Local Model Network Identification With Gaussian Processes

Gregor Gregorčič and Gordon Lightbody

***Abstract*—A Bayesian Gaussian process (GP) modeling approach has recently been introduced to model-based control strategies. The estimate of the variance of the predicted output is the most useful advantage of GPs in comparison to neural networks (NNs) and fuzzy models. However, the GP model is computationally demanding and nontransparent. To reduce the computation load and increase transparency, a local linear GP model network is proposed in this paper. The proposed methodology combines the local model network principle with the GP prior approach. A novel algorithm for structure determination and optimization is introduced, which is widely applicable to the training of local model networks. The modeling procedure of the local linear GP (LGP) model network is demonstrated on an example of a nonlinear laboratory scale process rig.**

***Index Terms*—Gaussian processes (GPs), local linear Gaussian process model network, nonlinear system identification, prediction variance, structure optimization.**

I. Introduction

NEURAL networks (NNs) and fuzzy-logic-based models are widely used for the modeling and control of nonlinear systems [1], [2]. The curse of dimensionality and a lack of transparency of the resulting model have shifted research interest towards a local modeling approach [3]–[6], in which each local model is a valid representation of the modeling function on a limited subspace of the operating space. Using validity functions, the local models are then combined, or blended, to form a local model network.

Difficulties related to partitioning of the operating space, structure determination, local model identification, and off-equilibrium dynamics are the main drawbacks of local modeling approaches. Despite such limitations, many neural architectures which embody the local modeling philosophy, have successfully been applied not only to nonlinear modeling and control problems, but also in other engineering applications. The local modeling principle was utilized in signal processing, for adaptive filtering in [7]. In [8], self-organizing maps were used for a partitioning of the operating space and structure determination. The resultant local linear mapping, an extension of self-organizing maps, was then utilized for time-series prediction. Self-organizing maps have been widely applied in robotic systems [9]. The application of local dynamic modeling to nonlinear system identification and control is demonstrated in [10], where a different approach for local model identification was taken. Instead of identification from the input/output samples, the parameters of local models were estimated from the weights of a global self-organizing map. The local models were then used for nonlinear control in a similar manner to these reported in [11] and [12]. Further applications of local model networks for nonlinear modeling and control can be found in [13] and [14] and some recent utilization of the local modeling principle in NN applications were published in [15]–[17].

It is obvious that a local model paradigm will only be able to accurately model the system close to the points where the local models have been identified. Away from these points, the accuracy of the network decreases. This is known as a problem of off-equilibrium dynamics [6], [18]. In an attempt to improve modeling performance between operating points, the use of nonparametrical probabilistic models, such as Gaussian process (GP) priors, was proposed [19]. GP models provide a prediction as well as an estimate of the variance of the predicted output. This variance can be interpreted as a level of confidence in the prediction. This is a main advantage of this type of model in comparison to NNs or fuzzy models.

From an NN point of view, a GP model can be characterized as a form of radial basis function network (RBFN). In this framework, the distribution of the parameters defines the network performance, rather than the parameters themselves. Assuming that the weights of an RBFN are Gaussian distributed, then for any given finite set of inputs, the network outputs will also be Gaussian distributed. This is a defining property of GPs. A specific choice of structure for the RBFN simplifies the model even more. By placing the centers of basis functions on each input point of the input space and assuming that the number of data points and hence number of basis functions approach infinity, the commonly used nonlinear covariance function has been derived [20], [21]. From the machine learning perspective, the covariance function can be understood as a kernel function, which to some extent, links GPs to kernel methods, such as the support vector machine (SVM) [22], [23]. In comparison with GPs, the SVM makes a point prediction, with the GP model generating a predictive distribution.

The GPs prior approach for curve fitting was first introduced in [24]. This work was left unnoticed until a comparison of GPs with widely used models were carried out in [25] and research into GPs expanded rapidly since then. Many learning algorithms have been proposed [26], [27] and GPs have been utilized in a number of applications. In the statistic and machine learning

Manuscript received January 19, 2006; revised August 23, 2006 and November 23, 2006; accepted December 24, 2006. This work was supported by the Ad futura, Science and Education Foundation of the Republic of Slovenia under Grant 521-15/2002.

G. Gregorčič is with the Anstalt für Verbrennungskraftmaschinen List (AVL List GMBH), A-8020 Graz, Austria (e-mail: gregor.gregorcic@avl.com).

G. Lightbody is with the Department of Electrical Engineering, University College Cork, Cork, Ireland (e-mail: g.lightbody@ucc.ie).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNN.2007.895825

community, GPs are widely implemented for attacking classification [20], [28], prediction [29], and regression [30] problems. An extended overview of GP approaches can be found in [20], [29], and [31] and their relation to other Bayesian kernel models and methods is given in [21] and [32].

The GP model has been applied for nonlinear system identification in [33]. In [34], the standard covariance function was extended to include a correlated noise model. In [35], GPs were used for validation of linear and nonlinear models. GPs have been introduced to control applications in [36], where minimum variance control using the GP model has been proposed. The GP model was applied for model-predictive control in [37] and introduced for internal model control in [38].

In common with NNs, the GP model is a black-box technique and does not provide any physical knowledge about the modeled system. Problems of the exact invertibility of the GP model have been pointed out in [38]. Since the GP model is not analytically invertible, numerical approaches have to be utilized to find the inverse, which increases the already large computation load. A numerical inversion can be avoided by the use of a linear covariance function [39]. The use of a linear covariance function also allows the order of the model to be lower than that required for a nonlinear covariance function. Importantly, a decrease in the model order will significantly reduce the computation load. It should also be noted that the procedure of model order selection is much easier when the linear covariance function is used [40].

When a linear covariance function is utilized, the GP model will lose its ability to model rapidly changing modeling surfaces. However, a linear GP (LGP) model using a linear covariance function will provide a linear approximation of the modeling surface. If the operating space is now partitioned into operating regimes and a local LGP model is found for each operating regime, then these local models can be formed into a local LGP model network. The cluster of data associated with a particular local model then consists of much less data points than the full training set. The computational load for each local GP model will, therefore, be decreased, and the overall computational cost of identifying the global model will be reduced.

There is a number of advantages in using local LGP models instead of conventional linear regression models for the local model network. LGP models are more robust to ill conditioning, less prone to bias, and they provide a measure of uncertainty in the prediction. This uncertainty can be then used for structure optimization of the local model network, constructed from local LGP models. This could offer optimum global performance utilizing only locally collected data. In many industrial applications, it could be difficult or even unsafe to drive the system rapidly across the operating space, and hence, global data is often not available. Therefore, the ability to train the global network using local data is an important benefit, which is, together with the measure of uncertainty, the main advantage of the proposed network in comparison to other local model networks.

Research published in [41] provides a basis for the work presented in this paper. It gives a broader overview on Bayesian approaches to modeling and demonstrates some possible utilizations of GPs for both nonlinear modeling and control. This paper, however, focuses primarily on the merging of local model principles with the GPs methodology. A brief overview of GPs is given in Section II. A local model network structure identification algorithm using the GP model is proposed in Section III. A local LGP model as a building block of the local LGP model network is then introduced in Section IV. A realization of the local linear GP network is discussed in Section V. The differences between blending the parameters and blending the outputs of the local LGP models are explained. A novel algorithm for network structure optimization, based on the estimate of the uncertainties of individual local LGP models, is proposed in Section VI. In Section VII, the proposed modeling algorithm is successfully applied to identification of a nonlinear model of a laboratory scale process rig.

II. Gaussian Process

A stochastic process is an indexed collection of random variables. It is defined by properties of the joint distribution for this collection. Thus, any finite set of random variables $\underline{y}$, which has a joint Gaussian distribution

$$P(\underline{y}|\mathbf{C},\mathbf{\Phi}) = \frac{1}{Q}e^{-(1/2)(\underline{y}-\underline{\mu})^T\mathbf{C}^{-1}(\underline{y}-\underline{\mu})} \tag{1}$$

is a GP. Here, $\mathbf{C}$ is the covariance matrix of the data, defined by the covariance function, Q is the normalizing constant, $\mathbf{\Phi}$ is any collection of inputs, and $\underline{\mu}$ is the mean vector of the distribution. The GP is fully represented by its mean vector and covariance function $C(\cdot)$, which defines a covariance matrix $\mathbf{C}$. In this paper, a zero-mean distribution is assumed

$$P(\underline{y}|\mathbf{C},\mathbf{\Phi}) = \mathcal{N}(0,\mathbf{C}). \tag{2}$$

Obviously, not all data can be modeled as a zero-mean process. If the data is properly scaled and detrended, then this assumption about the zero-mean distribution is correct [20].

Consider a noisy input/output set of data $\mathcal{D}$. The full matrix $\mathbf{\Phi}$ of Nd-dimensional input vectors $\underline{\psi}(k)$ is constructed as follows:

$$\mathbf{\Phi} = \begin{bmatrix} \psi_1(1) & \psi_2(1) & \dots & \psi_d(1) \\ \psi_1(2) & \psi_2(2) & \dots & \psi_d(2) \\ \vdots & \vdots & \vdots & \vdots \\ \psi_1(k) & \psi_2(k) & \dots & \psi_d(k) \\ \vdots & \vdots & \vdots & \vdots \\ \psi_1(N) & \psi_2(N) & \dots & \psi_d(N) \end{bmatrix}. \tag{3}$$

Scalar outputs are arranged in the output vector $\underline{y}_N$

$$\underline{y}_N = [\,y(1) \quad y(2) \quad \dots \quad y(k) \quad \dots \quad y(N)\,]^T. \tag{4}$$

The aim is to construct the model and then at some new input vector

$$\underline{\psi}(N+1) = [\,\psi_1(N+1) \quad \psi_2(N+1) \quad \dots \quad \psi_d(N+1)\,] \notin \mathcal{D} \tag{5}$$

find the distribution of the corresponding output $y(N+1)$. A general model for the kth output can be written as

$$y(k) = y_f\left(\underline{\psi}(k)\right) + \eta(k) \tag{6}$$

where $y(k)$ is a noisy output, y_f is the modeling function which produces noise-free output from the vector $\underline{\psi}(k)$, and $\eta(k)$ is additive noise. Assuming that additive noise is Gaussian and the output vector $\underline{y}_N$ is generated by a GP [see (1)], then

$$\hat{y}(N+1) \sim \mathcal{N}\left(\mu_{\hat{y}(N+1)}, \sigma^2_{\hat{y}(N+1)}\right). \tag{7}$$

From the system identification point of view, the GP model can be interpreted as follows.

Given an input vector $\underline{\psi}(N+1)$, the predicted model output $\hat{y}(N+1)$ is the mean of the Gaussian distribution $\mu_{\hat{y}(N+1)}$. The model uncertainty about this prediction is then defined by $\sigma_{\hat{y}(N+1)}$. This uncertainty or level of confidence in the model can also be represented as the error bars about the model prediction.

The estimate of the uncertainty in model prediction provided by the GP model is a key advantage of this approach in comparison with NNs and other parametrical models.

The components of the predictive distribution (7) are defined as

$$\mu_{\hat{y}(N+1)} = \underline{\vartheta}^T \mathbf{C}^{-1} \underline{y}_N \tag{8}$$

$$\sigma^2_{\hat{y}(N+1)} = \nu - \underline{\vartheta}^T \mathbf{C}^{-1} \underline{\vartheta} \tag{9}$$

where the covariance matrix $\mathbf{C} \in \Re^{N\times N}$, vector $\underline{\vartheta} \in \Re^N$, and the scalar ν can be constructed as follows:

$$\mathbf{C}_{m,n} = C\left(\underline{\psi}(m), \underline{\psi}(n)\right) \tag{10}$$

$$\underline{\vartheta} = \left[C\left(\underline{\psi}(1), \underline{\psi}(N+1)\right) \ldots C\left(\underline{\psi}(N), \underline{\psi}(N+1)\right)\right]^T \tag{11}$$

and

$$\nu = C\left(\underline{\psi}(N+1), \underline{\psi}(N+1)\right). \tag{12}$$

Here, $C(\cdot)$ is the covariance function (in the machine learning literature this can be called a kernel function) and is a function of inputs only. Any choice of the covariance function, which will generate a positive–definite covariance matrix for any set of input points, can be chosen [26]. If the covariance matrix $\mathbf{C}$ is not positive definite, the distribution (1) cannot be normalized. The flexibility of the covariance function selection offers the ability to include prior knowledge in the model. A sum of positive–definite functions is also a positive–definite function as well as product of positive–definite functions is also a positive–definite function. These properties give the ability to construct complex covariance functions from a set of simple functions.

The covariance function (13) has proven to work well and it has been widely used in practice

$$C\left(\underline{\psi}(m), \underline{\psi}(n)\right) = \theta_0 e^{-(1/2)\sum_{l=1}^{d} \theta_l \left(\underline{\psi}_l(m) - \underline{\psi}_l(n)\right)^2} + \theta_\eta \delta(m,n). \tag{13}$$

Here, $\underline{\theta} = [\theta_0 \quad \theta_1 \quad \ldots \quad \theta_d \quad \theta_\eta]^T$ is the vector of hyperparameters, d is the dimension of the input space, and $\delta(m,n)$ is a Kronecker delta defined as

$$\delta(m,n) = \begin{cases} 1, & \text{for } m = n \\ 0, & \text{for } m \neq n \end{cases}. \tag{14}$$

This covariance function assigns a similar prediction to input points which are close together. The θ_l parameter allows for a different distance measure in each input dimension. Parameter θ_0 controls the overall scale of the local correlation. The hyperparameter θ_η is the estimate of the noise variance. The additive noise is assumed to be white, zero-mean, and Gaussian. If the noise is correlated, then the noise model $\theta_\eta \delta(m,n)$ can be modified as shown in [34].

A. Training a GP Model

To be able to make a prediction, using (8) and (9), the hyperparameters have to be provided either from prior knowledge or by supervised training. Both Monte Carlo [27] and maximum likelihood[1] approaches have been widely used to train the hyperparameters. The Monte Carlo approach is very flexible, but requires large memory storage for the large matrices involved. For small data sets, where matrix storage is not an important issue, the Monte Carlo approach will give better results for fixed central processing unit (CPU) time. For larger data sets, the maximum-likelihood approach is faster and the trained model performs better [20], [25].

