Jožef Stefan Institute, Ljubljana, Slovenia

Gaussian Process Model Dynamic System Identification Toolbox for Matlab Version 1.02

Kristjan Ažman
Juš Kocijan

17. oktober 2009

1 Introduction

Our research activities included the use of Gaussian processes (GP) models to describe the (mainly nonlinear) dynamic systems. Although most of the code presented in this toolbox was already written for personal use of various authors, there were problems with its transparency and consistency, which presented some problems at the start of the work on the dynamic systems GP modelling. Beside simulated examples GP models proved useful also in practical applications. To promote their further use we decided to contribute at least the basics for the dynamical systems GP modelling toolbox.

The idea of this toolbox is to facilitate dynamic systems identification with GP models. The presented toolbox is not yet in the form we would like, but we still think it might be useful as a springboard for any future work.

While there are numerous methods for the identification of linear dynamic systems from measured data, the nonlinear systems identification requires more sophisticated approaches. The most common choices include artificial neural networks, fuzzy models and others. Gaussian process (GP) models present a new, emerging, complementary method for nonlinear system identification.

The GP model is a probabilistic, non-parametric black-box model. It differs from most of the other black-box identification approaches as it does not try to approximate the modelled system by fitting the parameters of the selected basis functions but rather searches for the relationship among measured data. Gaussian process models are closely related to approaches such as Support Vector Machines and specially Relevance Vector Machines [3].

The output of the Gaussian process model is a normal distribution, expressed in terms of mean and variance. The mean value represents the most likely output and the variance can be interpreted as the measure of its confidence. The obtained variance, which depends on the amount and quality of available identification data, is important information distinguishing the GP models from other methods. The GP model structure determination is facilitated as only the covariance function and the regressors of the model need to be selected. Another potentially useful attribute of the GP model is the possibility to include various kinds of prior knowledge into the model, see *e.g.* [46] for the incorporation of local models and the static characteristic. Also the number of model parameters, which need to be optimised is smaller than in other black-box identification approaches. The disadvantage of the method is the potential computational burden for optimization that increases with amount of data and number of regressors.

The GP model was first used for solving a regression problem in the late seventies, but it gained popularity within the machine learning community in the late nineties of the twentieth century. Results of a possible implementation of the GP model for the identification of dynamic systems were presented only recently, *e.g.* [11, 54]. The investigation of the model with uncertain inputs, which enables the propagation of uncertainty through the model, is given in [20, 33, 39] and illustrated in [27, 47] and many others.

Many of dynamic systems are often considered as complex, however simplified input/output behaviour representations are sufficient for certain purposes, *e.g.* feedback control design, prediction models for supervisory control, *etc.* In the paper it is explained how the advantages of Gaussian process models can be used in identification and validation of such models.

The purpose of the first part of manual is twofold. First, to present the procedure of dynamic system identification using the model based on Gaussian processes taken from [83]. Second, a comprehensive bibliography of published works on Gaussian processes application for modelling of dynamic systems with a short survey is given. The second part of manual serves as a reference guide.

2 Modelling of dynamic systems with Gaussian processes

2.1 Modelling with the GP model

Here, modelling with the GP model is presented only in brief, for a more detailed explanation see *e.g.* [79].

A Gaussian process is a Gaussian random function, fully described by its mean and variance. Gaussian processes can be viewed as a collection of random variables $f(\mathbf{x}_i)$ with joint multivariate Gaussian distribution: $f(\mathbf{x}_1), \dots, f(\mathbf{x}_n) \sim \mathcal{N}(0, \mathbf{K})$. Elements K_{ij} of the covariance matrix $\mathbf{K}$ are covariances between values of the function $f(\mathbf{x}_i)$ and $f(\mathbf{x}_j)$ and are functions of corresponding arguments $\mathbf{x}_i$ and $\mathbf{x}_j$: $K_{ij} = C(\mathbf{x}_i, \mathbf{x}_j)$. Any function $C(\mathbf{x}_i, \mathbf{x}_j)$ can be a covariance function, providing it generates a nonnegative definitive covariance matrix $\mathbf{K}$.

