Hammerstein system identification through support vector machines and method of splines

Veenema, T.M.

Award date:
2008

Eindhoven University of Technology
Department of Electrical Engineering
Control Systems

Hammerstein System Identification through Support Vector Machines and Method of Splines

by

T.M. Veenema

Master of Science thesis

Project period: June 2008
Report Number: 08A/04

Commissioned by:
Prof.dr.ir. A.C.M.P. Backx
Supervisor:
Dr. S. Weiland

Additional Committee members:
Dr. L. Ozkan (IPCOS)
Dr. V.M. Mladenov (TU/e)
Dr. J. Ludlage (IPCOS)

Hammerstein System Identification through Support Vector Machines and Method of Splines

Thijs Veenema

***Abstract*— A model, or a mathematical description of reality, which provides an approximation of a some more complex phenomenon, may be broadly classified as deterministic and probabilistic models. One of the model classes which is a subset of the probabilistic models is a Hammerstein system, which is a serial interconnection of a static nonlinear mapping with a linear dynamic system. This paper will describe two possible identification techniques for determination of the memoryless static nonlineatity followed by the estimation of the linear dynamic system parameters, where the interconnection is used to construct a Hammerstein system. The first part of this paper deals with identification and modeling of static neural network, which are used for the nonlinear mapping. The second part deals with identification of a nonlinear dynamic system, parameterized with the nonlinear neural network structures. The behavior of the different techniques will be showed in the last part, where a glass melting furnace will be modeled by means of identifying the fourier coefficients of a reduced order Proper Orthogonal Decomposed (POD) dynamic system.**

I. INTRODUCTION

Nonlinear systems have been shown to exhibit surprising and complex effects that can never occur in linear systems. Prominent examples of these include bifurcations, chaos, and solitons. A system (static or dynamic) can be seen as a mathematical object in which variables of different kinds interact and produce observable output signals that are the result of controllable user determined settings (like statistical properties of the input signal) and non-controllable external input signals. A dynamic system consist of an n-dimensional state space $\mathbb{R}^n$ where an element in the state space $x_0 \in \mathbb{R}^n$ at time instant t_0 will follow a certain trajectory for $t > t_0$, which is determined by the dynamic equations together with the inputs. Once there is interaction with a system, a description is needed of how its variables relate to each other. In system identification, one is interested in building a mathematical model of a dynamic system, based on observed data.

One of the approaches to control a nonlinear dynamic system is to linearize the nonlinear system in a certain working point, and to apply a linear controller around this working point. This technique restricts the system to work in a certain local operating range. If it is desirable that the system also operates accurate enough in an other working point, a possible solution would be to derive multiple linearized models, and switch between the different models if the working point will change. If the time of switching between two operation points will not be accurate enough, the system may become unstable.

In order to describe the behavior of a complete operating range of the nonlinear system, nonlinear system identification techniques have to be applied. One of the nonlinear model classes which will be described in this paper is the Hammerstein model, which consists of an interconnection of a static function f and a dynamic linear system H, which describes the dynamics of the model (see Fig. 1). Many techniques have been proposed for the identification of Hammerstein systems, which mainly differ in the way the static nonlinearity is parameterized and in the way the optimization problem is formulated. This paper contributes in the identification of NL systems, where the static nonlinearity will have the form of a neural network, with different basis functions, which results in different behavior of the NL system. The results will be applied to the identification of a nonlinear model of a 2D glass furnace.

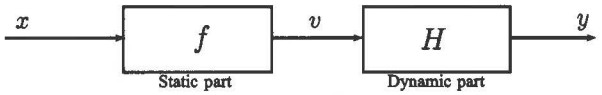


Fig. 1. General Hammerstein structure

II. LEARNING STATIC MODELS

Consider a static MISO (Multiple Input, Single Output) nonlinear mapping $x \rightarrow v$, where $x \in \mathcal{X} = \mathbb{R}^d$ and $v \in \mathcal{V} = \mathbb{R}$. The nonlinear function f can be learned from a measured input/output training set of examples $S_N = \{(x_n, v_n)\}_{n=1}^N \subseteq \mathbb{R}^d \times \mathbb{R}$, where each deterministic input sample x_n is related by a probabilistic relationship, because generally an element of $\mathcal{X}$ does not determine uniquely an element of $\mathcal{V}$, but rather a probability distribution $P(x, v)$ on $\mathcal{V}$, defined over the set $\mathcal{X} \times \mathcal{V}$. The problem of learning consist in providing an estimator $f : \mathcal{X} \rightarrow \mathcal{V}$. One way to solve this learning problem is to define a risk functional which measures the average amount of error (expected risk) associated with the function f and search for an estimator f_0 that minimizes the expected risk

$$I[f] = \int_{\mathcal{X},\mathcal{V}} V(v, f(x)) P(x, v) dx dv$$

where $V(v, f(x))$ is the so called loss function measuring the error between the measured output v and the estimator f. The best estimator (target function) f_0 can be found by searching in a function space $\mathcal{F}$ of functions $\mathcal{X} \rightarrow \mathcal{V}$ such that it minimizes the expected risk:

$$f_0 = \arg\min_{f \in \mathcal{F}} I[f]$$

Because in practice only a finite number of samples is available and the probability distribution $P(x, v)$ is unknown, an empirical risk minimization is carried out, which consist of using the dataset S_N to build a (stochastic) approximation of the risk, usually called the empirical risk, and is defined as

$$I_{\text{emp}}[f; N] = \frac{1}{N} \sum_{i=1}^{N} V(v_i, f(x_i))$$

If one is searching in a function space $\mathcal{F}$ for an estimator f_0 based on the empirical risk, one can find a subset $\mathcal{F}_{\text{emp}} \subset \mathcal{F}$ (which is the set of interpolating functions) where the empirical risk is zero: $\mathcal{F}_{\text{emp}} = \{f \in \mathcal{F} | I_{\text{emp}}[f; N] = 0\}$. The problem now is that it is possible that f_0 interpolates the data points, but has large variation on the non-sample points $x \neq x_i, 1 = 1, \ldots, N$, so the empirical risk at (x_i, v_i) is zero while f_0 does not match to the relation $v = f(x)$. This will become clear in Fig. 2, where the dashed line f is the function that has to be identified from the input/output data set $S_4 = \{(x_n, v_n)\}_{n=1}^{4}$. The empirical risk $I_{emp}[f_0, 4] = 0$, but the function f_0 does not match the function f. In order to

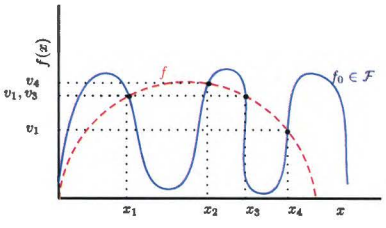


Fig. 2. Example poor generalization

avoid this problem, the regression technique will be extended with regularization. For a finite set of training samples, the search for the best model approximation function has to be constrained to an approximately "small" hypothesis space, which can also be thought of as a space of models or network architectures. If this space is too large, models can be found which will fit exactly the data samples, but will have a poor interpolation performance, so will have a poor predictive capability for new unseen data. Searching in a "small" space for the optimal function f_0 can be formulated by minimizing

$$H[f] = \frac{1}{N} \sum_{i=1}^{N} V(v_i, f(x_i)) + \lambda \|f\|^2$$

Where $V(\cdot, \cdot)$ is again the loss function and $\lambda \geq 0$ a regularization constant.

In this paper, the nonlinear function approximation f will be represented as a linear combination of weighted nonlinear basis functions, which can be written as a neural network structure:

$$v = f(x) = \sum_{i=1}^{M} \theta_i^{(l)} \Phi(x, \theta_i^{(nl)}) \tag{1}$$

The basis functions $\Phi : \mathcal{X} \times \Theta^{(nl)} \rightarrow \mathcal{V}$ are weighted with the linear parameters $\theta_i^{(l)} \in \Theta^{(l)}$, where the basis functions depend on the input x, and a vector of nonlinear parameters gathered in $\theta_i^{(nl)} \in \Theta^{(nl)}$. In order to realize a nonlinear mapping, the basis functions have to be nonlinear. There is a distinction between two different basis functions:

- Global basis functions: Contribute to the complete model output range. Global behavior exist if a change in the associated linear parameter $\theta_i^{(l)}$ of the basis function formulation significantly influences the model output over a wide operating regime.
- Local basis functions: Contribute only to a subset of the model output range locally. Local behavior exist if a change in the associated linear parameter $\theta_i^{(l)}$ of the basis function formulation significantly influences the model output over a small operating regime.

An important advantage of this kind of models, is that the model is linear in its weighting parameters θ_i^l. If the basis functions are fully specified, the linear parameters can be estimated by linear optimization. An illustrating example will be polynomial interpolation where $\Phi_i(x) := x^{i-1}$. If a data set $S_N = \{(x_i, v_i)\}_{i=1}^{N} \subset \mathbb{R} \times \mathbb{R}$ is available, one wants to achieve a optimal vector $\hat{\theta}^{(l)} = [\hat{\theta}_1, \ldots, \hat{\theta}_M]^T$: such that $\|X\hat{\theta}^{(l)} - V\|$ is minimized, where

$$X = \begin{bmatrix} \Phi_1(x(1)) & \ldots & \Phi_M(x(1)) \\ \Phi_1(x(2)) & \ldots & \Phi_M(x(2)) \\ \vdots & \vdots & \vdots \\ \Phi_1(x(N)) & \ldots & \Phi_M(x(N)) \end{bmatrix} \tag{2}$$

and $V = [v_1 \; v_2 \; \ldots \; v_N]^T$. this results in an least squares solution $\hat{\theta}^{(l)} = \left(X^T X\right)^{-1} X^T V$, which best describes the mapping $\mathcal{X} \rightarrow \mathcal{V}$, so the solution is a linear combination of nonlinear basis functions.