In the maximum-likelihood framework, the hyperparameters are optimized using standard gradient-based optimization routines such as the conjugate gradient technique. This type of optimization optimizes the negative log-likelihood function

$$\mathcal{L} = -\frac{1}{2}\left(\log\left(\det\left(\mathbf{C}_N\right)\right) + \underline{y}_N^T \mathbf{C}_N^{-1} \underline{y}_N + N\log\left(2\pi\right)\right) \tag{15}$$

using only the derivatives of $\mathcal{L}$ with respect to the hyperparameters and evaluation of the function itself is not required. Such gradient-based techniques are sensitive to the initial choice of the hyperparameters, and hence, multiple random restarts of the algorithm may be required to avoid the problem of local minima. This technique is heavily time-consuming. Rasmussen [25] suggests a single run, allowing a fixed number (about 150) of evaluations, by which time the likelihood is changing very slowly. Inappropriate choice of initial values makes the partial derivatives of the likelihood small, which will create a problem for the optimization algorithm. Choice of initial values $\theta_0 = 1$, $\theta_i = (1/d)$, and $\theta_\eta = e^{-2}$ suggested in [25] has proven to work well. Since the hyperparameters themselves must be always positive, the initial values $\underline{\theta}_{ini}$ can be chosen as $\underline{\theta}_{ini} = \log(\underline{\theta})$, where the log is applied elementwise $\underline{\theta}_{ini} = [\log(\theta_0) \quad \log(\theta_1) \quad \ldots \quad \log(\theta_d) \quad \log(\theta_\eta)]^T$. In this case, unconstrained optimization can be used.

1) Computation Load: The GP approach for modeling nonlinear systems from data is straightforward, with the main drawback being the computation cost.

A single prediction from a GP model, based on N data points, requires the inversion of an $N \times N$ covariance matrix.

[1]Also known as evidence maximization.

The training of the GP model also requires the inversion of the covariance matrix, which makes it time-consuming for large training sets. The computation load associated with the training of the model also increases with the number of hyperparameters. Since the number of hyperparameters is determined by the dimension of the input space, it is, therefore, desirable to choose the dimension of the input space to be as low as possible.

In many cases, an identification task is a reconstruction of the dynamics of the modeled system from input/output samples. It was shown in [42] that in a linear case the order of the model m_m should be selected as $m_m = m_o$, where m_o is the order of the original system. Much work has been done on the estimation of model order of linear systems, if it is unknown [42], [43].

It has been pointed out in [44] that, in order to accurately capture the dynamic of a nonlinear system, a nonminimal realization, with $m_m > m_o$, might be required. The Takens' embedding theorem for forced systems [45] provides a basis for the reconstruction of the unknown dynamics by analyzing the time-series data generated by a nonlinear system. In the noise-free case, a new state–space is constructed from time-series data. This new state–space can provide useful information about the selection of the order of the model. An application of Takens' theorem to the model order selection problem leads to the conclusion that $m_m \geq 2m_o + 1$. Theoretically, in the noise-free case, Takens' theorem provides sufficient conditions for the nonlinear, discrete-time model order selection if the order of the original system is known. However, so far, no reference is made on how to estimate the order of the model in the case when the true order of the original nonlinear system is unknown [46].

From the GP point of view, the selection of the order of the model corresponds to the selection of the dimension of the input space. According to Takens' theorem, the dimension of the input space should be higher than the order of the original system. This increase of dimension of the input space from the minimal realization will dramatically increase the computation load of the GP model.

B. Uncertainty of the GP Model

For parametrical models, the uncertainty is usually expressed as an uncertainty in parameters and does not take into account uncertainty about the model structure and the distance of the current point prediction from the training data used to estimate the model parameters [36]. In an attempt to estimate the model uncertainty, an extension to the RBFN, known as the VI net, was proposed in [47]. Based on the training data density, the VI net has the ability to indicate when the network is extrapolating and to calculate confidence limits for its prediction. In [48], some practical approaches for the measure of confidence for NNs are presented and compared. In this paper, it is considered that the prediction uncertainty depends of the data noise and model parameter mismatch. A number of error estimation methods for machine learning models and, in particular, for NNs are also reviewed in [49].

For the GP model, both the model prediction and its estimate of uncertainty are based on the training data. This allows the model to measure the distance, in the input space, between the input point for which the prediction is made and the training data set. This is important when a nonlinear function is modeled, based on locally collected data. When the model is asked to predict the output on the subspace where the training data is available, the model will produce a relatively accurate prediction, which will also result in a low uncertainty. If the model is asked to predict the output based on the input point which is further away from the well-modeled subspace, the model will produce a prediction with an increased uncertainty. This allows the GP model to be easily applied to outlier and novelty detection tasks [50]. Model uncertainty also depends on the model structure, with a more complex model providing a lower uncertainty. Finally, the ability to incorporate noise information in the model will significantly improve the accuracy of the uncertainty estimate.

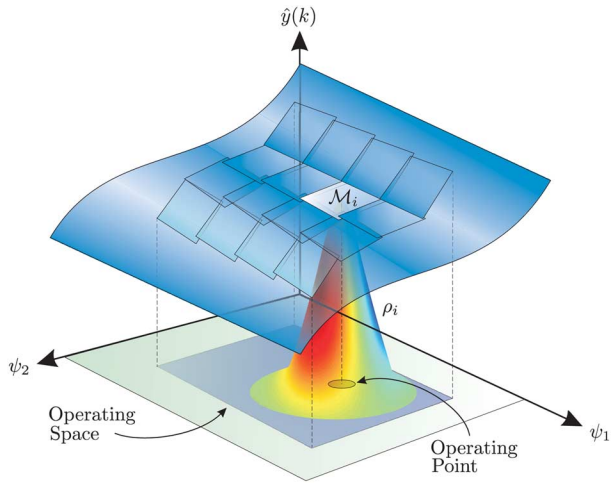

Fig. 1. Representation of the local model network structure.

III. Local Model Network

In multiple-modeling approaches it is assumed that a nonlinear process can be modeled using a weighted combination of local models blended to form a global model [5]. This modeling structure could be represented as

$$\hat{y}(k) = \sum_{i=1}^{M} \rho_i \left(\underline{\phi}(k)\right) \mathcal{M}_i(\underline{\psi}(k)) \tag{16}$$

where $\hat{y}(k)$ is the model output, M is the number of the local models, $\rho_i\left(\underline{\phi}(k)\right)$ is the ith validity function, and $\mathcal{M}_i(\underline{\psi}(k))$ is the ith local model. Here, $\underline{\psi}(k)$ is an input data vector. The scheduling vector $\underline{\phi}(k)$, which defines the operating point of the system, has a dimension which ideally is lower[2] than the dimension of the data vector $\underline{\psi}(k)$[3]. A possible representation of the multiple-model network structure is shown in Fig. 1. The operating space is first partitioned into a number of operating regimes [51]. Although nonlinear local models can be used, the most common choice are local models $\mathcal{M}_i(\underline{\psi}(k))$ which each are a linear hyperplane approximation of the surface at its operating point. A combination or blending of local models into a local model network is usually accomplished by utilization of membership or blending functions. A Gaussian-shaped blending function, equivalent to those used in Gaussian

[2]Scheduling vector $\underline{\phi}(k)$ is usually a subvector of the $\underline{\psi}(k)$.

RBFNs, is a common choice for most local model network applications. A partition of unity across the operating space is usually achieved by the normalization of the blending functions, which often forces the original blending functions to change their shape. This can significantly change their properties and could lead to undesirable side effects. Some of the side effects include loss of uniform shape, shifted maxima from their centers, no guarantee to decrease monotonically as distance from their centers increases, and the reactivation of the blending functions [5]. These reshaped blending functions are known as validity functions, which define the activation level of local models at a particular operating point. The validity functions $\rho_i\left(\underline{\phi}\left(k\right)\right)$ are functions of the scheduling vector $\underline{\phi}\left(k\right)$.

A. Local Model Network Structure Determination Using GP Model

One of the most difficult tasks in the various local modeling approaches is structure determination. How many local models are required and where to place them are the questions for which answers usually are not trivial. In this section, an algorithm for structure determination is proposed. The algorithm is based on a global GP model utilizing the nonlinear covariance function (13). Since this type of model is computational expensive, information provided by the global GP model will be used to construct a local LGP model network. A local LGP model network is in general computationally less expensive with the added benefit that it is more transparent than a global GP model. It is important to note that it is also analytically invertible, which makes it useful for control. Placing the centers, proposed in Section III-B, and clustering the data, proposed in Section III-C, do not only apply to the local LGPs model network, but can be used for structure identification for any type of local model network.

B. Placing the Centers

In the first step of the identification of a local model network model, the centers of the validity functions are placed across the operating space. If a number of specially designed experiments are performed to collect the local data, then this center allocation is not a difficult task. The centers are simply placed at the operating point around which the data was collected. If the data is collected during a real operation of a plant, then placing the centers, especially in the multidimensional case, can be more challenging. In this section, an algorithm for placing the centers of the validity functions for the local model network is proposed.

In many cases, clustering algorithms such as fuzzy clustering and k-means clustering have been used to place the centers of validity functions. A disadvantage of such algorithms is that they are based only on the location of the data in the input space and do not take into account the complexity of the modeled system [52]. The proposed algorithm is based on the predicted variance, computed from the global GP model. Instead of clustering the input space, the centers are placed at the points where the predicted variance has its minimum points.

In the first stage of this algorithm, a global GP model is trained using the nonlinear covariance function and locally collected data. Once a global GP model has been identified, any point from the operating space can be used as a test input. The prediction of the output for the particular test point is not a matter of interest in this case, as the valuable information is the predicted variance. Since the predicted variance is a function of the distance from the particular test point to the training data, the test points close to the region where the density of the training data is high will have a lower variance than the test points away from the training set. If the test points are placed uniformly across each input dimension of the operating space, the regions of operating space where the variance is low can be found. It is correct to assume that, in the regions where the predicted variance is low, accurate local models can be found. It then makes sense that the test points with the lowest variance should be chosen as centers of the validity functions. The precision of the algorithm, and hence, its computation load depends on the density of the test points spanning the operating space. The algorithm, therefore, can be summarized as follows.

Step 1: Find the global GP model using locally collected training data.

Step 2: Place equally spaced test points $t_1 \ldots t_n$ across the operating space.

Step 3: Compute the variance for each test point using a global GP model.

Step 4: Place the centers C_j on the test points with the minimum variance.

The algorithm is illustrated in the following example.
A static nonlinear function

$$f(x) = \tanh(x) + \eta \tag{17}$$

where η is additive noise, was used to generate a training set consisting of three clusters of data, as shown in Fig. 2.

A GP model was then trained and the predicted variance was plotted for the range of inputs $-3 < x < 3$. The minimum points of the predicted variance, where the centers of the validity functions should be placed, are also indicated. In general, these points do not lie on the "centers of mass" of the clusters. However, in cases where the training data points are uniformly distributed across each cluster, the minimum variance points could fall close or on the centers of mass.

The curvature of the predicted variance can also provide information about the initial width of the validity functions, which can be used in the later stage for structure optimization. A point with a sharper minimum should have a smaller initial width and vice versa.

The minimum points of the predicted variance can be easily found in the 1-D or 2-D operating space by visualization. In Section III-B1, an algorithm is proposed to search for the variance minima for any given dimension of the operating space.

1) Minimum Variance Search: This search algorithm is based on a comparison of the predicted variances for each test point of the operating space. First, the boundary around each test point is set. The test points inside the boundary create a cluster. The test point with the lowest variance in the cluster is chosen to

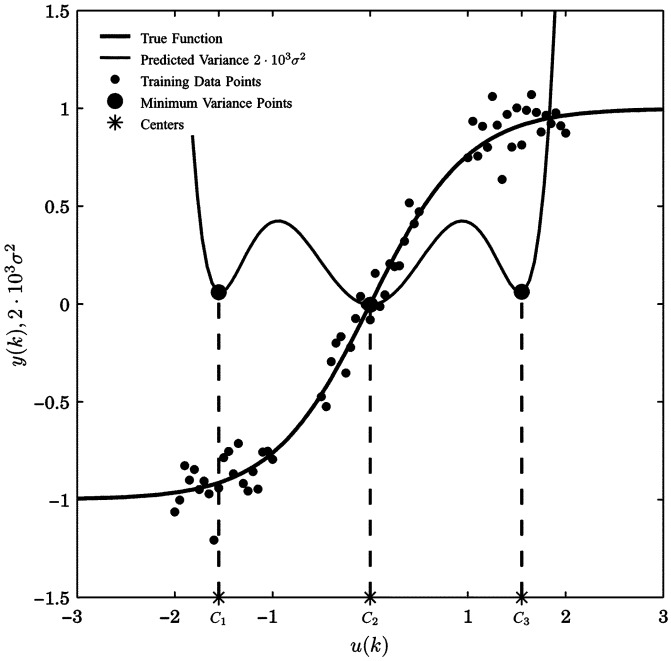

Fig. 2. Centers of the validity functions of the local model network could be placed where the predicted variance has its minimum points. The variance σ^2 is scaled, to improve clarity.

be a cluster winner. The cluster winners whose score is higher than a certain threshold are assumed to be the variance minimum points. The search algorithm, therefore, can be summarized as follows.

Step 1: Pick the test point t_i from the operating space.

Step 2: Create a cluster $\mathcal{C}_i$: Test points which are inside the hypersphere of radius ϵ, centered at the point t_i, define the cluster $\mathcal{C}_i$.

Step 3: Pick two test points from the cluster $\mathcal{C}_i$ and compare their predicted variances. The point with lower variance wins. Repeat until all the winners of each pair of points in the cluster are found. The test point with the highest score in the cluster is the cluster winner $t_{w_{\mathcal{C}_i}}$.

Step 4: Repeat steps 1–3 for all test points.

Step 5: Cluster winners $t_{w_{\mathcal{C}}}$ with a score higher than some threshold $\mathcal{T}$ are chosen to be the minimum variance points t_{m_j}.

Since the algorithm can be computationally expensive, it is advisable to place test points only on the region of the operating space where training data is available.

C. Clustering the Input Data

Once the centers are defined, the input data set can then be clustered. Each input data point belongs to one of the centers. Data points which belong to a particular center create a cluster, which defines the operating regime.

A hypersurface, as shown in Fig. 3, representing how the variance σ_k^2 depending on the data point d_k can be plotted over the input space spanned by the data points. The minimum variance points on this surface are selected as the centers of the clusters. In the example shown, there are two centers C_1 and C_2 with corresponding variance values t_{m_1} and t_{m_2}. In order to determine which cluster a data point d_k belongs to, the Euclidian distance between the point on the variance hypersurface (point σ_k^2 in Fig. 3) above and the minimum points on the hypersurface above the cluster centers (t_{m_1} and t_{m_2}) are calculated. The closest center is then selected. In this way, the cluster selection takes account not just of the proximity of the data point to each center, but also the shape of the variance hypersurface over the operating space.