Certain assumptions about the process are made implicitly with the covariance function selection. The *stationarity of the process* results in the value of covariance function $C(\mathbf{x}_i, \mathbf{x}_j)$ between inputs $\mathbf{x}_i$ and $\mathbf{x}_j$ depending only on their distance and being invariant to their translation in the input space, see *e.g.* [79]. *Smoothness of the output* reflects in outputs $f(\mathbf{x}_i)$ and $f(\mathbf{x}_j)$ having higher covariance when inputs $\mathbf{x}_i$ and $\mathbf{x}_j$ are closer together. The common choice [79] for the covariance function, representing these assumptions, is the Gaussian covariance function:

$$\begin{aligned} C(\mathbf{x}_i, \mathbf{x}_j) &= \mathrm{cov}[f(\mathbf{x}_i), f(\mathbf{x}_j)] \\ &= v \exp\left[-\frac{1}{2}\sum_{d=1}^{D} w_d (x_i^d - x_j^d)^2\right] + \delta_{ij} v_0 \end{aligned} \tag{1}$$

where D is the length of vector $\mathbf{x}$ and $\mathbf{\Theta} = [w_1, \ldots, w_D, v, v_0]^T$ is a vector of parameters called hyperparameters.[1] The first term in (1) corresponds to functional dependance under presumed stationarity, while the second term corresponds to noise. Hyperparameter v controls the magnitude of the covariance and hyperparameters w_i represent the relative importance of each component x^d of vector $\mathbf{x}$. The part $\delta_{ij} v_0$ represents the covariance between outputs due to white noise, where δ_{ij} is the Kronecker operator and v_0 is the white noise variance. When assuming different kinds of noise the covariance function should be changed appropriately, *e.g.* [8]. With the use of covariance function (1) the total number of the GP model parameters is $D+2$ for the size D input, where for example the number of comparable artificial neural networks parameters would be considerably larger.

The GP model fits nicely into the Bayesian modelling framework. The idea behind GP modelling is to place the prior directly over the space of functions instead of parameterizing the unknown function $f(\mathbf{x})$ [79]. The simplest type of such a prior is Gaussian. Consider the system

$$y(k) = f(\mathbf{x}(k)) + \epsilon(k) \tag{2}$$

with white Gaussian noise $\epsilon(k) \sim \mathcal{N}(0, v_0)$ with variance v_0 and the vector of regressors $\mathbf{x}(k)$ from operating space $\mathcal{R}^D$. We put the GP prior with covariance function (1) with unknown hyperparameters on the space of functions $f(.)$.

Within this framework we have $y_1, \ldots, y_N \sim \mathcal{N}(0, \mathbf{K})$ with $\mathbf{K} = \mathbf{\Sigma} + v_0 \mathbf{I}$, where $\mathbf{I}$ is $N \times N$ identity matrix. Based on a set of N training data pairs $\{\mathbf{x}_i, y_i\}_{i=1}^N$ we wish to find the predictive distribution of y_{N+1} corresponding to a new given input $\mathbf{x}_{N+1}$. For the

[1]The parameters of a Gaussian process are called hyperparameters due to their close relationship to the hyperparameters of a neural network [79].

collection of random variables $(y_1, \ldots, y_N, y_{N+1})$ we can write:

$$\begin{pmatrix} \mathbf{y} \\ y_{N+1} \end{pmatrix} \sim \mathcal{N}(0, \mathbf{K}_{N+1}) \tag{3}$$

with covariance matrix

$$\mathbf{K}_{N+1} = \begin{bmatrix} \begin{bmatrix} \\ \mathbf{K} \\ \\ \end{bmatrix} & \begin{bmatrix} \mathbf{k}(\mathbf{x}_{N+1}) \\ \\ \end{bmatrix} \\ \\ \begin{bmatrix} \mathbf{k}(\mathbf{x}_{N+1})^T \end{bmatrix} & \begin{bmatrix} k(\mathbf{x}_{N+1}) \end{bmatrix} \end{bmatrix} \tag{4}$$