III. Subspace Regression in Reproducing Kernel Hilbert Space

The benefit of learning static input/output relations between two spaces $\mathcal{X}$ and $\mathcal{V}$ (function approximation) in a Reproducing Kernel Hilbert Space H (RKHS, which is a Hilbert space in which all the point evaluations are bounded linear functionals), is that any kind of nonlinearity can be learned, while avoiding to solve a nonlinear optimization problem. This technique makes use of least squares regression models in the kernel context. The main idea is to change the representation of a data point into a higher dimensional mapping in a RKHS by means of a kernel function where the number of regression parameters can be controlled by projection of the RKHS to a lower dimensional subspace. Here, it is assumed that the unknown nonlinear function f belongs to the RKHS H that consist of functions $\mathcal{X} \rightarrow \mathcal{V}$.

In a reproducing kernel space H, there exist a positive definite kernel $k : \mathcal{X} \times \mathcal{X} \rightarrow \mathcal{V} : (x_1, x_2) \mapsto k(x_1, x_2)$, which provides a similarity measure between pairs of data points, and a dot product $\langle \cdot, \cdot \rangle_H$ such that

1) $\forall f \in H, \forall x \in \mathcal{X}, k$ has the reproducing property

$$\langle f, k(x, \cdot) \rangle_H = f(x), \text{ for } x \in \mathcal{X} \tag{3}$$

2) the inner product $\langle \cdot, \cdot \rangle_H$ induces a norm on $f \in H$: $\|f\|_H := \langle f, f \rangle_H^{\frac{1}{2}}$.

3) H is the closure of the span of all $k(x,\cdot)$ with $x \in \mathcal{X}$.

For a certain positive definite kernel function k, each sample $x \in \mathcal{X}$ is mapped by a reproducing kernel map $\varphi : \mathcal{X} \to H : x \mapsto k(x,\cdot)$, which can be evaluated at $x' \in \mathcal{X}$ to give $\varphi(x)(x')$. That is, $\varphi(x)(x') = k(x,x')$ for short. Now define $\varphi : \mathcal{X} \to H$ by $\varphi(x) = k(x,\cdot)$. So φ denotes a mapping from the input space $\mathcal{X}$ to the Hilbert space H (also known as feature space). The kernel k denotes the inner product between two elements x and x':

$$\varphi(x)(x') = \langle \varphi(x), \varphi(x') \rangle = \langle k(x,\cdot), k(x',\cdot) \rangle = k(x,x').$$

For example, the Gaussian kernel $k : \mathcal{X} \times \mathcal{X} \to \mathcal{V}$ is defined as $k(x,x') = e^{-\|x-x'\|_2^2/\sigma}$, which maps to a Gaussian centered at point x', where one can evaluate this Gaussian at point x. The RKHS can now be defined by construction of a vector space of linear combinations of the reproducing positive definite kernel:

$$f(\cdot) = \sum_{i=1}^{N} \theta_i k(\cdot, x_i) \tag{4}$$

So $\mathcal{F}$ consist of functions $f(x) = \langle f, k(x,\cdot) \rangle$, where f has the property that every evaluation operator and norm of any element in H is bounded. This makes the elements of a RKHS well-suited to interpolate smooth pointwise known functions. To find such an interpolating function $f \in H$ in the context of regularization, one can minimize a pointwise cost function V over data and monotonic smoothness function g

$$\sum_{i=1}^{N} V(v_i, f(x_i)) + g(\|f\|) \tag{5}$$

where V is convex in f and f has the representation 4. The general form includes unpenalized (low-dimensional) subspaces, which will differ for different regularization functions g. The solution of f is constrained to the subspace that is spanned by the mapped datapoints:

$$f(x) = \sum_{i=1}^{N} \theta_i \varphi(x_i)(x) = \sum_{i=1}^{N} \theta_i k(x, x_i) \tag{6}$$

According to Mercer's theorem, which is a representation of a symmetric positive-definite function on a square as a sum of a convergent sequence of product functions, there exist for each positive definite kernel an orthogonal set $\{\phi_i\}_{i=1}^{q}$ with non-negative eigenvalues $\lambda_i, i = 1,\ldots,q$ such that one defines the following spectral decomposition:

$$k(x,\cdot) = \sum_{i=1}^{q} \lambda_i \phi_i(x) \phi_i(\cdot)$$

With help of the reproducing property, one can derive that

$$f(x) = \langle f, k(\cdot,x) \rangle = \left\langle \sum_{n=1}^{q} c_n \phi_n(\cdot), \sum_{n=1}^{q} \lambda_n \phi_n(\cdot) \phi_n(x) \right\rangle$$
$$= \sum_{n=1}^{q} c_n \phi(x), \text{ where the inner product } \langle \phi_i, \phi_j \rangle = \frac{1}{\lambda_i} \delta_{i,j}$$

The basis functions $\phi_i, i = 1,\ldots,q$ can be scaled by a factor $\sqrt{\lambda_i}$ to make the orthogonal eigenvectors orthonormal. The mapping $\varphi(x_i), i = 1,\ldots,N$ can now be replaced by designing a $n \times q$ feature matrix Φ_q

$$\Phi_q = \begin{bmatrix} \sqrt{\lambda_1}\phi_1(x_1) & \sqrt{\lambda_2}\phi_2(x_1) & \ldots & \sqrt{\lambda_q}\phi_q(x_1) \\ \sqrt{\lambda_1}\phi_1(x_2) & \sqrt{\lambda_2}\phi_2(x_2) & \ldots & \sqrt{\lambda_q}\phi_q(x_2) \\ \vdots & \vdots & \ddots & \vdots \\ \sqrt{\lambda_1}\phi_1(x_N) & \sqrt{\lambda_2}\phi_2(x_N) & \ldots & \sqrt{\lambda_q}\phi_q(x_N) \end{bmatrix} \tag{7}$$

The goal now is to identify a function f that best matches the mapping $\mathcal{X} \to \mathcal{V}$ by applying linear regression of the data set $S_N = \{(\varphi(x_n), v_n)\}_{n=1}^{N}$ by searching for a solution

$$\min_{\theta} \|V - \Phi_q \theta\| \tag{8}$$

Where $V = [v_1,\ldots,v_N]^T$ is the observed datamatrix, and Φ_q the $N \times q$ feature matrix containing the q scaled basis vectors evaluated at x_i for $i = 1,\ldots,N$, and θ the unknown regression coefficient matrix.

Because the mapping φ maps the input space $\mathcal{X}$ to a much higher dimensional feature space H, which is spanned up by the mapped data points and is in general larger compared to the number of data samples N, it is hard to find a solution for θ. In order to restrict the infinite number of regression coefficients, one can introduce a projection matrix ϑ in order to project the infinite dimensional featurespace H to a subspace of finite dimension m. One can express the projection by gathering the basis vectors $\{v_i\}_{i=1}^{m}$ of the subspace in the columns of a $q \times m$ transformation matrix ϑ. The subspace of transformed regressors $Z = \Phi_q \vartheta$ is obtained by projecting the elements of the feature matrix to the new coordinates $z_i = \phi(x_i)^T v_i$, but this projection to a subspace can never be obtained due to the fact that the mapping φ to a higher dimensional featurespace H is not known. This problem can be avoided by making use of the so called kernel trick, which is based on the fact that the matrix Φ_q is unknown, but is is possible to obtain the positive semidefinite matrix $K = \Phi_q^T \Phi_q$. This can be obtained by decomposing the matrix $\vartheta = \Phi_q^T A$. So the columns of Φ_q^T are transformed by the transformation matrix A such that the original matrix ϑ will return. By decomposing the matrix ϑ, one is allwed to write $Z = \Phi_q \vartheta = \Phi_q \Phi_q^T A = KA$, and thus the model becomes

$$V_{obs} = (KA)\theta + E = K(A\theta) + E$$
$$\text{So } v(k) = \sum_{i=1}^{N} \theta_i k(x_i, x_k) + e(k)$$

where $E = [e_1,\ldots,e_N]^T$ denotes the modeling error, $K_{ij} = k(x_i,x_j) | \, i,j = 1,\ldots,N$ the kernel function and $\theta_i, i = 1,\ldots N$ the linear parameters which identify the nonlinear function f.

For example, take as Hilbert space the set of functions of the form $f(x) = \sum_{n=1}^{\infty} c_n \phi_n(x)$ in which the scalar product is defined as

$$\left\langle \sum_{n=1}^{\infty} c_n \phi_n(x), \sum_{n=1}^{\infty} c_n \phi_n(x) \right\rangle = \sum_{n=1}^{\infty} \frac{c_n^2}{\lambda_n}$$

where $\langle \phi_i, \phi_j \rangle = \delta_{i,j} \frac{1}{\lambda_i}$. If k is a Gaussian radial Kernel $k : \mathcal{X} \times \mathcal{X} \rightarrow \mathcal{V}$, with $k(x_1, x_2) = e^{-\frac{\|x_1 - x_2\|}{\sigma}}$, consider a one-dimensional case in which $x \in \mathcal{X} = [0, 2\pi]$, and $\phi_n(x) = e^{jnx}$, and c_n are the Fourier coefficients of the function f. f can be approximated by minimizing

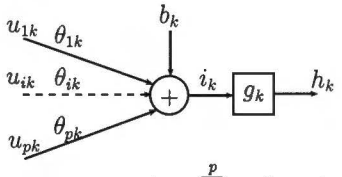
$$\min_{f \in \mathcal{H}} H[f] = \sum_{i=1}^{N} V(v_i, f(x_i)) + \lambda \frac{1}{2} \|f\|^2$$

where $\|f\|^2 = \sum_{n=1}^{\infty} \frac{c_n^2}{\lambda_n}$ and $\lambda \geq 0$. In this way, $\|f\|^2$ limits the amplitude of c_n, where small amounts of $\|f\|^2$ contributes to limited high-frequency contents, and therefor do not oscillate much at inter sample points so that $\|f\|^2$ is a measure of smoothness.