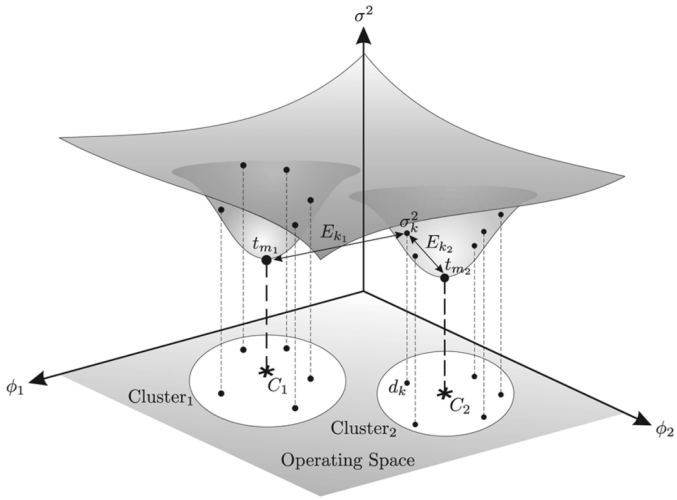

Fig. 3. Clustering of the input data.

IV. Local GP Model

Training of the GP model requires an inversion of the covariance matrix $\mathbf{C} \in \Re^{N \times N}$. The exact inversion of the $N \times N$ matrix has an associated computation cost of $\mathcal{O}(N^3)$, and hence, the time T_N required for training a GP model increases dramatically with the number of training data points N. The entire training data set can be divided into M subsets, for each of which a simpler local model can be identified [18], [44], [53]. If the ith subset consists of N_{S_i} training points, where $N_{S_i} < N$, then the associated computation time required for the training of the ith local model would be $T_{N_{S_i}} \ll T_N$. In order to cover an entire operating space, M local models have to be found, one for each of the M subsets of data. Taking into account the time required for clustering the training data T_{cl} and assuming that each subset contains approximately the same number of data points, the time required for training all local models becomes $T_{N_{S_M}} = T_{cl} + \sum_{i=1}^{M} T_{N_{S_i}}$, which significantly reduces the computation load. The overall time $T_{N_{S_M}}$ can be further reduced if the local models are trained simultaneously, using parallel computation [54]. These simpler local models are then blended to form a global network [5]. The drawback of this divide and conquer approach is that the operating space has to be partitioned by utilizing an appropriate clustering algorithm, a number of local models has to be found, and the network structure needs to be defined. A possible approach to deal with

some of these difficulties is to utilize the algorithm for the network structure determination and data clustering proposed in Section III-A. The network structure can then be further improved, as proposed in Section VI.

If local models are chosen to be GP models, then in general, any parametrization of the covariance function can be utilized. In [55], a standard nonlinear covariance function was used and then local models were combined to form a network. The advantage of this type of network is that it reduces the size of covariance matrices to the size of the training data for the individual subspace. This type of local model is very flexible, but since a nonlinear covariance function is used, it is still a black-box model. This means that apart from the estimate of the variance of the noise, the hyperparameters themselves do not provide any interpretable information about the underlying system [40]. If a linear covariance function is assumed, then the local GP models will each fit a hyperplane to their respective data set. This LGP, can be seen as a linear local approximation of the nonlinear function for each subset of the input space. This type of GP model is also known as a "linear in parameters GP" model [36]. As discussed in Section II-A1, the linearity of the LGP model also assures a minimal realization. Apart from the convenience of using well-studied order estimation techniques for linear systems, the minimal realization also significantly reduces the computation load.

In Section IV-A, an LGP model is derived and a local LGP network (LGPN) model is introduced.

A. LGP Model

If a linear relationship between input/output data is assumed, the following one-step-ahead prediction model structure can be utilized:

$$\begin{aligned}\hat{y}(k+1) &= a_1 y(k) + \cdots + a_p y(k-p+1) \\ &\quad + b_1 u(k) + \cdots + b_q u(k-q+1) + z + \eta(k) \\ &= \underline{\psi}(k)^T \underline{\Theta} + \eta(k) \end{aligned} \tag{18}$$

where (19) and (20), shown at the bottom of the page, hold, and $\eta(k)$ is assumed to be white zero-mean Gaussian noise of variance σ_η^2. If the noise is not white, the covariance function can be adapted to the noise model, as shown in [34]. The number of delayed inputs and outputs used in the model and the offset is denoted as q, p, and z, respectively. Assuming that parameters $a_1, \ldots, a_p, b_1, \ldots, b_q, z$ have Gaussian distribution with zero mean and variances $\sigma_{a_1}^2, \ldots, \sigma_{a_p}^2, \sigma_{b_1}^2, \ldots, \sigma_{b_q}^2, \sigma_z^2$, the covariance function of the LGP model can be written as follows [20], [56]:

$$C_L\left(\underline{\psi}(m), \underline{\psi}(n)\right) = \underline{\psi}(m)^T \mathbf{C}_\Theta \underline{\psi}(n) \tag{21}$$

where

$$\mathbf{C}_\Theta = \operatorname{diag}\left[\, \sigma_{a_1}^2 \quad \ldots \quad \sigma_{a_p}^2 \quad \sigma_{b_1}^2 \quad \ldots \quad \sigma_{b_q}^2 \quad \sigma_z^2 \,\right]. \tag{22}$$

Note that $\sigma_{a_1}, \ldots, \sigma_{a_p}, \sigma_{b_1}, \ldots, \sigma_{b_q}, \sigma_z$ can be seen as the hyperparameters and they have to be provided or trained from the data.

The covariance matrix $\mathbf{C}$ and vector $\underline{\vartheta}^T$ can then be written as

$$\mathbf{C} = \mathbf{\Phi} \mathbf{C}_\Theta \mathbf{\Phi}^T + \mathbf{I}\sigma_\eta^2 \tag{23}$$

$$\underline{\vartheta}^T = \underline{\psi}(\kappa)^T \mathbf{C}_\Theta \mathbf{\Phi}^T \tag{24}$$

where $\mathbf{I}$ is the identity matrix. The mean of the predictive distribution can be written as

$$\begin{aligned}\mu_L &= \underline{\psi}(\kappa)^T \mathbf{C}_\Theta \mathbf{\Phi}^T \mathbf{C}^{-1} \underline{y}_N \\ &= \underline{\psi}(\kappa)^T \underline{\Theta}_L \end{aligned} \tag{25}$$

where $\underline{\psi}(\kappa)^T$ is a new test input and $\underline{\Theta}_L$ is a constant vector, which is a function of the training data. Note that the symbol κ is used to denote the discrete time to emphasize that data points taken at κth sample do not belong to the training set taken at samples $1 \ldots k \ldots N$. The variance of the predictive distribution can then be written as

$$\begin{aligned}\sigma_L^2 &= \underline{\psi}(\kappa)^T \mathbf{C}_\Theta \underline{\psi}(\kappa) - \underline{\psi}(\kappa)^T \mathbf{C}_\Theta \mathbf{\Phi}^T \mathbf{C}^{-1} \mathbf{\Phi} \mathbf{C}_\Theta^T \underline{\psi}(\kappa) \\ &= \underline{\psi}(\kappa)^T \mathbf{V}_L \underline{\psi}(\kappa) \end{aligned} \tag{26}$$

where

$$\mathbf{V}_L = \mathbf{C}_\Theta - \mathbf{C}_\Theta \mathbf{\Phi}^T \mathbf{C}^{-1} \mathbf{\Phi} \mathbf{C}_\Theta^T. \tag{27}$$

Here, σ_L^2 depends on a new input vector $\underline{\psi}(\kappa)^T$ and the covariance matrix $\mathbf{V}_L$. The covariance matrix $\mathbf{V}_L$ is a function of the input training data and does not depend on the target vector $\underline{y}_N$. This means that the variance σ_L^2, and hence, the standard deviation σ_L is a valid measure of locality of the model. This measure of locality of the local model is related to the uncertainty of the model, or the level of confidence in the prediction. It is convenient to present the level of confidence as "error bars" or an "envelope" around the prediction. The prediction uncertainty envelope is then defined as

$$\varepsilon_L = \mu_L \pm \sigma_L. \tag{28}$$

The LGP model can be seen as a linear approximation of the underlying function. In order to model a nonlinear function, the operating space needs to be partitioned and a number of local GP models needs to be combined to form a local model network [5], [18], [44], [53].

$$\underline{\psi}(k) = \left[\, y(k) \quad \ldots \quad y(k-p+1) \quad u(k) \quad \ldots \quad u(k-q+1) \quad 1 \,\right]^T \tag{19}$$

$$\underline{\Theta} = \left[\, a_1 \quad \ldots \quad a_p \quad b_1 \quad \ldots \quad b_q \quad z \,\right]^T \tag{20}$$

V. Local LGP Network

A. Related Work—Mixtures of Experts

The GP model and, consequently, the LGP model represent a distribution and hence combining the outputs of the local GP models are similar to the mixture of experts[3] approach [57]. Here, the experts are chosen to be a linear function of the inputs.

A similar approach was taken in [55], where the experts were chosen to be GP models using a nonlinear covariance function, and then, a mixture model was used for modeling the hierarchical structure. For implementing inference, a hybrid Markov chain Monte Carlo algorithm was used. It is common that the mixture of experts approach employs a hierarchical architecture. In the architecture proposed in [53], for example, each expert produces an output, for each input vector. These outputs are then combined to form a tree and they are blended by the gating network outputs. The gating network outputs are scalars calculated from the input vector. This general tree-structured network becomes so-called "generalized linear," when the experts are chosen to be linear with a single-output nonlinearity.

B. Blending the Outputs

A generalized linear network can be modified to form a multiple local model network were the local models are chosen to be LGP models. The output of the network is the weighted sum of outputs of the local models. Using validity functions ρ_i, the weights $w_1, \ldots, w_M$ can be scheduled according to some scheduling vector $\underline{\phi}(\kappa)$, representative of the operating condition

$$w_i = \rho_i\left(\underline{\phi}(\kappa)\right). \tag{29}$$

In this paper, the output predictive distribution is a mixture of Gaussians and it is in general multimodal. Multimodality of the distribution is often a strong indication that the distribution is not Gaussian. The mean and variance of a mixture of Gaussians is given by its first and second moments [55], [57]

$$\mu^* = \sum_{i=1}^{M} w_i \mu_{L_i} \tag{30}$$

$$\sigma^{*2} = \sum_{i=1}^{M} w_i \left(\sigma_{L_i}^2 + \mu_{L_i}^2\right) - \mu^{*2}. \tag{31}$$

Note that the variance of a mixture of Gaussians given in (31) is not the same as the sum of weighted variances

$$\sigma^{*2} \neq \sum_{i=1}^{M} w_i \sigma_{L_i}^2. \tag{32}$$

The network proposed in this section is similar to the architecture proposed in [55]. The principal difference is in the choice of the covariance function of the local GP models. In [55], a standard nonlinear covariance function is chosen, where for the local LGPs network, a linear covariance function is applied. The utilization of the linear covariance function offers improved transparency and provides the possibility of blending parameters of the local models.

[3]A probabilistic model which models a subset of the data is known as an "expert." In this paper, an expert is an equivalent of a local model.

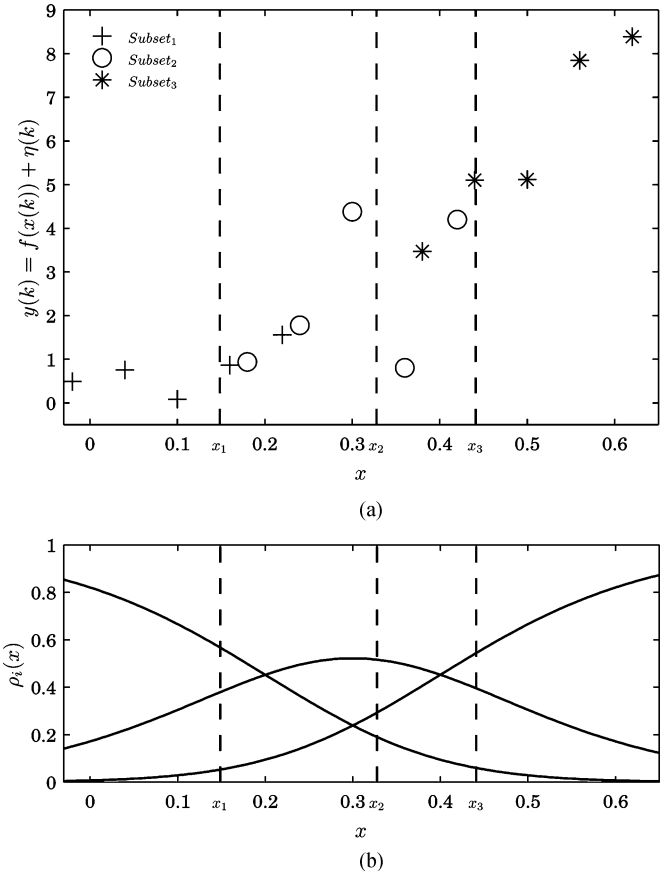

Fig. 4. Training data set used for identification of the linear local GP models. (a) Test points x_1, x_2, and x_3 where a prediction was made are indicated with dashed line. (b) Models validity functions.

