibas, J. Mitchell, and T. Roos. Voronoi diagrams of moving points in the plane. In *17th International Workshop on Graphtheoretic*, pages 113–125, Fischbachau, Germany, 1991.

[5] A. K. Cline and R. L. Renka. A storage-efficient method for construction of a thiessen triangulation. *Rocky Mountain Journal of Mathematics*, 14:119–139, 1984.

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

[7] I. N. Bronstein and K. A. Semendajew. *Taschenbuch der Mathematik*. Verlag Harri Deutsch, 21 edition, 1982.

[8] G. Albers, J. S. B. Mitchell, L. J. Guibas, and T. Roos. Voronoi diagrams of moving points. *International Journal of Computational Geometry & Applications*, 1996.