IV. Construction Mechanisms

In this section, two different construction mechanisms will be introduced, which are able to identify a nonlinear static function f. The two construction mechanisms are both family of the Neural Network structure.

A single Neuron k consists of a linear combination of p input signals $u_{1k}, \ldots, u_{pk}$ weighted by $\theta_{1k}, \ldots, \theta_{pk}$ (see Fig. 3) and a certain offset b_k, such that $i_k = \sum_{i=1}^{p} \theta_{ik} u_{ik} + b_k$ which will be evaluated by an activation function g_k. The output of neuron k is h_k and can be evaluated by $h_k = g_k(i_k)$. The single neurons can be connected to each other such that there exist different construction mechanisms. If P different network layers are placed in series, where a network layer consist of a parallel interconnection of m neurons k, different construction mechanisms can be derived.



$$i_k = \sum_{i=1}^{p} u_{ik} \theta_{ik} + b_k$$
$$h_k = g_k(i_k)$$

Fig. 3. Single Neuron

A static nonlinear function can be parameterized by a neural network, which is characterized by:

- Neural Network structure: The pattern of connections between the neurons
- the weights θ_{ik} and the offset values b_k
- activation function g_k

In this section, two construction mechanisms will be introduced:

1) Radial construction
2) Tensor Product Construction

which can be defined as a neural network, as illustrated in Fig. 4. The input $x \in \mathbb{R}^d$ in this construction enters the neurons un-weighted, where each of the m neurons have the same input x. The main difference of the two constructions is the difference in the activation function Φ (which is the same for all neurons $1, \ldots, m$), the values $c_i \in \mathbb{R}^d, i = 1, \ldots m$, and b, which will become clear in the next two subsections. In the output layer, the activation function g is chosen as a throughput, so $g(x) = x$.

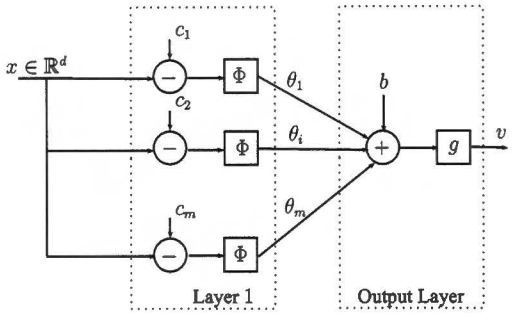


Fig. 4. Radial and Tensor Product Construction

A. Radial construction: Radial Basis Function (RBF) Networks

The radial construction is utilized for radial basis function networks, where the input $x \in \mathbb{R}^d$ is centered around m different centers $c_i \in \mathbb{R}^d, i = 1, \ldots, m$. The radial construction can be transformed to a Radial Basis Function (RBF) Network. Hereby, the nonlinear basis function Φ (as in Fig. 4) will be superposed by a kernel function

$$\Phi(x - c_i) = k(x, c_i) = e^{-\frac{\|x - c_i\|_2^2}{\sigma}}$$

where σ is the width of the Gaussian function. The network structure will be a linear combination of the m basis functions Φ (with different centers c_i). The nonlinear function $f : \mathbb{R}^d \rightarrow \mathbb{R}$ can now be written as

$$v = f(x) = \sum_{i=1}^{m} \theta_i k(x, c_i) + b \tag{9}$$

One of the difficulties is to find the optimum number of neurons (m), in order to match the identified nonlinear function f, so one is looking for a set of center points $c = [c_1, \ldots, c_m]$ for an unknown number m. One way to find the unknown parameter m and the set of centers c, is to make use of support vector machine's (SVM), which is a set of related supervised learning methods used for classification and regression, and belong to a family of generalized linear classifiers. Once again, the problem is to learn a functional relation between sampled input and output data $S_N = \{x_n, v_n\}_{n=1}^{N}$ by minimizing $H[f] = \frac{1}{N} \sum_{i=1}^{N} V(v_i, f(x_i)) + \lambda \|f\|^2$, where

$$V(v_i, f(x_i)) = |v_i - f(x_i)|_\epsilon = \begin{cases} 0 & \text{if } |v_i - f(x_i)| < \epsilon \\ |v_i - f(x_i)| - \epsilon & \text{otherwise} \end{cases}$$

which is called an ϵ-Insensitive Loss function, also denoted as V_ϵ. The V_ϵ loss function is used to allow functions f that lie between a 2ϵ bound from the real function f, which is

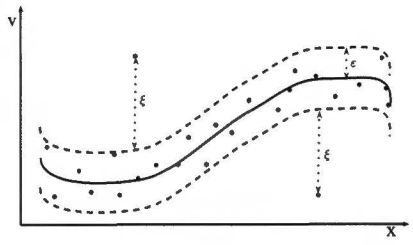


Fig. 5. 2ϵ bound

the solid line in Fig. 5. The parameter ϵ can be seen as the resolution at which one wants to look to the data, so the larger ϵ, the simpler the representation will be. This problem can be formulated by writing the following optimization problem:

$$\min_{f,\xi,\xi^*} \left[\frac{C}{N} \sum_{i=1}^{N} (\xi_i + \xi_i^*) + \frac{1}{2} \|f\|^2 \right]$$

$$\text{s.t.} \quad \left. \begin{array}{r} f(x_i) - v_i \leq \epsilon + \xi_i \\ v_i - f(x_i) \leq \epsilon + \xi_i^* \\ \xi_i, \xi_i^* \geq 0 \end{array} \right\} i = 1, \ldots, N$$

where $f(x) = \sum_{n=1}^{\infty} c_n \phi_n(x) + b$ and $C = \frac{1}{2\lambda}$ is a scaling constant, which can be seen as the regularization constant which will control the trade off between model complexity and model performance. Note that the penalty C is only paid when the absolute value of the error exceeds ϵ. The variables $\xi_i, \xi_i^*, i = 1, \ldots, N$ are the so called slack variables, which become active ($\xi, \xi^* \geq 0$) if a sample lies outside the 2ϵ bound (this explains the name Support Vector in SVM). It is possible to solve this QP problem numerical, but it will give problems if the size N of the learning set S_N grows. To avoid this problem, and to write the solution of optimal parameters in a mathematical expression, one can change the cost function V to a least squares problem, such that $V(v, f(x)) = (v - f(x))^2$.

Consider a regression model $v_t = f(x_t) + e_t$ where $f : \mathbb{R}^d \to \mathbb{R}$ is an unknown real valued smooth function which has to be derived and $e_1, \ldots e_N$ are uncorrelated random errors with $E[e_t] = 0$ and $E[e_t^2] = \sigma_e^2 < \infty$
The following model is assumed:

$$f(x) = \sum_{n=1}^{q} \theta_i \phi_i(x) + b = w^T \varphi(x) + b$$
$$w = [\theta_1 \ \theta_2 \ \ldots \ \theta_q]^T$$
$$\varphi(x) = [\phi_1(x) \ \phi_2(x) \ \ldots \ \phi_q]^T$$

So the input is first transformed into the feature space H, and by linear regression, the function f is obtained. Given S_N, consider the cost function ζ as function of w and e, given as:

$$\begin{array}{lrcl} \min\limits_{w,b,e} & \zeta(w,e) & = & \frac{1}{2} w^T w + \frac{\gamma}{2} \sum\limits_{t=1}^{N} e_t^2 \\ \text{s.t.} & v_t & = & w^T \varphi(x_t) + b + e_t, \quad t = 1, \ldots, N \end{array}$$

where $\gamma \in \mathbb{R}_0^+$ is again the regularization constant. To solve the constrained optimization problem, a Lagrangian is constructed:

$$\mathcal{L}(w, b, e; \alpha) = \zeta(w, e) - \sum_{t=1}^{N} \alpha_t (w^T \varphi(x_t) + b + e_t - v_t)$$

$$\frac{\partial \mathcal{L}}{\partial w} = 0 \rightarrow \quad w = \sum_{t=1}^{N} \alpha_t \varphi(x_t) \tag{10a}$$

$$\frac{\partial \mathcal{L}}{\partial b} = 0 \rightarrow \quad \sum_{t=1}^{N} \alpha_t = 0 \tag{10b}$$

$$\frac{\partial \mathcal{L}}{\partial e_t} = 0 \rightarrow \quad \alpha_t = \gamma e_t, t = 1, \ldots, N \tag{10c}$$

$$\frac{\partial \mathcal{L}}{\partial \alpha_t} = 0 \rightarrow \quad v_t = w^T \varphi(x_t) + b + e_t, t = 1, \ldots, N \tag{10d}$$

Now substitute equations (10a) and (10c) into (10d):

$$\left[\begin{array}{c|c} 0 & 1_N{}^T \\ \hline 1_N & \Omega + \gamma^{-1} I_N \end{array} \right] \left[\begin{array}{c} b \\ \hline \alpha \end{array} \right] = \left[\begin{array}{c} 0 \\ \hline v \end{array} \right] \tag{11}$$