C. Blending the Parameters

A linear local GP model is fully defined by the vector of parameters $\underline{\Theta}_L$ and covariance matrix $\mathbf{V}_L$. Given a set of local GP models, it is possible to realize the global model either by the blending of the model outputs or by the blending of the model parameters. An alternative based on the blending of model parameters is given in (33) and (34). In this case, the overall network will have the structure of a single LGP model with a particular choice of weights. Therefore, the predictive distribution will be Gaussian defined by its mean and variance

$$\mu_L^* = \underline{\psi}(\kappa)^T \sum_{i=1}^{M} w_i \underline{\Theta}_{L_i} = \underline{\psi}(\kappa)^T \underline{\Theta}_L^* \tag{33}$$

$$\sigma_L^{*2} = \underline{\psi}(\kappa)^T \left(\sum_{i=1}^{M} w_i \mathbf{V}_{L_i}\right) \underline{\psi}(\kappa) = \underline{\psi}(\kappa)^T \mathbf{V}_L^* \underline{\psi}(\kappa). \tag{34}$$

For a particular choice of weights, the mixture of parameters of the network is defined as a linear combination of the parameters of local models. An observation of the parameters can provide useful information about the system at a particular operating point. This means that when local LGP models are used, the parameters of the network carry information about the linear approximation of the modeled function at that particular operating point. A network using blending of the parameters is,

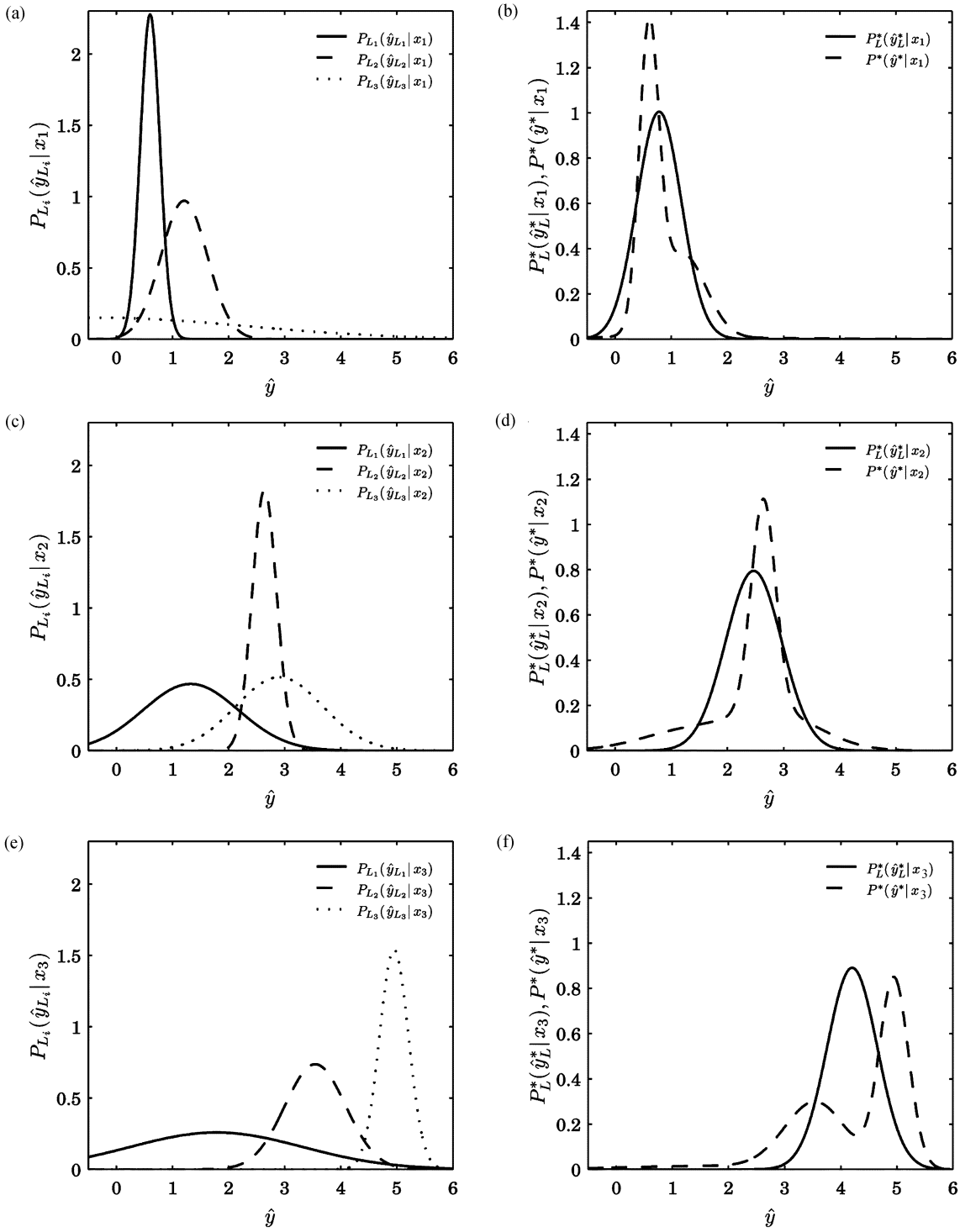

Fig. 5. (a), (c), and (e) Predictive distributions of the linear local GP models at different test inputs. The predictive distributions are Gaussian and each of the local GP models generate a sharp distribution in the region, where it was trained. The remaining models generate a less sharp distribution. (b), (d), and (f) Predictive distributions of the network utilizing the outputs blending and the parameter blending realizations.

therefore, more transparent than the one in which the outputs are blended.

D. Variances σ^{*2} and σ_L^{*2}

In the preceding example, it was evident that the mean of the predictive distribution when the outputs are blended and when parameters are blended are the same. Indeed, when local LGP models are used to form the network, the equality of means can be easily proven

$$\mu^* = \sum_{i=1}^{M} w_i \mu_{L_i} = \sum_{i=1}^{M} w_i \underline{\psi}(\kappa)^T \underline{\Theta}_{L_i} = \sum_{i=1}^{M} \underline{\psi}(\kappa)^T w_i \underline{\Theta}_{L_i} = \underline{\psi}(\kappa)^T \sum_{i=1}^{M} w_i \underline{\Theta}_{L_i} = \mu_L^*. \quad (35)$$

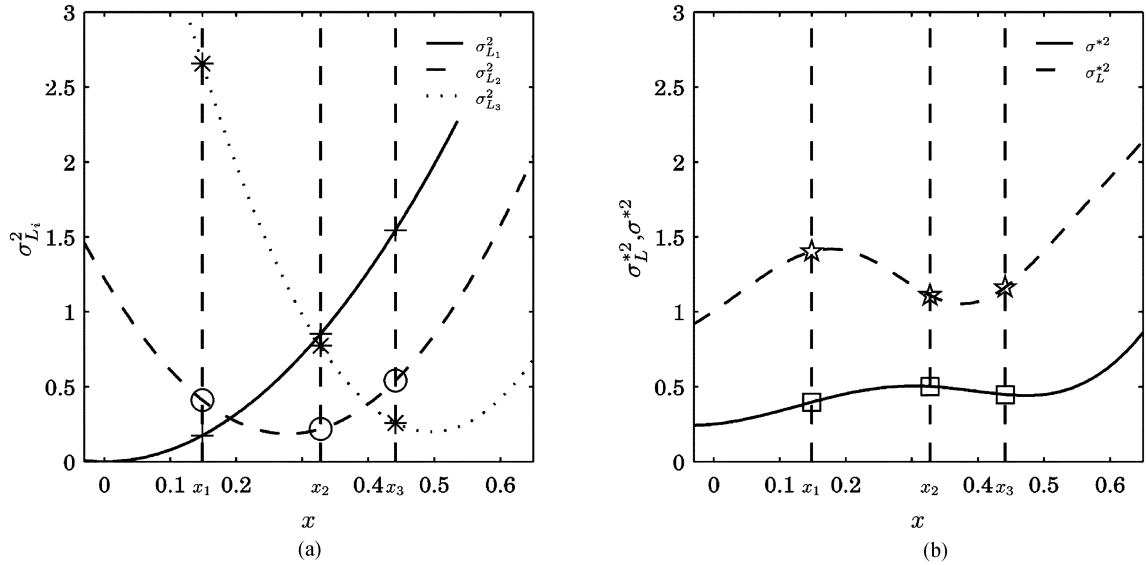

Fig. 6. (a) Variances of the linear local GP models, which indicate the sharpness of the predictive distributions of each local model. (b) Variances of the overall distributions generated by the network utilizing output blending and the network using parameter blending.

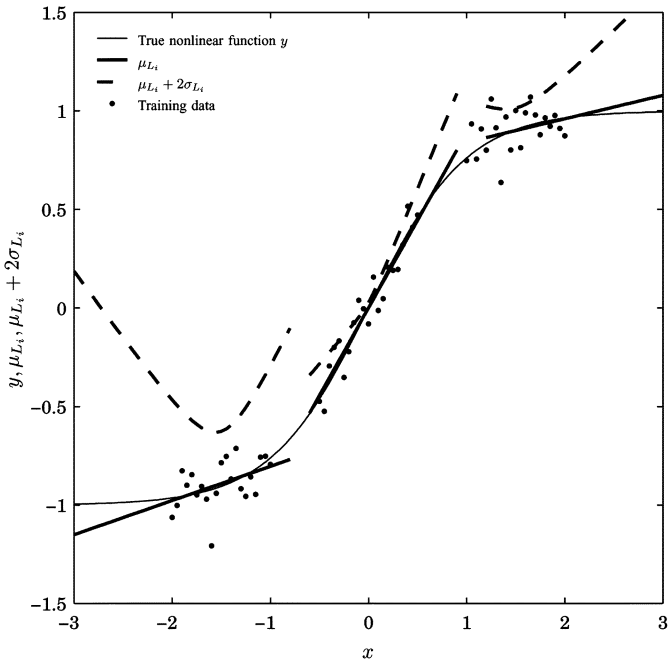

Fig. 7. Three linear local GP models and their uncertainty envelopes estimated at different points.

The variance of (31) using LGP models can be rewritten as

$$\begin{aligned}\sigma^{*2} &= \sum_{i=1}^{M} w_i \left(\sigma^2_{L_i} + \mu^2_{L_i}\right) - \mu^{*2} \\ &= \sum_{i=1}^{M} w_i \sigma^2_{L_i} + \sum_{i=1}^{M} w_i \mu^2_{L_i} - \mu^{*2}_L \\ &= \sum_{i=1}^{M} w_i \underline{\psi}(\kappa)^T \mathbf{V}_{L_i} \underline{\psi}(\kappa) + \sum_{i=1}^{M} w_i \left(\underline{\psi}(\kappa)^T \underline{\Theta}_{L_i}\right)^2 - \mu^{*2}_L \\ &= \underline{\psi}(\kappa)^T \mathbf{V}^*_L \underline{\psi}(\kappa) + \underline{\psi}(\kappa)^T \left(\sum_{i=1}^{M} w_i \underline{\Theta}_{L_i} \underline{\Theta}^T_{L_i}\right) \underline{\psi}(\kappa) - \mu^{*2}_L \\ &= \sigma^{*2}_L + \Omega^{*2} - \mu^{*2}_L. \end{aligned} \tag{36}$$

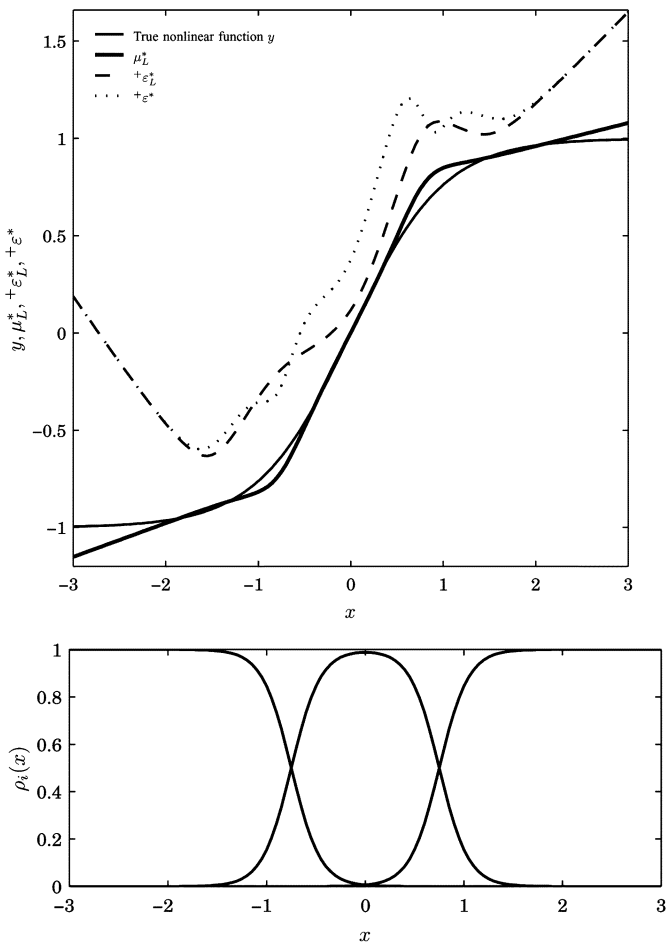

Fig. 8. Given validity functions ρ_i shown on the lower plot, the envelope of the network where the outputs are blended does not provide information about locality of the local models. Envelope, where the parameters are blended is a valid measure of locality.

Equation (36) shows that the variance of the predictive non-Gaussian distribution, generated by output blending, is a function of the variances σ^{*2}_L and predictions μ^*_L

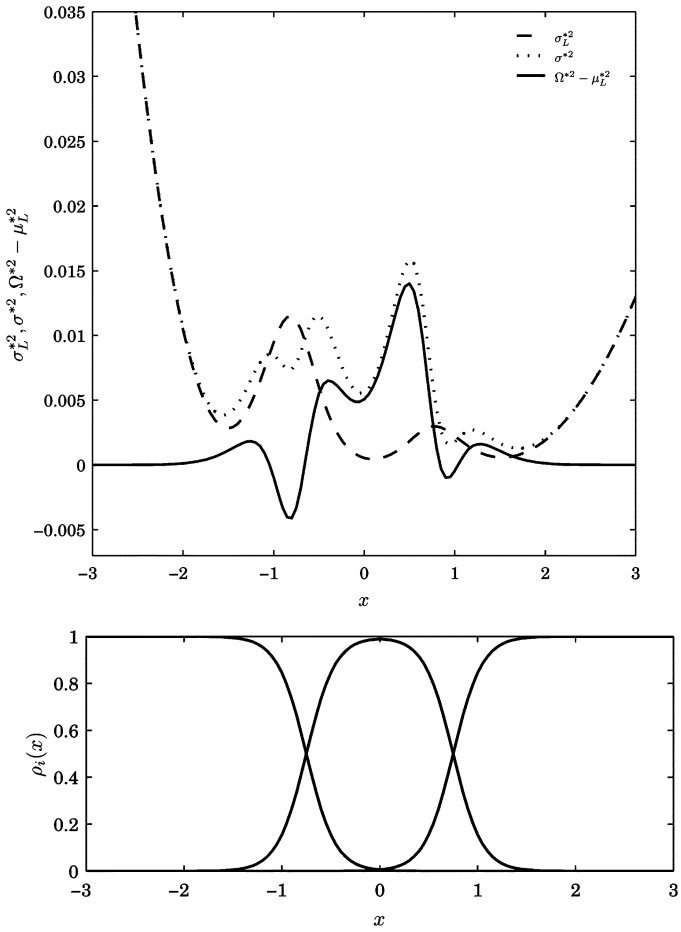

Fig. 9. At the splits of the validity functions, the contribution of local predictions $\Omega^{*2} - \mu_L^{*2}$ is large and corrupts the locality information of σ^{*2}.

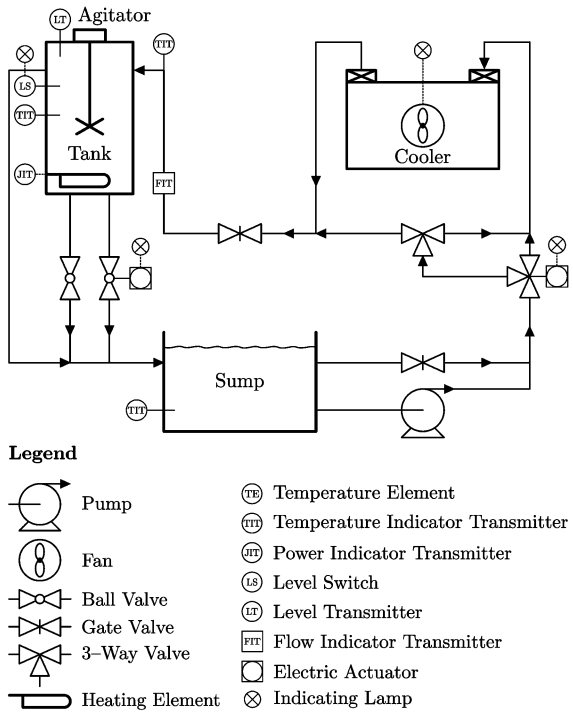

Fig. 10. Block diagram of the process rig.