where $\mathbf{y} = [y_1, \ldots, y_N]^T$ is an $N \times 1$ vector of training targets, $\mathbf{k}(\mathbf{x}_{N+1}) = [C(\mathbf{x}_1, \mathbf{x}_{N+1}), \ldots, C(\mathbf{x}_N, \mathbf{x}_{N+1})]^T$ is the $N \times 1$ vector of covariances between training inputs and the test input and $k(\mathbf{x}_{N+1}) = C(\mathbf{x}_{N+1}, \mathbf{x}_{N+1})$ is the autocovariance of the test input. We can divide this joint probability into a marginal and a conditional part. The marginal term gives us the likelihood of the training data: $\mathbf{y}|\mathbf{X} \sim \mathcal{N}(0, \mathbf{K})$, where $\mathbf{X}$ is the $N \times D$ matrix of training inputs.

We need to estimate the unknown hyperparameters $\mathbf{\Theta} = [w_1, \ldots, w_D,\ v,\ v_0]^T$ of the covariance function (1). This is usually done via maximization of the log-likelihood

$$\begin{aligned} \mathcal{L}(\mathbf{\Theta}) &= \log(p(\mathbf{y}|\mathbf{X})) = \\ &= -\frac{1}{2}\log(|\ \mathbf{K}\ |) - \frac{1}{2}\mathbf{y}^T\mathbf{K}^{-1}\mathbf{y} - \frac{N}{2}\log(2\pi) \end{aligned} \tag{5}$$

with the vector of hyperparameters $\mathbf{\Theta}$ and $N \times N$ training covariance matrix $\mathbf{K}$. The optimization requires the computation of the derivative of $\mathcal{L}$ with respect to each of the parameters:

$$\frac{\partial \mathcal{L}(\mathbf{\Theta})}{\partial \Theta_i} = -\frac{1}{2}\text{trace}\left(\mathbf{K}^{-1}\frac{\partial \mathbf{K}}{\partial \Theta_i}\right) + \frac{1}{2}\mathbf{y}^T\mathbf{K}^{-1}\frac{\partial \mathbf{K}}{\partial \Theta_i}\mathbf{K}^{-1}\mathbf{y} \tag{6}$$

Here, it involves the computation of the inverse of the $N \times N$ covariance matrix $\mathbf{K}$ at every iteration, which can be computationally demanding for large N. The reader is referred to *e.g.* [79] for alternative methods of parameter optimisation.

Given that the hyperparameters are known, we can obtain a prediction of the GP model at the input $\mathbf{x}_{N+1}$. The conditional part of (3) provides the predictive distribution of y_{N+1}:

$$p(y_{N+1}|\mathbf{y}, \mathbf{X}, \mathbf{x}_{N+1}) = \frac{p(\mathbf{y}, y_{N+1})}{p(\mathbf{y}|\mathbf{X})} \tag{7}$$

It can be shown [79] that this distribution is Gaussian with mean and variance:

$$\mu(\mathbf{x}_{N+1}) = \mathbf{k}(\mathbf{x}_{N+1})^T \mathbf{K}^{-1} \mathbf{y} \tag{8}$$

$$\sigma^2(\mathbf{x}_{N+1}) = k(\mathbf{x}_{N+1}) - \mathbf{k}(\mathbf{x}_{N+1})^T \mathbf{K}^{-1} \mathbf{k}(\mathbf{x}_{N+1}) + v_0. \tag{9}$$

Vector $\mathbf{k}(\mathbf{x}_{N+1})^T\ \mathbf{K}^{-1}$ in (8) can be interpreted as a vector of smoothing terms which weights training outputs $\mathbf{y}$ to make a prediction at the test point $\mathbf{x}_{N+1}$. If the new input is far away from the data points, the term $\mathbf{k}(\mathbf{x}_{N+1})^T\ \mathbf{K}^{-1}\ \mathbf{k}(\mathbf{x}_{N+1})$ in (9) will be small, so that the predicted variance $\sigma^2(\mathbf{x}_{N+1})$ will be large. Regions of the input space, where there are few data or are corrupted with noise, are in this way indicated through higher variance.