Where

$$v = [v_1, \ldots, v_N]^T \in \mathbb{R}^N$$
$$1_N = [1, \ldots, 1]^T \in \mathbb{R}^N$$
$$\alpha = [\alpha_1, \ldots, \alpha_N]^T \in \mathbb{R}^N$$
$$\Omega_{ij} = k(x_i, x_j) = \varphi(x_i)^T \varphi(x_j) \forall_{i,j} = 1, \ldots, N$$
$$k(x_i, x_j) = e^{-\frac{\|x_i - x_j\|_2}{\sigma}}$$

With k the positive definite kernel function. If the solution for b and α is calculated, the (non)linear function f evaluated on a new point $x^* \in \mathbb{R}^d$ can now be written as a linear combination of Lagrangian multipliers $\alpha_i, i = 1, \ldots, N$ and an offset $b \in \mathbb{R}$:

$$f(x) = \sum_{i=1}^{N} \alpha_t k(x, x_i) + b \tag{12}$$

Note that the length of α, and so the complexity of the model f is dependent on the number of learning points N. The linear growing complexity becomes a big problem if the number of learning points becomes huge. If one takes a closer look to the final model structure (equation 12), it can be seen that the model consist of N neurons, and the centers are represented by the learning points x_i, so if one is refer back to Fig. 4, observation learns that $m = N$ and $c_i = x_i$ for $i = 1, \ldots, N$. Due to the dependancy of the number of learning points N, which is defined as the model complexity, a reduction step is considered in section V, where a model reduction step is introduced which reduces the complexity N of the final model.

B. Tensor Product Construction: MISO Spline interpolating

Tensor Product Constructions operate on a set of univariate functions, where n_i different functions are defined for each input dimension i, where $i = 1, \ldots, d$. The tensor product construction can be rewritten to a neural network structure as in Fig. 4, by defining for each neuron $k = 1, \ldots, m$ the same basis function $\Phi : \mathbb{R}^d \to \mathbb{R}$, such that the d input dimensions

span a Tensor product, where the linear combination of the m spanned Tensor products defines the nonlinear function f (without an offset b). The basis functions are calculated by forming the tensor product of univariate functions ς_{ij}, where each univariate function ς_{ij} for $j = 1, \ldots, n_i$ belongs to input dimension i for $i = 1, \ldots, d$, is multiplied with each other univariate function of the other dimensions p for $p \neq i$. For example, if $\hat{\varsigma}_i = [\varsigma_{i,1}, \varsigma_{i,2}, \ldots, \varsigma_{i,n_i}]$ denotes a set of univariate functions for input dimension i, where $\varsigma_{i,j} : \mathbb{R} \rightarrow \mathbb{R}$. The set of basis functions K^d are constructed by the span of the univariate functions:

$$K^d := \hat{\varsigma}_1 \times \hat{\varsigma}_2 \times \ldots \times \hat{\varsigma}_d$$

where the total number of basis functions, and therefore the total number of neurons m in Fig. 4 becomes

$$m = \prod_{i=1}^{d} n_i$$

The centers c_l (see Fig. 4), $l = 1, \ldots, m$ will be calculated by defining for each dimension i a equidistance grid vector $c_i \in \mathbb{R}^{n_i}, i = 1, \ldots, d$, where $c_i(j+1) - c_i(j) = \bar{c}_i, j = 1, \ldots, (n_i - 1)$ where the grid space C_{grid} is defined by calculating all the combinations of elements of each of the i grid vectors c_i, so $C_{grid} \in \mathbb{R}^{d \times m} = c_1 \otimes \ldots \otimes c_d = [\hat{c}_1 \ldots, \hat{c}_m] \in \mathbb{R}^{d \times m}$, which denotes the tensor product of the d grid vectors c_i. Now each basis function Φ is the product of d univariate function $\Phi(x - \hat{c}_l) = \prod_{i=1}^{d} \varsigma(x(i) - \hat{c}_l(i))$ for $i = 1, \ldots, d$ and $l = 1, \ldots, m$, where $x(i)$ denotes the i^{th} dimension from $x \in \mathbb{R}^d$, and $\hat{c}_l(i)$ the i^{th} element from the l^{th} column vector from the vector space $C_{grid} \in \mathbb{R}^{d \times m}$.

A tensor product example is given in Fig. 6, where

$$x = [x_1 \; x_2]^T \in \mathbb{R}^2 \quad x_i \in [-1 \; 1], i = 1, 2$$
$$\varsigma(\cdot) = e^{-\frac{\|\cdot\|_2^2}{\sigma}}, \sigma = \frac{1}{2}$$

Define

$$c_1 = [-0.5 \; 0.5]^T \quad c_2 = [-0.5 \; 0.5]^T$$
$$C_{grid} = \left[\begin{bmatrix} c_1(1) \\ c_2(1) \end{bmatrix}, \begin{bmatrix} c_1(1) \\ c_2(2) \end{bmatrix}, \begin{bmatrix} c_1(2) \\ c_2(1) \end{bmatrix}, \begin{bmatrix} c_1(2) \\ c_2(2) \end{bmatrix} \right] = [\hat{c}_1 \hat{c}_2 \hat{c}_3 \hat{c}_4]$$

The model in Fig. 4 becomes

$$\Phi(x - \hat{c}_l) = e^{-\frac{\|x(1) - \hat{c}_l(1)\|_2^2}{\sigma}} e^{-\frac{\|x(2) - \hat{c}_l(2)\|_2^2}{\sigma}}, l = 1, 2, 3, 4$$
$$b = 0, f(x) = \sum_{i=1}^{m} \theta_i \Phi(x - \hat{c}_i)$$

So the nonlinear function f is a linear combination of the spanned tensor products of the 4 neurons: $f(x) = \sum_{i=1}^{4} \theta_i \Phi(x - c_i)$. The basis functions Φ are shown in Fig. 6.

A technique which can be applied by a tensor product construction is Spline Interpolation. Splines are piecewise polynomials with pieces that are smoothly connected together. The joining points of the polynomials are called knots, where the set of all knots will be denoted by c_1. For a spline of degree n, each segment is a polynomial of degree n, which would suggest that there are needed $(n+1)$ coefficients to describe each piece. However, there is an additional smoothness constraint that imposes the continuity of the spline and its derivatives up to an order of $(n-1)$ at the knots, so that there is only one degree of freedom per segment. Due to Schoenberg [17], these splines can be uniquely characterized in terms of a B-spline expansion:

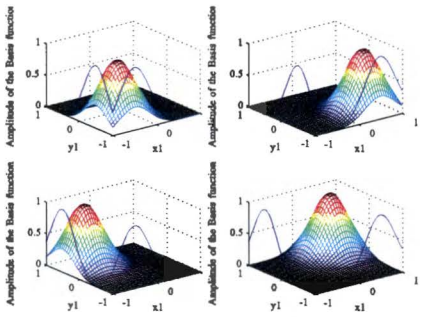


Fig. 6. Example Tensor product

$$f(x) = \sum_{k \in \mathbb{Z}} \theta_k \beta^n (x - k) \tag{13}$$

where $x, \theta \in \mathbb{R}$, $f : \mathbb{R} \rightarrow \mathbb{R}$ and which involves the integer shifts of the central B-spline of degree n denoted by $\beta^n(x)$. The parameters of the model are the B-spline coefficients θ, which will be derived from the sampled learning set S_N, which is constrained that it contains the input/output relations $(k_i, v_i), i = 1, \ldots, N$. The basis function $\Phi = \beta^n(x)$ can be derived from the $(n+1)$-fold convolution of a rectangular pulse β^0.

$$\beta^0(x) = \begin{cases} 1, & -\frac{1}{2} < x < \frac{1}{2} \\ \frac{1}{2}, & |x| = \frac{1}{2} \\ 0, & \text{otherwise} \end{cases}$$

$$\beta^n(x) = \underbrace{\beta^0 * \beta^0 * \ldots * \beta^0(x)}_{(n+1)\text{times}}$$

Now consider the spline interpolation problem, where the coefficients are determined such that the function goes through the data points $(k_i, v_i), i = 1, \ldots, N$ exactly. For splines of degree $n = 0$ (piecewise constant) and $n = 1$ (piecewise linear), this is trivial matter because the B-spline coefficients are identical to the signal samples: $\theta_k = v_k$. For higher-degree splines, however the situation is more complex. To derive this type of signal processing algorithm, introduce the B-spline kernel $b^n_{\Delta_f}$, which is obtained by sampling the B-spline of degree n expanded by a factor of Δ_f.

$$b^n_{\Delta_f}(k) = \beta^n(x/\Delta_f)|_{x=k} \leftrightarrow^z B^n_{\Delta_f}(z) = \sum_{k \in \mathbb{Z}} b^n_{\Delta_f}(k) z^{-k}$$

Now, given signal samples from S_N, determine coefficients θ_k such that there is a perfect fit at the integers:

$$\sum_{k \in \mathbb{Z}} \theta_k \beta^n (x - k) \Big|_{x=k} = v_k = (b^n_1 * \theta)(k)$$

Define the inverse convolution operator

$$(b_1^n)^{-1}(k) \leftrightarrow^z \frac{1}{B_1^n(z)}$$

So the solution can be written as:

$$\theta_k = ((b_1^n)^{-1} * v)(k) \tag{14}$$

This means that the coefficients θ can be found by using a cascade of first order causal and anti causal filters [18].