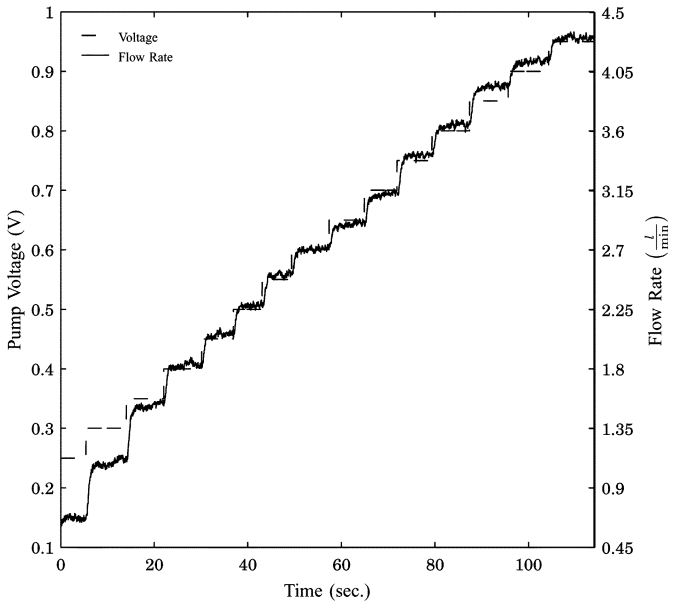

Fig. 11. Flow rate response across the operating space.

of the constituent local models and the term $\Omega^{*2} = \underline{\psi}(\kappa)^T \left(\sum_{i=1}^{M} w_i \underline{\Theta}_{L_i} \underline{\Theta}_{L_i}^T\right) \underline{\psi}(\kappa)$. The variance of each local model depends strictly on the input training points. However, the predictions and Ω^{*2} are a function of both input training points and the targets. The variance (34), generated by parameter blending, is a linear function of local variances, and hence, a function of only the input training points and not the targets.

To illustrate the difference between the network architectures, consider the following example.

Three LGP models were trained from samples of data $\mathcal{D} = \{x(k), y(k)\}, k = 1 \ldots 15$, shown in Fig. 4(a). Each local model was trained from the local subset of noisy data generated by the static nonlinear function

$$y(k) = f(x(k)) + \eta(k) \tag{37}$$

where η is noise. In Fig. 4, "+" indicates a data point which is a member of first subset, "○" indicates a member of second subset, and "×" is a member of the third subset. Fig. 4(b) shows the validity functions $\rho_i(x)$, $i = 1, 2, 3$, of the local models, which determine the values of the weights w_i. A prediction was made at three test input data points x_1, x_2, and x_3, indicated with dashed lines. Fig. 5(a), (c), and (e) shows the predictive distribution (output) of each local LGP model at the selected test points. It can be seen that each distribution is Gaussian with a particular mean and variance, depending on the position of the test input in the operating space. The local model trained from the subset of data, which was the closest to the given test input, results in the sharpest distribution.

Fig. 5(b), (d), and (f) shows the predictive distributions, generated from the network when the outputs are blended as well as the distribution when the parameters are blended. It can be seen that when the outputs are blended that the predictive distribution is not Gaussian. This distribution is multimodal and has a different shape for each test point. Any conclusion about the prediction from the shape of the multimodal distribution is difficult to make. The distribution generated from the network when the

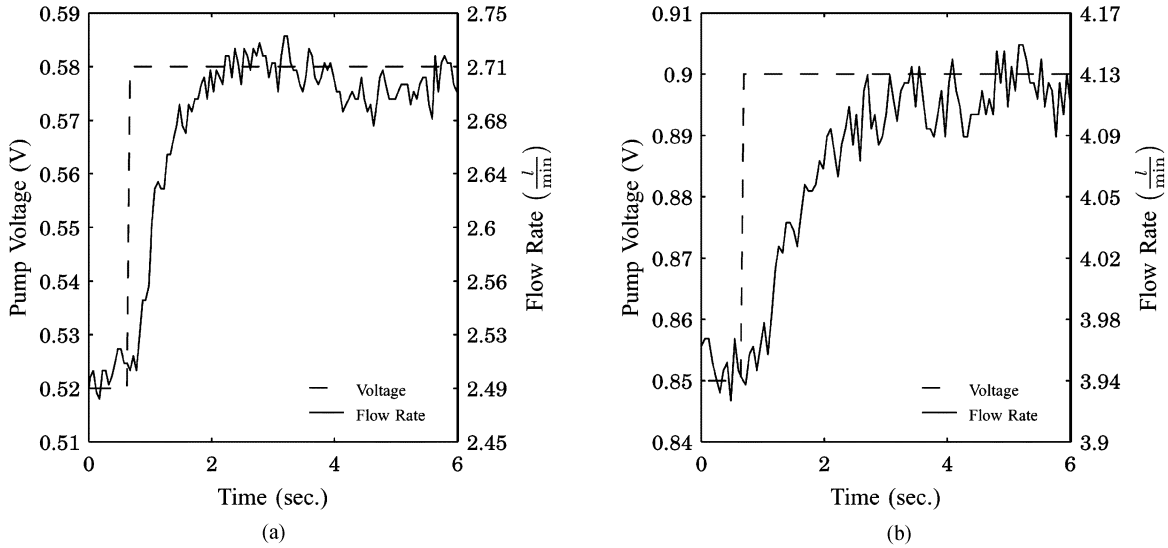

Fig. 12. Flow rate response where the pump voltage was stepped: (a) from 0.52 to 0.58 V and (b) from 0.85 to 0.9 V.

parameters are blended is Gaussian for each test input. When the Gaussian distribution is sharper, the prediction is more confident.

Fig. 6(a) shows the variances of the predictive distribution of each local model. The variance of the particular local model is lower in regions where the local model was trained. Therefore, the variance of each local model can be interpreted as a measure of its locality. This statement holds since the variance of each local model depends on the input training data. The variance for the particular local model will then be lower in the region populated by input training points. Fig. 6(b) shows the variance σ^{*2} of the predictive distribution generated by output blending along with the variance σ_L^{*2} generated by parameter blending. It can be seen that the variances are different. Variance σ^{*2} does not provide any information about the shape of the overall prediction distribution, while σ_L^{*2} indicates the sharpness of the output Gaussian distribution.

1) Network Uncertainty and Measure of Locality: It was shown in Section II that the variance, and hence, the standard deviation of the predictive Gaussian distribution define the uncertainty of the model prediction. Since the variance of the Gaussian distribution depends on input training data and does not depend on the targets, it was shown in Section V-D that this variance can be also interpreted as a measure of the locality of the model.

The variance of the non-Gaussian distribution is given as the second moment of the distribution. When the local model network with blended outputs is utilized, the variance of the predictive distribution is given in (36). Equation (36) clearly shows that the variance σ^{*2} is not only dependent on the training input points and a test point, but also on the predictions of the local models. The predictions of the local models can be far away from the training inputs.[4] The variance, and hence, the standard deviation of the non-Gaussian distribution are not therefore a measure of locality.

Equations (31) and (34) also show that both variances σ^{*2} and σ_L^{*2} are functions of weights. Information about the variance of the network can, therefore, be used to help select the validity functions, which will generate the weights, in order to minimize the uncertainty of the network.

2) Network Envelope: The error envelope (or error bars) is a representative measure of the model uncertainty [29]. Extending (28) representing the envelope of a single LGP local model, the envelope for the network can be defined as

$$\varepsilon^* = \mu^* \pm \sigma^* \tag{38}$$

$$\varepsilon_L^* = \mu_L^* \pm \sigma_L^* \tag{39}$$

where ε^* is the uncertainty envelope of the network, where the outputs are blended and ε_L^* is the envelope of the network where the parameters are blended. For each network, the upper envelope can be defined as

$$^{+}\varepsilon^* = \mu^* + \sigma^* \tag{40}$$

$$^{+}\varepsilon_L^* = \mu_L^* + \sigma_L^*. \tag{41}$$

Equations (29)–(34) and (38)–(41) show that the envelope is a function of the prediction of the local models, their validity functions, and the estimate of their standard deviations. Therefore, the envelope is a combination of the orientation of the local plane and its uncertainty. If the position of the local models is fixed and if the parameters of local models are known in advance, then minimizing the envelope by varying the widths of validity functions will result in a model, which will have the envelope as close as possible to its prediction. This "envelope tightening" will also produce the best fit to the data.

VI. Network Structure Optimization

In network optimization, the widths of the validity functions should be optimized so that the local models are valid in the portion of the operating space where they were identified, while providing the best possible combination of local models between the well-modeled regions. The envelope-tightening algorithm can be used to optimize the widths of the validity func-

[4]How far away from the training inputs the predictions of the local models are depends of the underlying nonlinear function.

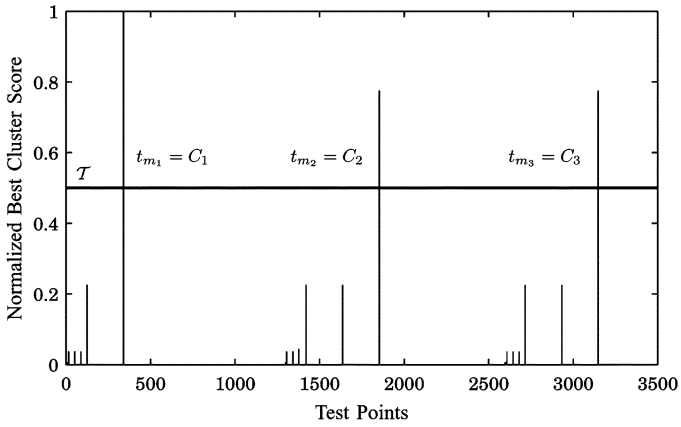

Fig. 13. Graph of the winners cluster score against the test points. The points with the minimum variance are chosen to be the centers of the validity functions.

tions. The crucial part in this algorithm is correct interpretation of network uncertainties.

To highlight the difference between the uncertainties provided by both network structures, consider the following example: Three local LGP models were trained from the noisy data to approximate a nonlinear function, at three operating points. As shown in Fig. 7, the envelopes of the local models are the closest to its prediction when the density of the training data is high. For given validity functions $\rho_i(x)$, it is expected that the network would have higher uncertainty in the region between two models.

Observing the network envelopes given in Fig. 8, it is apparent that the envelope ${}^{+}\varepsilon_L^*$ gives the correct estimate of network uncertainty. ${}^{+}\varepsilon_L^*$ is closer to the prediction in the region, where a single local model is active and it moves away from the prediction in the transition between valid local models. However, the envelope ${}^{+}\varepsilon^*$ is close to the prediction in the transition region, where the highest deviation from the prediction would be expected. The reason lies in the definition of the variances of each network structure. The variance σ_L^{*2} depends only on a weighted sum of local variances, which consequently provide a measure of locality of the local models. The variance σ^{*2} depends of the local variances as well as local predictions. The effect of local predictions corrupts the information about locality.

Fig. 9 shows the plot of σ_L^{*2}, σ^{*2}, and the effect of local predictions $\Omega^{*2} - \mu_L^{*2}$ on σ^{*2}. It can be clearly seen that the effect of local predictions $\Omega^{*2} - \mu_L^{*2}$ overrules the measure of locality at the transition between validity functions. This example clearly shows that the structure of a local model network where the outputs are blended cannot be optimized using the envelope-tightening optimization. Even when validity functions are known, this structure does not provide correct measure of the network uncertainty.

A. Cost Function and Importance of Normalization of the Validity Functions

In order to optimize the structure of the network, the following cost function is proposed:

$$J = \sum_{k=1}^{N} \sum_{i=1}^{M} \rho_i \mu_{L_i}(k) + \rho_i R \sigma_{L_i}(k) \tag{42}$$

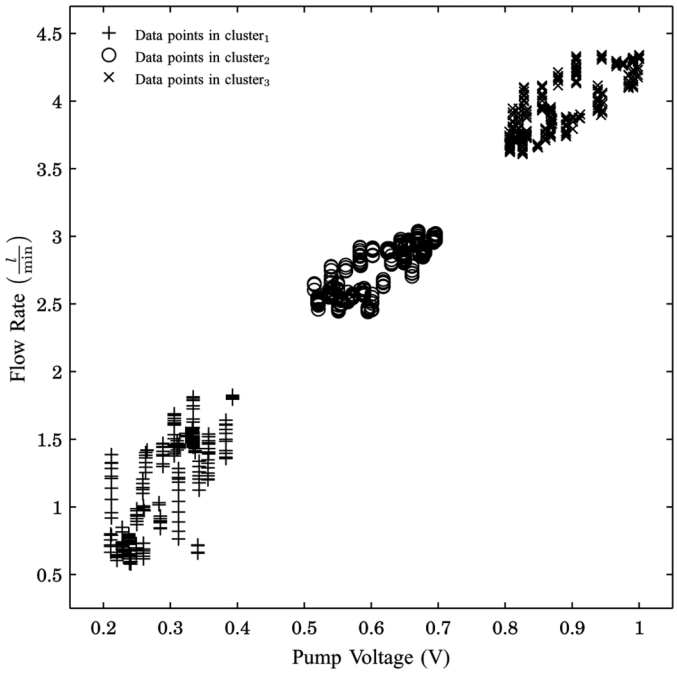

Fig. 14. Clustered training data.