2.2 Dynamic system identification

The presented GP model was originally used for modelling static nonlinearities, but it can be extended to model dynamic systems as well [39, 54, 3]. Our task is to model the dynamic system (2), where

$$\mathbf{x} = [y(k-1), \ldots, y(k-L),\ u(k-1), \ldots, u(k-L)] \tag{10}$$

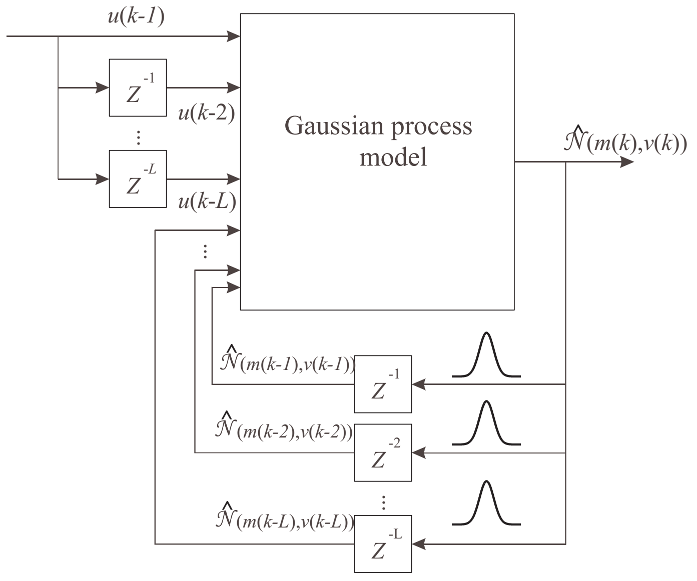
is the vector of regressors that determines nonlinear ARX model structure and be able to make multi-step ahead model prediction.

One way to do multi-step ahead prediction is to make iterative one-step ahead predictions up to desired step whilst feeding back the predicted output. Two general approaches to iterated one-step ahead prediction are possible using the GP model. In the first only the mean values of the predicted output are fed back to the input. In this, so called "naive" approach, the input vector $\mathbf{x}$ into the GP model at time step k is:

$$\mathbf{x} = [\hat{y}(k-1), \ldots, \hat{y}(k-L),\ u(k-1), \ldots, u(k-L)] \tag{11}$$

Although this approach is approximate, as the variance of the lagged output estimates on the right-hand side of Equation (11) is neglected, it has been used when modelling dynamic systems with neural networks or fuzzy models. This way of generating multiple-step-ahead predictions is commonly referred to as 'output error' in the identification literature. However, it has been shown to lead to unrealistically small variances for the multiple-step-ahead predictions when modelling with GP models and with the predictive distribution calculated with Equations (8) and (9) [10].

In [10, 20, 33, 39, 54] the iterative, multiple-step-ahead prediction is done by feeding back the mean of the predictive distribution as well as the variance of the predictive distribution at each time-step, thus taking the uncertainty attached to each intermediate prediction into account. In this way, each input for which we wish to predict becomes a normally distributed random variable. However, this is still an approximation, as is explained in more detail in [39]. The illustration of such a dynamical model simulation is given in Figure 1.