To bring out the connection between the spline interpolation process and the traditional approach for band-limited functions, it is helpful to introduce the cardinal spline basis function that are the spline analogs of the sinc function. Combining (13) and (14):

$$f(x) = \sum_{k\in\mathbf{Z}} \left((b_1^n)^{-1} * v\right)(k)\beta^n(x-k) \tag{15}$$

$$= \sum_{k\in\mathbf{Z}} v(k) \sum_{l\in\mathbf{Z}} (b_1^n)^{-1}(l)\beta^n(x-l-k) \tag{16}$$

$$= \sum_{k\in\mathbf{Z}} v(k)\varsigma^n(x-k) \tag{17}$$

$$\text{where } \varsigma^n(x) = \sum_{k\in\mathbf{Z}} (b_1^n)^{-1}(k)\beta^n(x-k) \tag{18}$$

Now define a grid space $c_1 \in \mathbb{Z}$, where the output $v = f(x)$ is dependent of the linear combination of shifted basis functions ς^n where the grid space c_1 can be scaled by multiplication of a constant Δ_f. Note that if the grid space is scaled, the basis function ς^n has to be expended by a factor $\frac{1}{\Delta_f} : \varsigma^n(\frac{1}{\Delta_f}x)$. Equation (17) provides a spline interpolation formula that uses the signal values as coefficients. The formula works because $\varsigma^n(x)$ has the same interpolation property as the sinc function: it vanishes for all integers except at the origin, where it takes the value one. The fourier transform of (17) is

$$H^n(w) = \left(\frac{\sin(w/2)}{w/2}\right)^{n+1} \frac{1}{B_1^n(e^{jw})} \tag{19}$$

which will converge to the sinc function if $n \to \infty$. Spline interpolating can be extended to MISO systems in the following way. Assume the mapping $\mathcal{X} \to \mathcal{V}$, where $x = [x_1, \ldots, x_d]^T$. In stead of a one dimensional grid space c_1, define a d-dimensional grid space $C_{grid} = c_1 \otimes c_2 \otimes \ldots \otimes c_d$, where each grid vector c_i defines n_i grid points for each dimension $i = 1, \ldots, d$. The one dimensional output v is the linear combination of the set of m basis functions

$$\Phi(x - \hat{c}_l) = \varsigma^n(x(1) - \hat{c}_l(1)) \ldots \varsigma^n(x(d) - \hat{c}_l(d))$$

$$c_l \in C_{grid} \in \mathbb{R}^{d\times m}$$

so $f(x) = \sum_{i=1}^m \theta_i \Phi(x - \hat{c}_i)$

V. Orthogonal Least squares Learning Algorithm for static modeling

In section III, it was derived that each static function f can be written as a linear combination of nonlinear kernel functions $k(\cdot,\cdot)$. If k is chosen as a Gaussian kernel, that is $k(x_1, x_2) = e^{\frac{-\|x_1-x_2\|_2^2}{\sigma}}$ for $x_1, x_2 \in \mathcal{X}$ and $\sigma \in \mathbb{R}^+$, a Radial Basis Function Neural Network structure will appear, which has been visualized in Fig. IV-A (which can be regarded as a special two-layer neural network which is linear in the parameters). In section III, one can see that the order of the model grows linear with the number of learning samples N. If one takes a closer look at the model structure, N fixed learning points $x_i \in \mathcal{X}, i = 1, \ldots, N$ are stored, which are considered at center points. In this section, a reducing algorithm will be derived in order to reduce the complexity of the resulting nonlinear RBF, which will be measured as the number of neurons m (which is N if no reduction has been applied). Reducing the number of neurons from N to $N_s < N$ here consists of selecting a subset of center points from the learning set $x_j \in S_N, j = 1, \ldots, N_s$ in such a way that $v_i - f(x_i)$ is minimal for $i = 1, \ldots, N$.

A RBF neural network can be seen as a linear regression model

$$\hat{v}_j = f(x_j) = \sum_{i=1}^{N} \theta_i k(x_j, c_i) \tag{20}$$

that predicts $\hat{v}_j, j = 1, \ldots, N$ as the desired j^{th} output sample. Here, θ_i are the linear parameters and $k(x_i, c_j) = e^{-\frac{\|x_i-c_j\|_2^2}{\sigma}}$ is known as the regressor which in this case is chosen as a Gaussian kernel and $e(t)$ is assumed to be uncorrelated with the regressor sample $k(x_j, c_i)$. In this section, the center points c will be chosen as the learning points x, such that the vector of center points $c = [x_1, \ldots, x_N]$.

Equation (20) can be written in matrix notation for $j = 1, \ldots, N : V_{obs} = K\theta + E$, where

$$V_{obs} = [v_1, v_2, \ldots, v_N]^T \tag{21}$$

$$K = [k_1, \ldots, k_N]$$

$$\theta = [\theta_1, \ldots, \theta_N]^T \tag{22}$$

$$k_i = [k(x_1, x_i), k(x_2, x_i) \ldots, k(x_N, x_i)]^T, i = 1, \ldots, N$$

$$E = [e_1, \ldots, e_N]^T\text{: modeling error where } e_j = v_j - \hat{v}_j$$

So the center points are chosen in such a way that they correspond to the N different inputvalues x_i, so that each column k_i of K corresponds to a Gaussian shaped curve centered at x_i, and evaluated to the N different input samples x_j. The regressor columns of K form a set of linear independent basis vectors, and the Least Squares solution is $\hat{\theta} = (K^TK)^{-1}K^TV_{obs}$. Each regressor vector k_i will contribute to the total output energy $V_{obs}^TV_{obs}$, but due to the fact that there is a certain correlation between the N different regression vectors, it is not clear how an individual regressor vector k_i contributes to the total amount of energy. This can be solved by decompose the matrix $K = WA$, where W is a $N \times N$ matrix with orthogonal column's w_i, such that $W^TW = H$, where H is diagonal with elements

$h_i = w_i^T w_i, i = 1, \ldots, N$, and A has the following structure:

$$A = \begin{bmatrix} 1 & a_{12} & a_{13} & \ldots & a_{1N} \\ 0 & 1 & a_{23} & \ldots & a_{2N} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \ldots & \ldots & \ldots & 1 & a_{(N-1)N} \\ 0 & \ldots & \ldots & 0 & 1 \end{bmatrix} \quad (23)$$

The matrix A is chosen in such a way that it applies a transformation to the orthogonal set of basis functions in W in order to recover the original regression matrix K. The space spanned by the N vectors w_i is now the same as the space spanned by the N different Gaussian centered around x_i which are evaluated on the learning input samples, and thus equation (21) can be written as $V_{obs} = Wg + E$ where the Least squares solution will be given by $\hat{g} = (W^T W)^{-1} W^T V_{obs}$ or $\hat{g}_i = (w_i^T w_i)^{-1} w_i^T V_{obs}, i = 1, \ldots, N$.

Once the matrix K has been made orthogonal, the reduction step now involves selecting a subset from the N orthogonal regressors, which are the columns of the regression matrix W. So one is looking for $N_s < N$ significant centers (regressors) out of a set of N different regression vectors which

This can be done by the so-called forward selection. First, each single regressor k_i out of all N possible ones is selected and the performance with each of these regressors k_i is evaluated by optimizing the associated parameters $\hat{g}_i$. Then, the regressor that approximated the output V_{obs} best, i.e. the most significant one, is selected. This regressor and its associated parameter (center point x_i) will be denoted as $\bar{k}_1$ and $\bar{g}_1$. Second, the part of the output V_{obs} not explained by $\bar{k}_1$ can be calculated as $\bar{V}_1 = V_{obs} - \bar{k}_1 \bar{g}_1$. Next, each of the remaining $N-1$ regressors is evaluated for explaining $\bar{V}_1$, again by optimizing the associated parameters. This procedure can be repeated until N_s regressors have been selected.

The first significant regressor can be selected due to the fact that w_i and w_j are orthogonal for $i \neq j$, and the total energy of the output V can be written as

$$V_{obs}^T V_{obs} = \sum_{i=1}^{N} g_i^2 w_i^T w_i + E^T E \quad (24)$$

Now one is searching for the direction (regressor) that adds the most amount of energy to the output. This can be done by searching for the largest value of the error reduction ratio, which is defined as: $[err]_i = \frac{g_i^2 w_i^T w_i}{V_{obs}^T V_{obs}}$ for $i = 1, \ldots, N$.

The regression selection procedure is summarized as follows:

1) At the first step, for $1 \leq i \leq N$, compute

$$w_1^{(i)} = k_i \quad (25)$$

$$g_1^{(i)} = \left(\left(w_1^{(i)} \right)^T w_1^{(i)} \right)^{-1} \left(w_1^{(i)} \right)^T V_{obs} \quad (26)$$

$$[err]_1^{(i)} = \left(g_1^{(i)} \right)^2 \left(w_1^{(i)} \right)^T w_1^{(i)} (V_{obs}^T V_{obs})^{-1} \quad (27)$$

Find $[err]_1^{(i_1)} = \max\{[err]_1^{(i)}, 1 \leq i \leq N\}$
i_1 now correspond to the index number of the N different center points, that adds the most amount of energy to the output V for one of the N directions. Select the first regressor $w_1 = w_1^{i_1} = k_{i_1}$

2) At the kth step, where $k \geq 2$, for $1 \leq i \leq N, i \neq i_1, \ldots, i \neq i_{k-1}$, compute

$$\alpha_{jk}^{(i)} = \left(w_j^T w_j \right)^{-1} w_j^T k_i, 1 \leq i \leq k$$

$$w_k^{(i)} = k_i - \sum_{j=1}^{k-1} \alpha_{jk}^{(i)} w_j$$

$$g_k^{(i)} = \left(\left(w_k^{(i)} \right)^T w_k^{(i)} \right)^{-1} \left(w_k^{(i)} \right) V_{obs}$$

$$[err]_k^{(i)} = \left(g_k^{(i)} \right)^2 \left(w_k^{(i)} \right)^T w_k^{(i)} \left(V_{obs}^T V_{obs} \right)^{-1}$$