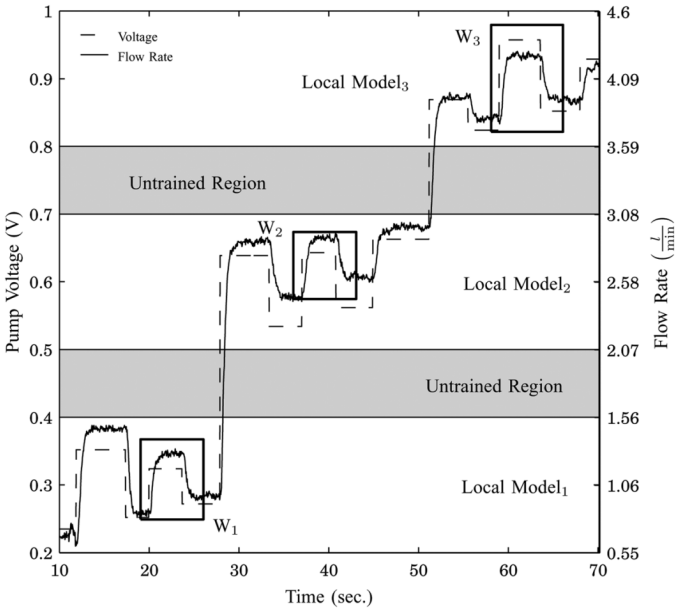

Fig. 15. Validation signal which was used to examine the local models.

where R is the appropriate regularization constant. The impact of the regularization constant R on the optimization algorithm is discussed in Section VII-A3. The validity functions ρ_i in (42) need to be normalized

$$\rho_i = \frac{\widetilde{\rho}_i}{\sum_{j=1}^{M} \widetilde{\rho}_j} \tag{43}$$

where ρ_i and $\widetilde{\rho}_i$ are the ith nonnormalized and normalized validity functions, respectively. Normalization effectively means that across the operating space the sum of the validity function contributions is unity. The denominator of (43) shows that

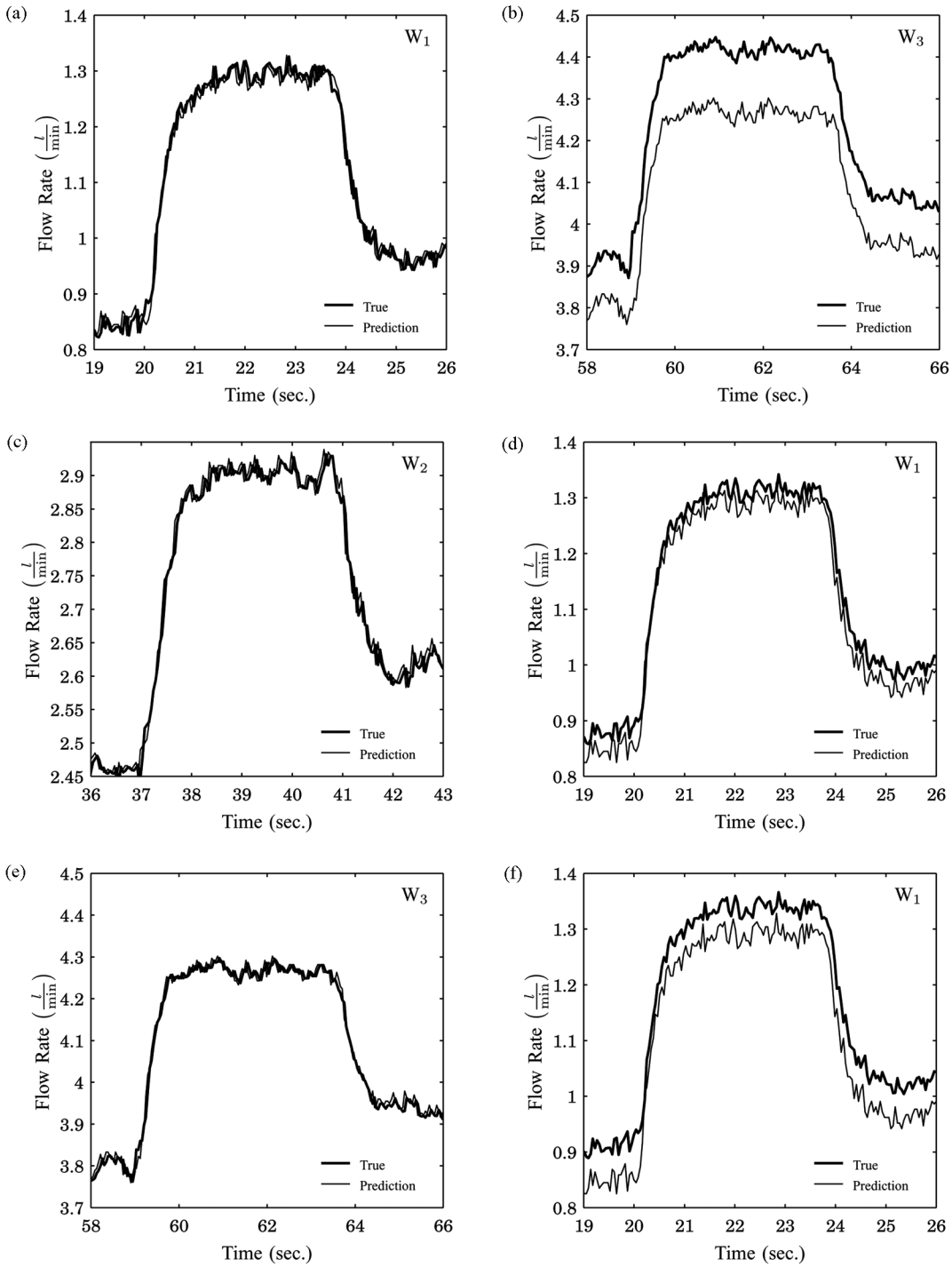

Fig. 16. One-step-ahead prediction of the local models at the different operating points. (a) First local model at the first operating point. (b) First local model at third operating point. (c) Second local model at the second operating point. (d) Second local model at the first operating point. (e) Third local model at the third operating point. (f) Third local model at the first operating point.

the normalization introduces an interaction between the validity functions which will prevent the widths shrinking to zero independently. Side effects of normalization are discussed in [5].

A number of advantages of the network architecture where the parameters are blended over the network can be pointed out as follows.

1) Shape of predictive distribution: When the parameters are blended, the predictive distribution is Gaussian. A Gaussian distribution is more interpretable than the multimodal distribution, generated by the network where the outputs are blended.
2) Network uncertainty: Blending the parameters results in a variance and standard deviation which is a valid measure of network locality, and hence, network uncertainty. The structure of this network can be optimized utilizing the envelope-tightening algorithm. The variance generated by

blending the outputs depends on local predictions which corrupts the information about locality. This is not a valid measure of network uncertainty.

3) Parameterization and transparency: When the parameters of the local models are blended, a model is obtained which represents a linear approximation of the underlying nonlinear system at the operating point. Therefore, global parameters provide a high level of transparency. When the outputs are blended, the global parameters are not available, which makes this type of network a black-box model.

VII. Case Study: Modeling a Laboratory Scale Process Rig

The laboratory scale process rig of Fig. 10 was used to demonstrate the performance of the local LGP model network. The voltage $v(k)$ applied to the pump was chosen as an input to the system. The output is the flow rate $q(k)$, measured by an impeller flow meter, indicated as "FIT" on Fig. 10. The sampling time was 50 ms. The relationship between the voltage $v(k)$ and the pump flow rate could be modeled as a first-order response with a gain, time constant, and an offset, which are all dependent on the operating point. This nonlinearity is due to the geometry of the pump impeller. An additional nonlinearity, which is influenced by the fluid dynamics of the system, can also be identified. Taking into account both nonlinearities, the system can be modeled as a second-order transfer function with operating-point-dependent parameters.

A series of step tests were carried out across the operating space. Fig. 11 shows how the system gain and offset change widely across the operating space.

Two step tests at different operating points are shown in Fig. 12. In Fig. 12(a), the pump voltage was stepped from 0.52 to 0.58 V. At this operating point, the system behaves as a second-order, underdamped system. Fig. 12(b) shows the flow rate response when the pump voltage was changed from 0.85 to 0.9 V. At this operating point, the system response can be modeled as a second-order overdamped response. These responses clearly indicate that the system parameters change with the operating point.

A. Identification of the Local GP Network

1) Placing the Centers and Data Clustering: An additional pseudorandom binary sequence (PRBS) of maximum magnitude of $\pm$ 0.1 V was injected at three operating points, defined by the input voltages of 0.3, 0.6, and 0.9 V. At each operating point, 500 data points were collected. A second-order global GP model with a nonlinear covariance function, as defined in (13), was identified using all available training data.

The predicted variance was then computed for a set of 3500 equally spaced test points spanning the operating space. The algorithm for the minimum variance search, presented in Section III-B1 was utilized to find the centers. The cluster radius was chosen to be $\epsilon = 1$ and the threshold $\mathcal{T}$ was set to $\mathcal{T} = 0.5$. The cluster radius and the threshold were chosen empirically and, as seen in Fig. 13, utilizing $\epsilon = 1$, the algorithm performs well. The high score difference between the minimum points t_{m_1}, t_{m_2}, and t_{m_3} and the remaining cluster winners is significant and indicates that the choice of the threshold $\mathcal{T}$ does not have a major effect on the performance of the algorithm. The minimum points t_{m_1}, t_{m_2}, and t_{m_3} were then chosen to be the centers of the validity functions.

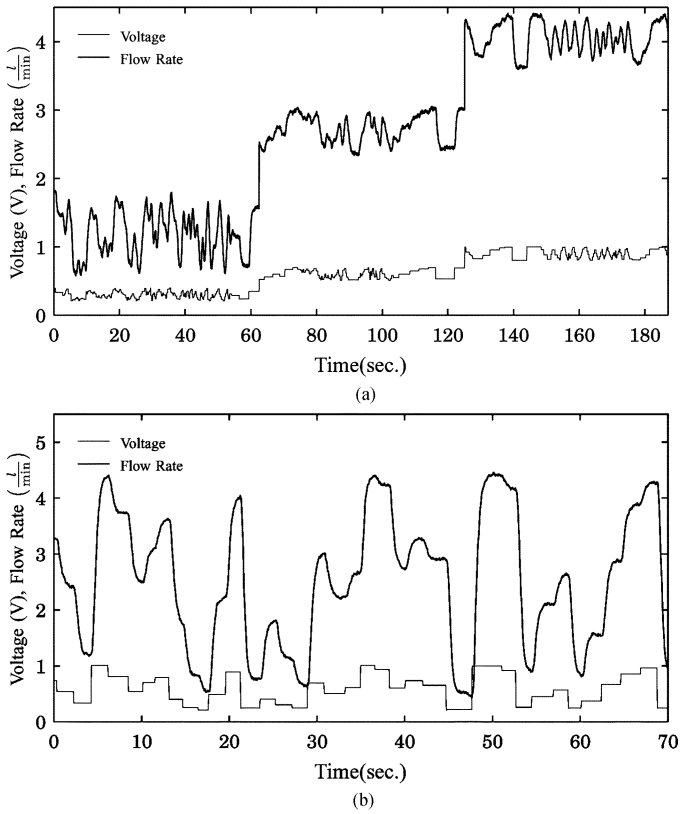

Fig. 17. Training data used for optimizing the validity functions. (a) Local data. (b) Global data.

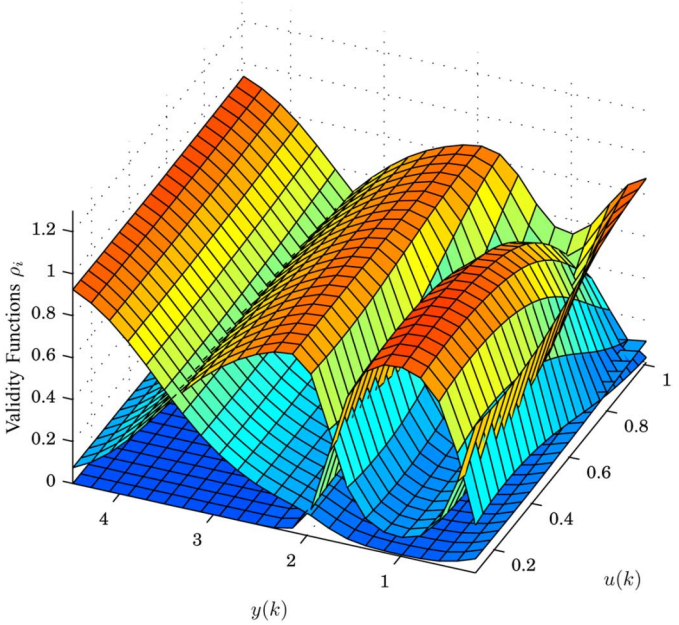

Fig. 18. Optimized validity functions using global data and the MSE minimization.

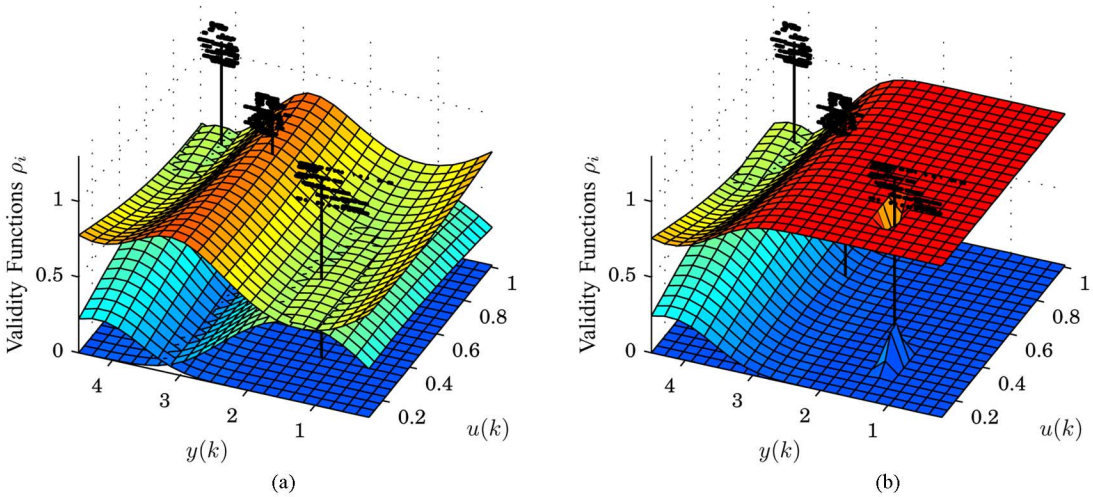

Fig. 19. Optimized validity functions using the minimization of the MSE with local data set. (a) Validity functions overlap too much. (b) Single model dominates across the operating space. The training data and the centers are also shown.

After centers have been placed, each training data point can be associated with one of the centers, using the clustering algorithm presented in Section III-C. The training data set, therefore, forms three clusters, as shown in Fig. 14.

2) Identification and Validation of the Local Models: For each cluster of data, a local LGP model was identified. The hyperparameters of each local model were trained independently, using the maximum-likelihood framework. Here, a standard conjugate-gradient optimization technique was utilized. Multiple random restarts of the optimization were used to deal with the problem of sensitivity to the choice of the initial values of hyperparameters. The validation signal shown in Fig. 15 was used to test the local models.

A series of the step tests was carried out in the well-modeled regions of the operating space. The rectangular regions W_1, W_2, and W_3 denote the sections of the validation signal which were used to examine the accuracy of the local models. The prediction of each local model was assessed using all three data sets. Fig. 16 focuses on the step tests at different operating points denoted by regions W_1, W_2, and W_3 on Fig. 15. The one-step-ahead predictions of the individual local models together with the true responses of the system are shown. It is clear that excellent modeling performance is achieved close to the operating regions where local models were identified. The error of the particular model increases with the distance between the operating point where the model was identified and the actual operating point.