Find

$$[err]_k^{(i_k)} = \max\{[err]_k^{(i)}, 1 \leq i \leq N, i \neq i_1, \ldots, i \neq i_{k-1}\}$$

$$\text{select } w_k^{(i_k)} = k_{i_k} - \sum_{j=1}^{k-1} \alpha_{jk} w_j, \text{where } \alpha_{jk}^{i_k}, 1 \leq j \leq k$$

3) The procedure is terminated at the N_sth step, when the total amount of energy crosses a certain energy level ρ:

$$1 - \sum_{j=1}^{N_s} [err]_j \leq \rho, \quad 0 \leq \rho \leq 1 \quad (28)$$

which can be considered as searching for a number N_s regressors, such that the total energy that is not added to the system reach a certain energy level ρ

VI. Hammerstein system Identification

In this section, a nonlinear Hammerstein identification method will be derived by parameterizations of two nonlinear neural network structures, which will represent the nonlinear function f, as shown in Fig. 1. The dynamics will consist of a linear ARX model, which can be described by

$A(q)y(t) = B(q)v(t) + e(t)$ where $y(t), e(t) \in \mathbb{R}^{ry}, v(t) \in \mathbb{R}^d$
$q^i y(t) = y(t-i)$ denotes the shift operator and

$$A(q) = I + \sum_{i=1}^{na} A_i q^{-i}, B(q) = \sum_{i=1}^{nb} B_i q^{-i}$$

$$\theta_{arx} = [A_1 \ \ldots \ A_{na} \ B_1 \ \ldots \ B_{nb}]^T$$

$$\phi_{arx} = [-y(t-1) \ \ldots \ y(t-na) \ v(t-1) \ \ldots \ v(t-nb)]^T$$

$$\hat{y}(t|\theta_{arx}) = \phi_{arx}^T \theta_{arx}$$

The two different neural networks that will be superposed into the nonlinear mapping f will be:

1) Method of splines: which was described in section IV-B
2) RBF: which has been described in section IV-A

A. Identification with method of splines

According to section IV-B, the mapping $\mathcal{X} \rightarrow \mathcal{V}$ can be evaluated by a linear combination of nonlinear basis

functions:

$$f(x) = \sum_{i=1}^{m} \theta_i \Phi(x - \hat{c}_i) = \Theta_S \Phi_S(x) \tag{29}$$

where $\Phi_S(x) = [\Phi_1(x - \hat{c}_1) \ldots, \Phi_m(x - \hat{c}_m)]^T$ and $\Theta_S = [\theta_1, \ldots, \theta_m]$. If one wants to model a mapping $\mathcal{X} \rightarrow \mathcal{V}$, where $\mathcal{V} = \mathbb{R}^d$ (so it has the same dimension as the input $\mathcal{X}$), the model structure (29) must be placed d times in parallel, that $\Phi_S(x) \in \mathbb{R}^m$ and $\Theta_S \in \mathbb{R}^{d \times m}$.

For the linear model, assume an ARX MIMO model: $\hat{y}(t|\theta_{arx}) = \phi_{arx}^T(t)\theta_{arx}$ and substitute with (29):

$$\begin{aligned} A(q)y(t) &= B(q)v(t) + e(t) = \\ A(q)y(t) &+ B(q)\Theta_S\Phi_S(t) + e(t) = \\ A(q)y(t) &+ \bar{B}(q)\Phi_S(t) + e(t) \end{aligned} \tag{30}$$

The main emphasis here is that different nonlinear systems can now be represented using a set of coefficient matrices $\bar{B} = B(q)\Theta_S$ and knot vectors Φ_S which has a very attractive appeal because all the nonlinearity is embedded in the transformation from the actual inputs $x(t)$ into the virtual inputs $v(t)$. This parameterized structure is very beneficial to both numerical simulations and control design. With the transformation $v = \Theta_S\Phi_S$, the identification problem using the Hammerstein model can be formulated using the output y and transformed input Φ_S: $A(q)y(t) = \bar{B}(q)\Phi_S(t) + e(t)$, So the nonlinearity, which is hidden in $\bar{B} = [\bar{B}_1, \ldots, \bar{B}_{nb}]^T$, can be found by linear optimization, where the unique solution $\hat{\theta}_{arx} = [\hat{A}_1, \ldots, \hat{A}_{na}, \hat{\bar{B}}_1, \ldots, \hat{\bar{B}}_{nb}]$ can be found by the least square optimization. Once $\hat{\bar{B}}$ is obtained, $\hat{B}$ can be calculated by an SVD.

$$\begin{aligned} \hat{\bar{B}} &= USV^T = [U_1 U_2] \begin{bmatrix} S_d & 0 \\ 0 & S_{\bar{d}} \end{bmatrix} \begin{bmatrix} V_1^T \\ V_2^T \end{bmatrix} \\ &= U_1 S_d V_1^T + U_2 S_{\bar{d}} V_2^T \\ &= \underbrace{[\hat{B}_1\ \hat{B}_2\ \ldots\ \hat{B}_{nb}]^T}_{\hat{B}} \underbrace{S_d V_1^T}_{\hat{\theta}_S} + U_2 S_{\bar{d}} V_2^T \approx \hat{B}\hat{\theta}_S \end{aligned}$$

So the nb matrices $\hat{B} = [\hat{B}_1, \ldots, \hat{B}_{nb}]$ and the approximated coefficients matrix $\hat{\theta}_S$, which contains the nonlinear information, can be approximated by applying a truncation to the SVD of the identified $\hat{\bar{B}}_i, i = 1, \ldots, n_b$.

The final model now becomes

$$\hat{A}(q)y(t) = \hat{B}(q)v(t) \tag{31}$$

$$\text{where } v(t) = \hat{\theta}_S \Phi_s(x(t)) \tag{32}$$

B. Identification with SVMs

In section IV-A, a technique has been described how to identify a static nonlinear function f, by applying SVM techniques which function f can be described by a linear combination of N kernel functions k, centered around the input learning points x_i. In this section, a technique [5] will be described how to transform this static mapping to a dynamic mapping, by apply superposition of SVM into a linear ARX model. This can be done by define a dynamic cost function, which involves an ARX model.

Given an ARX model

$$y_t = \sum_{i=1}^{n_a} a_i y_{t-i} + \sum_{j-0}^{n_b} b_j v_{t-j} + e_t$$

where e_t is assumed to be a white noise sequence with $E[e_t] = 0, E[e_t^2] = \sigma^2 < \infty$. Assume a nonlinear input mapping $f : \mathbb{R} \rightarrow \mathbb{R} : x \rightarrow v = f(x)$, write (VI-B) as

$$y_t = \sum_{i=1}^{n_a} a_i y_{t-i} + \sum_{j-0}^{n_b} b_j f(x_{t-j}) + e_t \tag{33}$$

with $f(\cdot)$ the nonlinear mapping, which will be assumed that it has the following structure:

$$f(x) = w^T \varphi(x) + d_0 \tag{34}$$

Substitute (34) in (33):

$$y_t = \sum_{i=1}^{n_a} a_i y_{t-i} + \sum_{j-0}^{n_b} b_j [w^T \phi(u_{t-j}) + d_0] + e_t \tag{35}$$

To solve the parameters a_i, b_i, w and d_0, use the method described in paragraph IV-A. So try to minimize

$$\min_{w,a,b,d_0,e} = \frac{1}{2} w^T w + \frac{\gamma}{2} \sum_{t=r}^{N} e_t^2 \tag{36a}$$

$$\text{s.t.} \quad y_t = \sum_{i=1}^{n_a} a_i y_{t-i} + \sum_{j=0}^{n_b} b_j [w^T \varphi(x_{t-j}) + d_0] + e_t \tag{36b}$$

with $r = \max(n_a, n_b) + 1$. The Lagrangian of problem 36 is given as:

$$\mathcal{L}(w, d_0, b, e, a; \alpha) = \frac{1}{2} w^T w + \frac{\gamma}{2} \sum_{t=r}^{N} e_t^2 \tag{37}$$

$$- \sum_{t=r}^{N} \alpha_t \left(\sum_{i=1}^{n_a} a_i y_{t-i} + \sum_{j-0}^{n_b} b_j [w^T \phi(u_{t-j}) + d_0] + e_t - y_t \right) \tag{38}$$

The conditions for optimality can be calculated by setting the derivatives $\frac{\partial \mathcal{L}}{\partial w}, \frac{\partial \mathcal{L}}{\partial a_i}, \frac{\partial \mathcal{L}}{\partial d_0}, \frac{\partial \mathcal{L}}{\partial e_t}, \frac{\partial \mathcal{L}}{\partial \alpha_t}, \frac{\partial \mathcal{L}}{\partial b_j}$ to zero.