3) Optimizing the Widths of the Validity Functions: Once the local models have been identified and their centers placed, the widths of the validity functions were then chosen to minimize the following cost function:

$$J_{\mathrm{MSE}} = \sum_{i=1}^{M} \sum_{k=1}^{N} \left(y_i(k) - \hat{y}_i(k)\right)^2. \tag{44}$$

To insure a good performance of the global model across the operating space, a global data set is required. This algorithm, in general, may produce validity functions, which overlap to such degree that the contribution of a local model may be experienced well outside its region of accuracy. Effectively, this means that the resulting global model becomes an average of the local models. If global data is not available (which is the case in many of industrial processes), the problem of optimizing the widths becomes even more difficult. Because of the lack of off-equilibrium data, minimizing the mean square error (MSE) does not usually provide a useful structure for the network. Either the validity functions overlap too much and the resulting model is the average of number local models, or, indeed, a single validity functions may dominate across most of the operating space.

The envelope-tightening algorithm proposed in Section VI takes into account the importance of the locality of the validity functions, and hence, prevents domination of a single local model over the entire operating space while also avoiding the averaging of the local models. Two data test sets are collected in order to compare the performance of the envelope-tightening algorithm with the MSE minimization. For the first test set, the system is excited locally at the operating points defined by the input voltages 0.3, 0.6, and 0.9 V. The locally collected data set is shown in Fig. 17(a). In the second test, the system is excited across the whole operating space; the global data set is shown in Fig. 17(b). First, the global data is used to optimize the widths using the MSE minimization. In this paper, the scheduling vector is chosen to be a full data vector $\underline{\phi}(k) = \underline{\psi}(k) = [v(k) \quad q(k) \quad \dot{v}(k) \quad \dot{q}(k)]^T$. The resulting validity functions are shown in Fig. 18. For visualization convenience, the validity functions $\rho_i\left(\underline{\psi}(k)\right)$ are plotted as a surface over the 2-D space spanned by $[v(k) \quad q(k)]^T$.

When the MSE optimization is utilized using local data, the algorithm fails. Since no restrictions regarding the locality of the local models are included in the algorithm, then depending on the initial conditions, the model is either dominated by a single local model [see Fig. 19(a)] or formed by an average of local models [see Fig. 19(b)]. By utilization of the envelope-tightening approach, the cost function of (42) is minimized. Here, the optimization is not as sensitive to the choice of the initial

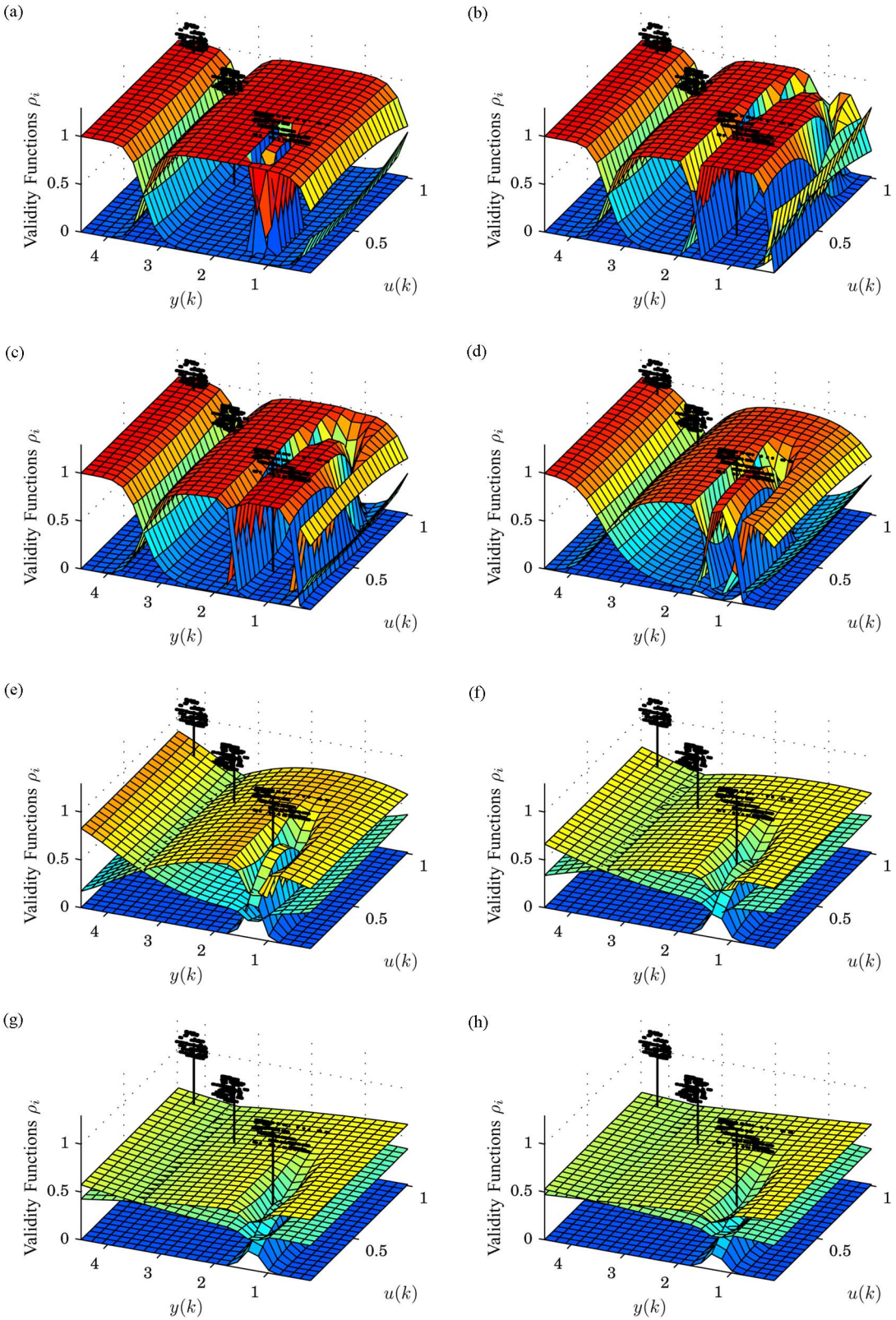

Fig. 20. Model validity functions found using the envelope optimization with different values of R: (a) $R = 0.21$, (b) $R = 0.2694$, (c) $R = 0.3$, (d) $R = 0.5$, (e) $R = 1$, (f) $R = 2$, (g) $R = 5$, and (h) $R = 100$.

condition. The choice of the regularization factor R in (42) however, plays a major role in the performance of the algorithm. R defines a level of impact of the uncertainty, and hence, locality to the curvature of the cost function. Fig. 20 shows the optimized validity functions for several choices of R. When R is too small Fig. 20(a), the second model dominates across the most

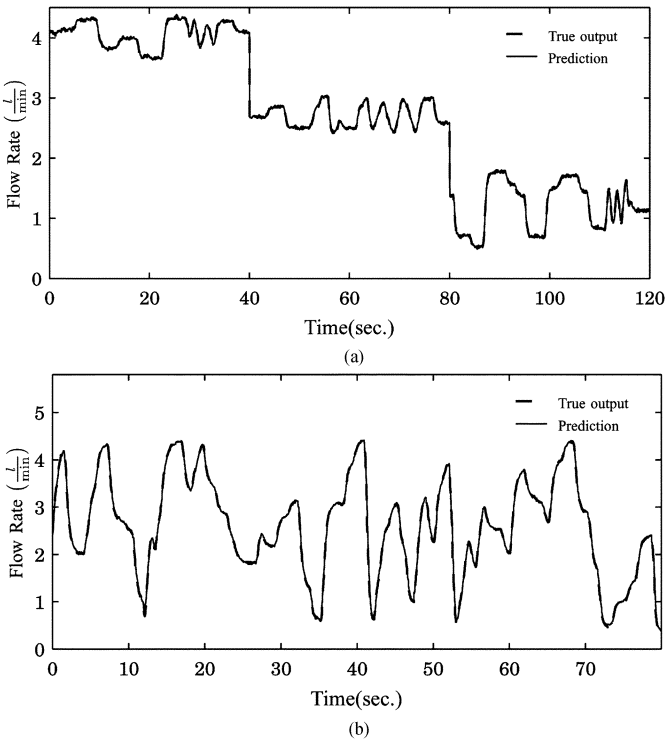

Fig. 21. One-step-ahead prediction of the network. (a) Local validation data. (b) Global validation data.

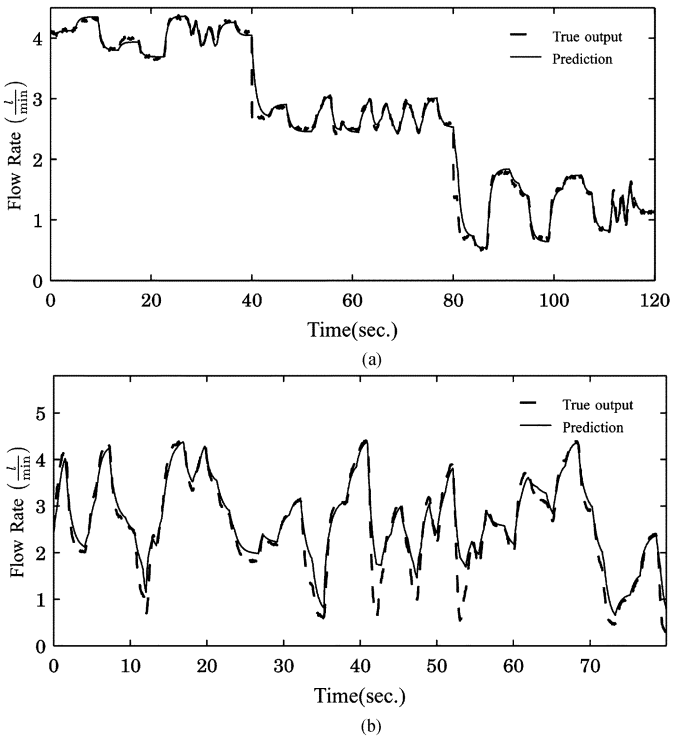

Fig. 22. Prediction of the network utilized as a parallel model. (a) Local validation data. (b) Global validation data.

of the operating space. When R is too large, the averaging effect can be seen clearly in Fig. 20(e)–(h). Fig. 20(b)–(d) shows the optimized validity function where a reasonable value of R is chosen. Since the curvature of the envelope depends on the curvature of the modeled function and the number and accuracy of the local models, it is, therefore, obvious that the correct choice of R will be problem dependent. It is advisable to perform a line search to find a reasonable interval of R, and then, utilize a bisection method to find the optimum value. Here, the optimal choice is $R = 0.2694$. The optimized validity functions, shown in Fig. 20(b), are then used to construct the local LGPN.

B. Validation of the Network

The local linear GP network was validated using both local and global validation data sets. The performance of the network was examined as a one-step-ahead prediction model as well as a parallel model. Fig. 21 shows the one-step-ahead prediction of the network. The prediction matches well with the true output for both the local and the global validation data.

In Fig. 22, the prediction of the network utilized as a parallel model is shown. From Fig. 22(a), it is clearly observed that there is only small mismatch between the model and the true output. This mismatch which can be more clearly seen in Fig. 22(b) is partly due to a loss of accuracy in the nontrained region of the operating space, and off-equilibrium behavior, when the operating point changes rapidly. It is important to highlight here that the widths of the network were trained utilizing the envelope-tightening optimization using only local collected data. This example shows that when only locally collected data is available the structure optimization based on the minimization of the MSE may not be successful. On the other hand, it is shown that the structure optimization utilizing the envelope-tightening algorithm produces a better structure for the network.

VIII. Conclusion

A divide and conquer approach for the modeling of nonlinear systems based on GPs was introduced in this paper. A novel algorithm for structure identification for the local model network was proposed. The estimate of the predicted variance provided by a global GP model was used for structure identification and clustering the training data. A local LGP model was introduced, which was then identified for each cluster of data. The uncertainty of each local LGP model was utilized to help optimize the widths of the validity functions of the local linear GP model network.

An important analysis of realizations of the local LGPN was provided, in which it was shown that although the network in which the outputs of the local models are blended can provide a correct prediction, the variance of the network, however, cannot be used as a measure of model uncertainty. On the other hand, the realization utilizing parameter blending provided a prediction along with a variance, which was seen as a valid measure of model uncertainty. It was shown that this realization is also transparent.

A number of simulated examples was used to illustrate the benefits of the proposed approach. A laboratory-scale process rig was used as a case study to help demonstrate the potential of the proposed techniques for the modeling of industrial plant from real process data. In this example, it was shown that with the help of a measure of locality provided by the local models, it

is possible to optimize the widths of the validity functions using only locally collected data.

Perhaps the most important contribution coming from this paper is the merging of existing, well-established, parametrical, multiple-model approaches, with the Bayesian GP prior methodology. This combination resulted in a novel modeling algorithm which benefits from both techniques. The resulting model has proved to be flexible, transparent, easy to use, and suitable for model-based control. However, a number of research issues which were identified in this paper must be pointed out. At present, no theoretical foundation is provided for the correct choice of the cluster radius and the threshold in the minimum variance search algorithm. The correct choice of these parameters depends on the curvature of the variance surface. Sound and theoretically well-defined recommendations for a correct choice of the cluster radius and the threshold will make the minimum variance search algorithm much more straightforward and easy to use.

As a building block of the local LGP model network, a local LGP submodel was proposed in this paper. A standard optimization approach is usually utilized to train the hyperparameters, which is usually sensitive to initial conditions. The method of random restarts proved to be an effective while time-consuming way of overcoming this problem. Assuming a well-conditioned problem, the vector of model parameters can also be found using ordinary least squares. This suggests that the hyperparameters of the local LGP can be found analytically or numerically from the parameter vector determined using standard parametric system identification techniques, and hence, will eliminate optimization from the local model identification procedure. Avoiding optimization will dramatically improve the stability of calculations and make the LGP model much more convenient to use.

The model-predictive control approach is a popular choice for nonlinear control. It demands a parallel model, in which the model output is delayed and fed back to the model input. In this case, the uncertainty of the model increases with each further prediction step—a process known as uncertainty propagation. Recently, a method for the propagation of uncertainties has been developed for GP models using either nonlinear or linear covariance functions [56]. In order to use the local LGPN for model-predictive control, an uncertainty propagation for the local LGP model network has yet to be developed.

It is worth mentioning that the computation load, which is a drawback of the GP model and local LGPN, will become less significant in the future with the continual improvements in CPU power.