This problem can be solved by the following set of equations:

$$\left[\begin{array}{c|c|c} 0 & 0 & \vec{b} \cdot 1_{N-r+1}^T \\ \hline 0 & 0 & \mathcal{Y} \\ \hline \vec{b} \cdot 1_{N-r+1} & \mathcal{Y}^T & L^0 + \frac{1}{\gamma} I_{N-r+1} \end{array}\right] \left[\begin{array}{c} d_0 \\ \hline a \\ \hline \alpha \end{array}\right] = \left[\begin{array}{c} 0 \\ \hline 0 \\ \hline Y \end{array}\right] \tag{39}$$

with

$$\mathcal{Y} = \begin{bmatrix} y_{r-1} & y_r & \cdots & y_{N-1} \\ y_{r-2} & y_{r-1} & \cdots & y_{N-2} \\ \vdots & \vdots & & \vdots \\ y_{r-n} & y_{r+1-n} & \cdots & y_{N-n} \end{bmatrix} \quad (40a)$$

$$Y = [y_r \; y_{r+1} \; \cdots \; y_N]^T \in \mathbb{R}^{N-r+1} \quad (40b)$$

$$\alpha = [\alpha_r, \ldots, \alpha_N]^T \in \mathbb{R}^{N-r+1} \quad (40c)$$

$$\hat{b} = \sum_{j=1}^{n_b} b_j \quad (40d)$$

Because the b_j are not known, it is impossible to solve this set of equations. A solution to this problem would be to write

$$y_t = \sum_{i=1}^{n_a} a_i y_{t-i} + \sum_{j=0}^{n_b} b_j [w^T \varphi(x_{t-j}) + d_0] + e_t \text{ as}$$

$$y_t = \sum_{i=1}^{n_a} a_i y_{t-i} + \sum_{j=0}^{n_b} w_j^T \varphi(x_{t-j}) + d + e_t \quad (41)$$

A result of this is that this model belongs no longer to a Hammerstein structure due to the replacements of one single w by several w_j, $j = 1, \ldots, n_b$. To avoid this problem, extra constraints can be added to ensure that the several w_j's are collinear with each other, which means that the direction of all w_j's are the same, but only the length is different. This is the same by applying the demand that the matrix$[w_1|\ldots|w_{nb}]$ has a rank 1. Because one apply a mapping from the input domain $\mathbb{R}$ to a higher dimension by $\varphi(\cdot)$, which is in general unknown, adding such constraints will make the optimization problem non-convex. Therefore, remove the extra constraint and project the obtained model onto the Hammerstein model by a first rank SVD approximation.

If the collinearity constraint is neglected, the optimization problem that is ultimately solved is the following:

$$\min_{w_j, a, d, e} = \frac{1}{2} \sum_{j=0}^{n_b} w_j^T w_j + \gamma \frac{1}{2} \sum_{t=r}^{N} e_t^2 \quad (42a)$$

$$\text{s.t.} \quad y_t = \sum_{i=1}^{n_a} a_i y_{t-i} + \sum_{j=0}^{n_b} w_j^T \varphi(u_{t-j}) + d + e_t \quad (42b)$$

$$\sum_{t=1}^{N} w_j^T \varphi(x_t) = 0 \quad (42c)$$

$$\text{with } t = r, \ldots, N \text{ and } j = 0, \ldots, n_b \quad (42d)$$

The extra constraint $\sum_{t=1}^{N} w_j^T \varphi(u_t) = 0$ centers the nonlinear functions $w_j^T \varphi(\cdot), j = 0, \ldots, n_b$ around their average over the training set. This removes the uncertainty resulting from the fact that that any set of constraints can be added to the terms of the additive nonlinear function, as long as the sum of the constants is zero. This constraint gives a clear meaning of the bias parameter d:

$$d = \sum_{j=0}^{n_b} b_j \left((1/N) \sum_{k=1}^{N} f(u_k) \right) \quad (43)$$

The solution to the system is:

$$\left[\begin{array}{c|c|c|c} 0 & 0 & \mathbf{1}_{N-r+1}^T & 0 \\ \hline 0 & 0 & \mathcal{Y} & 0 \\ \hline \mathbf{1}_{L-r+1} & \mathcal{Y}^T & L^0 + \frac{1}{\gamma} I_{N-r+1} & K^{0T} \\ \hline 0 & 0 & K^0 & \left(\sum_{t=1}^{N} \sum_{l=1}^{N} K(u_l, u_t)\right) I_{m+1} \end{array}\right] \begin{bmatrix} d \\ a \\ \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ Y \\ 0 \end{bmatrix}$$

Now the identified system can be written as:

$$\hat{y}_t = \sum_{i=1}^{n_a} a_i \hat{y}_{t-i} + d + \sum_{j=0}^{n_b} \left[\sum_{T=r}^{N} \alpha_T K(u_T, u^*(t-j)) + \beta_j \sum_{T=1}^{N} K(u_T, u_{t-j}^*) \right]$$

here u^* is a new input which not belongs to the training set S.

VII. Results: Real World Problem on Identification of a Glass Furnace

This section deals with the identification of a real world process, which aims to find a relation (identification) between an input $x(t)$ (amount of new glass entering the furnace, measured in [Tons/Day]) and the temperature $T(\xi, t)$ as function of a spatial coordinate ξ in the furnace geometry $\mathbb{X}$ and time t. The physical phenomena in a glass furnace [4],[16], are is visualized in Fig. 7. The dynamics, like Temperature, Pressure and Velocity can be described by means of a Distributed Parameter System (DPS) which is a system whose states space is infinite-dimensional, and is usually described by partial differential equations. One way to model the time-dependant transport phenomena in fluids is to use Computational Fluid Dynamics (CFD) software, where the CFD simulates at each time step $t \in \mathbb{T}$ the physical quantities at a discrete point in the spatial domain $\mathbb{X}$. To simulate the fluid dynamics, the partial differential equations (Navier-Stokes equations) have to be approximated through discretisation of the spatial domain $\mathbb{X}$ and then solved numerically at each and every time step. This simulation cost a lot of computation power, whereas identification of the physical time dependant data is difficult due to the dependance of place. In order to avoid this problem, the spatial and temporal domains can be divided by means of a Singular Value Decomposition.

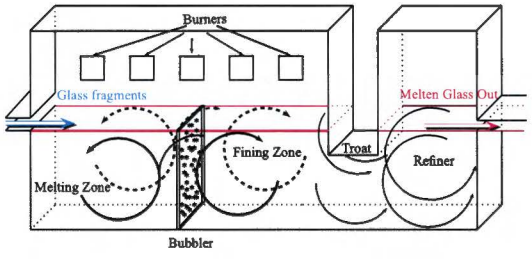


Fig. 7. GlassFurnace

Suppose that a numerical simulation model is available, which calculates the temperature of the glass at discrete time instance t_k, where $t_k = k\triangle_t, k \in \mathbb{N}$ and $\triangle_t$ the sampling interval, as function of L place dependent coordinates $\xi_l \in \mathbb{Z} \subset \mathbb{R}^2$ which denotes a set of grid positions. Now define the temperature vector $T(t_k) = [T(\xi_1, t_k), \ldots, T(\xi_L, t_k)]^T$, which stores the temperature at each grid position $\xi_l \in \mathbb{X}$ and store a set of these temperature vectors in a snapshot matrix T_{snap}, which is a matrix of temperatures stored at time instances t_k, for $k = 1, \ldots, N$

$$T_{\text{snap}} = [T(t_1) \ldots\ T(t_N)] = \begin{pmatrix} T(\xi_1,t_1) & \ldots & T(\xi_1,t_N) \\ \vdots & \ddots & \vdots \\ T(\xi_L,t_1) & \ldots & T(\xi_L,t_N) \end{pmatrix} \quad (44)$$

If T_{snap} is decomposed by a Singular Value Decomposition (SVD), that is $T_{\text{snap}} = \Phi\Sigma\Psi$, the temperature (as in [4]) can be written as a linear combination of time dependent Fourier coefficients $a(t)$ and place dependent basis function $\Phi = [\phi_1 \ldots, \phi_L]$ where ϕ_i is the i^{th} left singular vector of T_{snap}:

$$T(t) = \sum_{i=1}^{n} a_i(t)\phi_i + \epsilon_T(t) = \Phi_n a^n(t) + \Phi_\epsilon a^\epsilon(t),$$

where $a^n(t) = [a_1(t), \ldots, a_n(t)]^T$

$a^\epsilon(t) = [a_{n+1}(t), \ldots, a_L]^T$

$\Phi_n = [\phi_1, \ldots, \phi_n]$

$\Phi_\epsilon = [\phi_{n+1}, \ldots, \phi_L]$

where $\epsilon_T = \Phi_\epsilon a^\epsilon(t)$ denotes the model reduction error obtained by truncation of the spectral decomposition, and the temperature is approximated by $T^n(t) = \Phi_n a^n(t)$ with truncation error $\epsilon_T(t) = T(t) - T^n(t)$. The time dependent parameter $a_i(t)$ can be calculated by projecting the temperature $T(t)$ to the i^{th} orthonormal spatial basis functions ϕ_i, which can be archived by calculating the inner product of the snapshot matrix and the basis vector $a_j(t) = \langle \phi_j, T(t) \rangle, j = 1, \ldots, n$. $a^n = [a_1, \ldots, a_n]^T$ is the vector of (time dependent) Fourier coefficients.

The identification of the temperature profile in the spatial domain can now be obtained by learning the behavior of the time evaluation of the Fourier coefficients $a_n(t)$ of the truncated model. In this paper, an identification model has been derived between the pullrate x [tons/day] (which determines the input of new fragments of broken glass measured in [tons/day]) and Fourier coefficients $a^n(t)$, so one wants to derive the behavior $x \rightarrow a^n$ for $n = 3$ and the learningset $S_N = \{(x_t, a_t^n)\}_{t=1}^N$ is filled with $N = 1000$ time samples. Due to MIMO identification, the n Fourier coefficients $a^n(t)$ will be scaled by the first n singular values σ_{sv}^n, where

$$\Sigma = \begin{bmatrix} \sigma_{sv}^n & 0 \\ 0 & \sigma_{sv}^\epsilon \end{bmatrix} \quad (45)$$

which result in the scaled fourier coefficients $a_S^n = a^n (\sigma_{sv}^n)^{-1}$. The simulation process is depicted in Fig. 8. So the model identification strategy will be:

1) Find a low order basis Φ_n

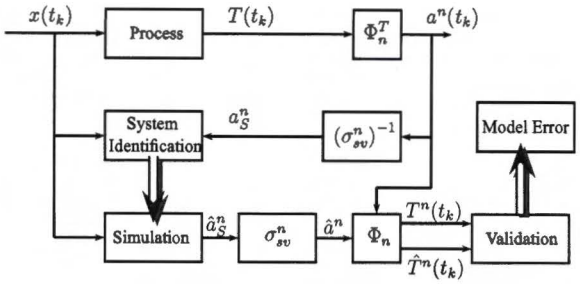


Fig. 8. Identification Process

2) Find a low order model which describes the relation between pullrate x and the time dependant scaled fourier coefficients $a_S^n(t)$
3) Validate the model by projecting the n different basis $\phi_i, i = 1 \ldots, n$ to the temperature $\hat{T}^n(t)$ which is a linear combination of the identified fourier coefficients $\hat{a}_i^n(t), i = 1, \ldots, n$.

Because the identified temperature $\hat{T}^n$ is dependant of the n fourier coefficients $\hat{a}^n$, the first three measured fourier coefficients $a^n, n = 1, 2, 3$ will be compared with the identified fourier coefficients $\hat{a}^n$, derived with the following identification techniques

- Identification with Method of Splines
- Identification with SVMs

The used input is defined as a Pseudo Random Multi-Valued Sequence $x = [2.6\ 3.4]$ with normalized bandwidth $w = [0\ 0.4\pi]$ [rad/sec].

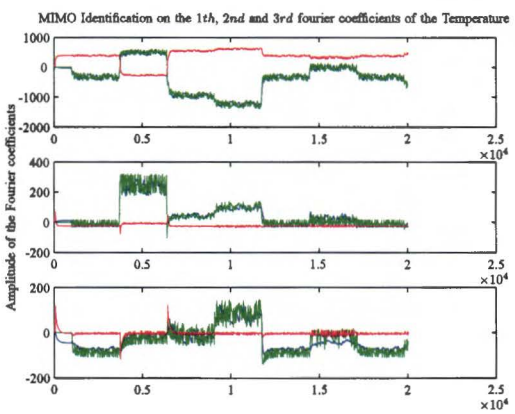


Fig. 9. Fit on Identification Data

Two different models have been derived by MIMO Identification, that is M_{SVM}, which is a Hammerstein model derived with SVMs (see section VI-B), and M_S, which is a Hammerstein model derived with the method of splines (see section VI-A). Using M_{SVM} and M_S, two outputs $x \rightarrow y_{SVM}$ and $x \rightarrow y_S$ have been simulated for two different input signals x. In Fig. 9, the results are plotted where x is the same input signal used for identification. The red line denotes y_S and the blue line denotes y_{SVM}. These two graphs are compared with the green line, which is the

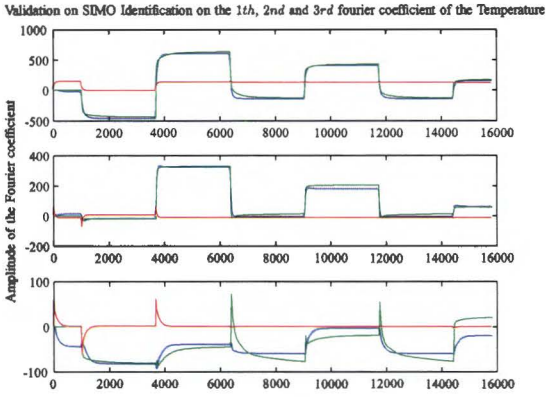


Fig. 10. Fit on Identification Data

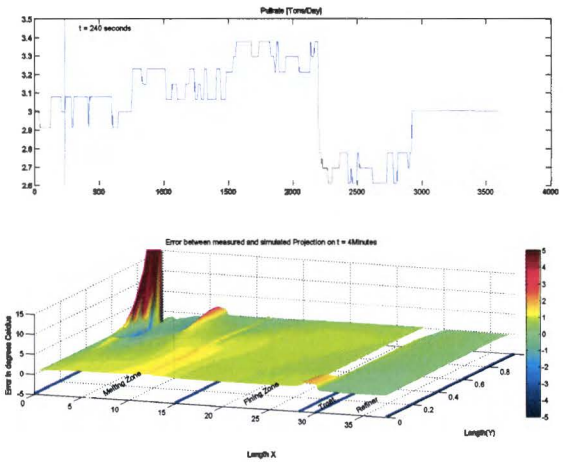


Fig. 11. Projection to Temperature of the error

measured signal $y = a^n$. The same holds for Fig. 10, where x is a validation input signal.

For both graphs (Fig. 9 and Fig. 10), it can be seen that the SVM model fits the behavior of the glass furnace best. In fig. 11, the identification error $e^n(t) = a^n(t) - \hat{a}^n(t)$ is projected back to the spatial domain for one time index at $t = 240$ seconds, where observation learns that the critical errors lie in the melting and the throat zones. This problem may be improved by choosing an other spatial basis function $\bar{\Phi}$.

VIII. Conclusions

In this paper, two different Hammerstein identification methods have been derived, where the nonlinear mapping f has been replaced by a neural network with two different structures, which are the radial basis function construction and tensor product construction. The number and values of the parameters and exact size of the nonlinear model have been derived by SVMs and Method of Splines. For the static and dynamic SVM model, the complexity grows linear with the number of learning points N, which can be reduced using Orthogonal Least Squares for the static model. The Method of splines is based on a linear combination of nonlinear basis functions. It can be seen that for the identification of a glass furnace, the SVM model fits the data best.

IX. Recommendations

The Identification techniques described in section VI-B and VI-A are based on linear ARX models. To improve the results in section VII, where a Glass furnace has been modeled, one can replace the ARX model by a linear state space model [7], which is a wider model class than ARX.

To deal with the complexity problem of section VI-B, one has to solve the complexity problem, which states that the order of the final SVM model grows linear with the size of the learning data N. It has already been showed that it is possible to apply a reduction step for a static SVM model in section V.

References

[1] J.G.F. Francis, The QR Transformation I, *Comput. J.*, vol. 4, 1961, pp 265-271.

[2] H. Kwakernaak and R. Sivan, *Modern Signals and Systems*, Prentice Hall, Englewood Cliffs, NJ; 1991.

[3] David H. Gay and W. Harmon Ray, "Identification and Control of distributed parameter systems by means of the singular value decomposition,*Chemical Engineering Science*

[4] L. Huisman and S. Weiland, "Identification and Model Predictive Control of an Industrial Glass-Feeder",*13th IFAC Symposium on System Identification*

[5] Ivan Goethals, Kristiaan Pelckmans, Johan A.K. Suykens, Bart De Moor, "Identification of MIMO Hammerstein models using least squares support vector machines",*Automatica*, vol.41, 2005, pp1263-1272

[6] Kwong Ho Chan, Jie Bao, William J. Whiten, "Identification of MIMO Hammerstein systems using cardinal spline functions", *Joernal of Process Control*, vol.16, 2006, pp.659-670

[7] Ivan Goethals, Kristiaan Pelckmans, Johan A.K. Suykens, Bart De Moor, "Subspace Identification of Hammerstein Systems Using Least Squares Support Vector Machines", *IEEE Transactions on Automatic Control*, vol.50, 2005, No.10, pp.1509-1519

[8] I. Goethals, K. Pelckmans, J.A.K. Suykens, B. De Moor, "On Model Complexity Control in Identification of Hammerstein Systems", *Proceedings of the 44th Conference on Decision and Control, and the European Control Conference 2005*, 2005, pp.1203-1208

[9] Vinay Prasas, B. Wayne Bequette, "Nonlienar System Identification and model reduction using artificial neural networks", *Computers and Chemical Engineering*, vol.27, 2003, pp.1741-1754

[10] Yuh-Jye Lee, Wen-Feng Hsieh, Chien-Ming Huang, "ϵ-SSVR: A Smooth Support Vector Machine for ϵ-Intensive Regression", *IEEE Transactions on Knowledge and Data Enginereing*, Vol.17, No.5, 2005, pp.678-685

[11] Theodoros Evgeniou, Massimiliano Pontil, Tomaso Poggiom, "Regularization Networks and Support Vector Machines", *Advances in Computational Mathematics*, vol.13, 2000, pp.1-50

[12] S. Chen, S.A. Billings, W. Luo, "Orthogonal least squares methods and their application to nonlinear system identification", *Int. J. Control*, vol.50, 1989, No. 5, pp.1873-1896

[13] Jos De Brabanter, "LS-SVM Regression Modeling and its Applications"

[14] W.J. Whitten, "The use of multi-Dimensioal Cubic Spline Functions for Regression and Smoothing" *The Australian Computer Journal*, Vol. 3 No. 2, 1971, pp.81-88

[15] Haitao Zhang, Yongji Wang, Wei Xiang, Ming Li, "SVD-based Identification Algorithmn for Hammerstein-typed Nonlinear Systems"

[16] Patricia Astrid, "Reduction of Process Suimulation Models"

[17] I.J. Schoenberg, "Contribution to the problem of approximation of equidistant data by analitic functions", *Quart. Appl. Math.*, Vol. 4, pp-45-99, 112-141, 1946

[18] M. Unser, A. Aldroubi. and M. Eden, "Fast B-Spline transforms for continuous image representation and interpolation", *IEEE Trans. Pattern Anal. Machine Intell.*, Vol. 13, No. 3, pp. 277-285, 1991