ACKNOWLEDGMENT

The authors would like to thank J. Kocijan, C. O'Driscoll, and C. Berton for comments and useful suggestions.

REFERENCES

[1] K. S. Narendra and K. Parthasarathy, "Identification and control of dynamical systems using neural networks," *IEEE Trans. Neural Netw.*, vol. 1, no. 1, pp. 4–27, Mar. 1990.

[2] T. Takagi and M. Sugeno, "Fuzzy identification of systems and its applications to modeling and control," *IEEE Trans. Syst. Man Cybern.*, vol. SMC–15, no. 1, pp. 116–132, Jan.–Feb. 1985.

[3] T. A. Johansen and B. A. Foss, "Constructing NARMAX models using ARMAX models," *Int. J. Control*, vol. 58, no. 5, pp. 1125–1153, 1993.

[4] M. D. Brown, G. W. Irwin, and G. Lightbody, "Local model networks for nonlinear system identification," in *Proc. 11th IFAC Symp. System Identif. (Invited Session)*, Kitakyushu, Japan, Jul. 1997, pp. 709–714.

[5] R. Murray-Smith and T. A. Johansen, Eds., *Multiple Model Approaches to Modelling and Control*. London, U.K.: Taylor & Francis, 1997.

[6] R. Shorten, R. Murray-Smith, and R. Bjørgan, "On the interpretation of local models in blended multiple model structures," *Int. J. Control*, vol. 72, no. 7/8, pp. 620–628, 1999.

[7] A. Zaknich, "A practical sub-space adaptive filter," *Neural Netw.*, vol. 16, no. 5/6, pp. 833–839, 2003.

[8] T. M. Martinetz, S. G. Berkovich, and K. J. Schulten, "Neural-gas network for vector quantization and its application to time-series prediction," *IEEE Trans. Neural Netw.*, vol. 4, no. 4, pp. 558–569, Jul. 1993.

[9] G. A. Barreto, A. F. R. Araújo, and H. J. Ritter, "Self-organizing feature maps for modeling and control of robotic manipulators," *J. Intell. Robot. Syst.*, vol. 36, no. 4, pp. 407–450, Apr. 2003.

[10] J. Principe, L. Wang, and M. Motter, "Local dynamic modeling with self-organizing maps and applications to nonlinear system identification and control," *Proc. IEEE*, vol. 86, no. 11, pp. 2241–2258, Nov. 1998.

[11] K. S. Narendra, J. Balakrishnan, and M. K. Ciliz, "Adaptation and learning using multiple models, switching, and tuning," *IEEE Control Syst. Mag.*, vol. 15, no. 3, pp. 37–51, Jun. 1995.

[12] K. S. Narendra and J. Balakrishnan, "Adaptive control using multiple models," *IEEE Trans. Autom. Control*, vol. 42, no. 2, pp. 171–187, Feb. 1997.

[13] M. D. Brown, G. Lightbody, and G. W. Irwin, "Nonlinear internal model control using local model networks," *Inst. Electr. Eng. Proc. —Control Theory Appl.*, vol. 144, no. 6, pp. 505–514, Nov. 1997.

[14] K. J. Hunt, T. A. Johansen, J. Kalluhl, H. Fritz, and T. Göttsche, "Speed control design for an experimental vehicle using a generalized gain scheduling approach," *IEEE Trans. Control Syst. Technol.*, vol. 8, no. 3, pp. 381–395, May 2000.

[15] V. Cherkassky and Y. Ma, "Multiple model regression estimation," *IEEE Trans. Neural Netw.*, vol. 16, no. 4, pp. 785–798, Jul. 2005.

[16] J. Cho, J. C. Principe, D. Erdogmus, and M. A. Motter, "Modeling and inverse controller design for an unmanned aerial vehicle based on the self-organizing map," *IEEE Trans. Neural Netw.*, vol. 17, no. 2, pp. 445–460, Mar. 2006.

[17] F. Flentge, "Locally weighted interpolating growing neural gas," *IEEE Trans. Neural Netw.*, vol. 17, no. 6, pp. 1382–1393, Nov. 2006.

[18] D. J. Leith and W. E. Leithead, "Analytic framework for blended multiple model systems using linear local models," *Int. J. Control*, vol. 72, no. 7/8, pp. 605–619, 1999.

[19] R. Murray-Smith, T. A. Johansen, and R. Shorten, "On transient dynamics, off-equilibrium behaviour and identification in blended multiple model structures," in *Proc. Eur. Control Conf.*, Karlsruhe, Germany, 1999, pp. BA–14.

[20] M. N. Gibbs, "Bayesian Gaussian processes for regression and classification," Ph.D. dissertation, Dept. Physics, Cavendish Lab., Univ. Cambridge, Cambridge, U.K., 1997.

[21] D. J. C. MacKay, *Information Theory, Inference and Learning Algorithms*. Cambridge, U.K.: Cambridge Univ. Press, 2003.

[22] C. M. Bishop and M. E. Tipping, C. Boutilier and M. Goldszmidt, Eds., "Variational relevance vector machines," in *Proc. 16th Conf. Uncertainty Artif. Intell.*, 2000, pp. 46–53.

[23] A. Shilton, M. Palaniswami, D. Ralph, and A. C. Tsoi, "Incremental training of support vector machines," *IEEE Trans. Neural Netw.*, vol. 16, no. 1, pp. 114–131, Jan. 2005.

[24] A. O'Hagan, "Curve fitting and optimal design for prediction (with discussion)," *J. Roy. Statist. Soc. B*, vol. 40, no. 1, pp. 1–42, 1978.

[25] C. E. Rasmussen, "Evaluation of Gaussian processes and other methods for non-linear regression," Ph.D. dissertation, Dept. Comp. Sci., Univ. Toronto, Toronto, ON, Canada, 1996.

[26] C. K. I. Williams and C. E. Rasmussen, "Gaussian processes for regression," in *Advances in Neural Information Processing Systems 8*, M. E. H. Touretzky and M. C. Mozer, Eds. Cambridge, MA: MIT Press, 1996, pp. 514–520.

[27] D. J. C. MacKay, "Introduction to Monte Carlo methods," in *Learning in Graphical Models*, ser. NATO Science Series, M. I. Jordan, Ed. Norwell, MA: Kluwer, 1998, pp. 175–204.

[28] C. K. I. Williams and D. Barber, "Bayesian classification with Gaussian processes," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 20, no. 12, pp. 1342–1351, Dec. 1998.

[29] C. K. I. Williams, "Prediction with Gaussian processes: From linear regression to linear prediction and beyond," Aston Univ., Birmingham, U.K., Tech. Rep. NCRG/97/012, 1998.

[30] C. K. I. Williams, *Regression With Gaussian Processes*, ser. Mathematics of Neural Networks: Models, Algorithms and Applications. Norwell, MA: Kluwer, 1997.
[31] D. J. C. MacKay, "Introduction to Gaussian processes," in *Neural Networks and Machine Learning*, ser. NATO ASI Series, C. M. Bishop, Ed. Norwell, MA: Kluwer, 1998, pp. 133–166.
[32] M. Seeger, "Bayesian Gaussian process models: PAC-Bayesian generalization error bounds and sparse approximations," Ph.D. dissertation, Inst. Adaptive Neural Comput., Div. Inf., Univ. Edinburgh, Edinburgh, U.K., 2003.
[33] J. Kocijan, A. Girard, B. Banko, and R. Murray-Smith, "Dynamic systems identification with Gaussian processes," *Math. Comput. Model. Dynam. Syst.*, vol. 11, no. 4, pp. 411–424, Dec. 2005.
[34] R. Murray-Smith and A. Girard, "Gaussian process priors with ARMA noise models," in *Proc. Irish Signals Syst. Conf.*, Maynooth, Ireland, Jun. 2001, pp. 147–152.
[35] A. Lundgren and J. Sjöberg, "Gaussian processes framework for validation of linear and nonlinear models," in *Proc. 13th IFAC Symp. System Identif. (SYSID)*, Rotterdam, The Netherlands, Aug. 2003, pp. 67–72.
[36] R. Murray-Smith and D. Sbarbaro, "Nonlinear adaptive control using nonparametric Gaussian process models," in *Proc. 15th Int. Federation Autom. Control Triennial World Congr.*, Barcelona, Spain, Jul. 2002, pp. 934–939.
[37] J. Kocijan, R. Murray-Smith, C. E. Rasmussen, and B. Likar, "Predictive control with Gaussian process models," in *Proc. IEEE Region 8 Eurocon 2003: Computer as a Tool*, Ljubljana, Slovenia, Sep. 2003, pp. 352–356.
[38] G. Gregorčič and G. Lightbody, "Internal model control based on a Gaussian process prior model," in *Proc. Amer. Control Conf.*, Denver, CO, Jun. 2003, pp. 4981–4986.
[39] G. Gregorčič and G. Lightbody, "An affine local Gaussian process model network (Invited paper)," in *Proc. IEE Int. Conf. Syst. Eng.*, Coventry, U.K., Sep. 2003, pp. 206–210.
[40] G. Gregorčič, "Data-based modelling of nonlinear systems for control," Ph.D. dissertation, Dept. Electr. Eng., Univ. College Cork, Cork, U.K., 2004.
[41] G. Gregorčič and G. Lightbody, A. E. Ruano, Ed., "Gaussian process approaches to nonlinear modelling for control," in *Intelligent Control Systems Using Computational Intelligence Techniques*, London, U.K., 2005, pp. 177–217, 6.
[42] L. Ljung, *System Identification: Theory for the User*, 2nd ed. Upper Saddle River, NJ: Prentice-Hall, 1999.
[43] W. E. Larimore, "Optimal reduced rank modeling, prediction, monitoring, and control using canonical variate analysis," in *Proc. IFAC Int. Symp. Adv.Control Chem. Processes*, Banff, Canada, 1997, pp. 61–66.
[44] D. J. Leith, W. E. Leithead, E. Solak, and R. Murray-Smith, "Divide & conquer identification using Gaussian process priors," in *Proc. 41st IEEE Conf. Decision Control*, Las Vegas, NV, Dec. 2002, pp. 624–629.
[45] J. Stark, D. S. Broomhead, M. E. Davies, and J. Huke, "Delay embeddings of forced systems: II stochastic forcing," *J. Nonlinear Sci.*, vol. 13, no. 6, pp. 519–577, Dec. 2003.
[46] T. J. Dodd and C. J. Harris, "Identification of nonlinear time series via kernels," *Int. J. Syst. Sci.*, vol. 33, no. 9, pp. 737–750, 2002.
[47] J. A. Leonard, M. A. Kramer, and L. H. Ungar, "Using radial basis functions to approximate a function and its error bounds," *IEEE Trans. Neural Netw.*, vol. 3, no. 4, pp. 624–627, Jul. 1992.
[48] G. Papadopoulos, P. J. Edwards, and A. F. Murray, "Confidence estimation methods for neural networks: A practical comparison," *IEEE Trans. Neural Netw.*, vol. 12, no. 6, pp. 1278–1287, Nov. 2001.
[49] Y. Reich and S. V. Barai, "Evaluating machine learning models for engineering problems," *Artif. Intell. Eng.*, vol. 13, no. 3, pp. 257–272, 1999.
[50] S. Faul, G. Gregorčič, G. Boylan, W. Marnane, G. Lightbody, and S. Connolly, "Gaussian process modelling as an indicator of neonatal seizure," in *Proc. 3rd IASTED Int. Conf. Signal Process. Pattern Recognit. Appl. (SPPRA)*, Innsbruck, Austria, Feb. 2006, pp. 177–182.
[51] T. A. Johansen and B. A. Foss, "Identification of non—Linear system structure and parameters using regime decomposition," *Automatica*, vol. 31, pp. 321–326, 1995.
[52] R. Murray-Smith, "A local model network approach to nonlinear modelling," Ph.D. dissertation, Dept. Comp. Sci., Univ. Strathclyde, Strathclyde, U.K., 1994.
[53] M. I. Jordan and R. A. Jacobs, "Hierarchical mixtures of experts and the em algorithm," *Neural Comput.*, vol. 6, pp. 181–214, 1994.
[54] L. J. Cao, S. S. Keerthi, C. J. Ong, J. Q. Zhang, U. Periyathamby, X. J. Fu, and H. P. Lee, "Parallel sequential minimal optimization for the training of support vector machines," *IEEE Trans. Neural Netw.*, vol. 17, no. 4, pp. 1039–1049, Jul. 2006.
[55] J. Shi, R. Murray-Smith, and D. M. Titterington, "Bayesian regression and classification using mixtures of Gaussian processes," *Int. J. Adapt. Control Signal Process.*, vol. 17, no. 2, pp. 149–161, 2003.
[56] A. Girard, "Approximate methods for propagation of uncertainty with Gaussian process models," Ph.D. dissertation, Comp. Sci. Dept., Univ. Glasgow, Glasgow, U.K., 2004.
[57] S. Waterhouse, D. MacKay, and T. Robinson, "Bayesian methods for mixtures of experts," in *Advances in Neural Information Processing Systems*, D. S. Touretzky, M. C. Mozer, and M. E. Hasselmo, Eds. Cambridge, MA: MIT Press, 1996, vol. 8, pp. 351–357.

Gregor Gregorčič received the degree in electrical engineering and computer science from the University of Maribor, Slovenia, in 1998 and the Ph.D. degree in electrical engineering from the University College Cork, Cork, Ireland, in 2004.

Currently, he is with the Control and Automation Development Department, AVL (Anstalt für Verbrennungskraftmaschinen List), Graz, Austria. His research area covers adaptive and model-based predictive control systems, system identification as well as applications of Takagi–Sugeno local model networks and neuro/fuzzy systems for nonlinear modeling and control. Recently, his research focused on Gaussian processes as a nonparametric approach to modeling and control of nonlinear systems. His current research is concentrated on development of advanced control strategies applied to complex automation systems for the automotive industry.

Gordon Lightbody received the M.Eng. (with distinction) and Ph.D. degrees in electrical and electronic engineering from the Queen's University, Belfast, Ireland, in 1989 and 1993, respectively.

After completing a one-year Postdoctoral position funded by Du Pont, he was appointed by Queen's University as a Lecturer in Modern Control Systems. At this stage, his research focused primarily on the application of intelligent control, modeling, and fault detection/diagnosis to the chemical process industry. In 1997, he was appointed to a Lectureship in Control Engineering at the University College Cork, Cork, Ireland. His current research interests include nonparametric modeling, local model networks for process modeling and control, model-based predictive control, fuzzy/neural systems, and nonlinear control. His work has focused on key application areas, which include, biomedical applications, wind power, power system control and harmonic analysis, chemical process control, and controlled bioremediation. He has published over 83 papers in these areas.

Dr. Lightbody is an Associate Editor of *Control Engineering Practice*.