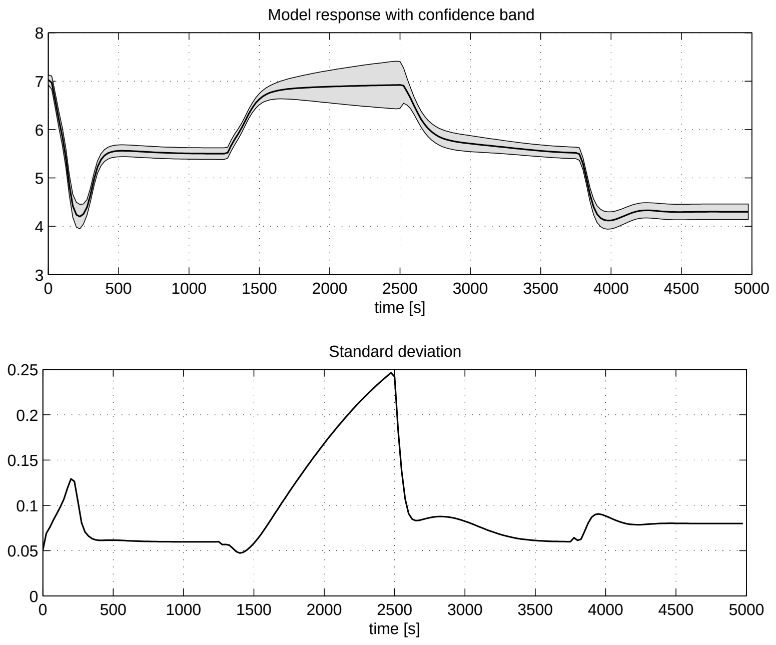
Slika 1: Illustration of simulation principle for a Gaussian process model of dynamic system [53]

A demonstration of a Gaussian process model response is given in Figure 2.

3 Gaussian process model identification methodology

In this section the framework for dynamic system identification with GP models taken from [83] is given. The identification framework consists of roughly six stages:

- defining the purpose of the model,
- model selection,
- design of the experiment,



Slika 2: Simulated response of a dynamic system modelled by Gaussian process model

- realisation of the experiment and data processing,
- training of the model and
- model validation.

The model identification is an iterative process. Returning to some previous procedure step is possible at any step in the identification process and is usually necessary.

3.1 The model purpose and model selection

The decision for the use of a specific model derives from the model purpose and from the limitations met at the identification process. In this paper selection of the GP model is presumed. This approach can be beneficial when the information about the system exists in the form of input/output data, when data are corrupted, *e.g.* by noise and measurement errors, when a measure of confidence in model prediction is required and when there is a relatively small amount of data in respect to the selected number of regressors.

After the model is selected, its structure must be determined next. In the case of the GP model this means selecting the covariance function and the model regressors. The choice of the covariance function reflects the relationship between data and is based on prior knowledge of the process. The standard choice for smooth and stationary processes is function (1). Prior knowledge about other attributes, *e.g.* periodicity, non-stationarity, can be expressed through a different choice of the covariance function [79].

The second part of structure determination is the choice of proper regressors. In the case of a dynamic system model this also means selecting the model order, which is the area of intensive research, as it is common to all nonlinear identification methods.

The most frequent approach for regressor selection is the so called *validation based regressor selection*, where the search for the optimal vector of regressors is initiated from some basic set of regressors. After the model optimisation and cross-validation, the regressors are added to or taken from the model. Prospering models according to selected performance are kept while dissatisfying models are rejected. In the case of normalised inputs the influence of each regressor can be observed through the value of the associated hyperparameter. If the associated regressor is not relevant enough it can be removed from the perspective model.

3.2 Obtaining data – design of the experiment, experiment and data processing

Data describing the unknown system is very important in any black-box identification. For a good description of the process the influential variables and proper sample time must be chosen.

The design of the experiment and the experiment itself are, as is always the case in systems modelling, very important parts of the identification procedure. The quality of the model depends on the system information contained in the measurement data, regardless of the identification method. Nevertheless, the design of the experiment is not the focus of this paper.

As already mentioned the Gaussian process modelling approach relies on the relation among input/output data and not on approximation with basis functions. Consequently, this means that the distribution of identification data within the process operating region is crucial for the quality of the model. Model predictions can be informative only if the inputs to the model lie in the regions, where training data is available. The GP model is good for interpolation, but not for extrapolation, which is indicated by large variances of model predictions.

Consequently, the data for model training should be chosen reasonably, which can be obstructed by the nature of the process (*e.g.* limitations in the experiment design in industrial processes, physical limitations of the system). The preprocessing of measured data, such as normalisation to cancel the influence of different measuring scales, can be pursued.

3.3 Model training

In the GP model approach training means optimization of hyperparameters $\mathbf{\Theta}$ from (1). Each hyperparameter w_d expresses the relative importance of the associated regressor, similar to the automatic relevant detection (ARD) method [79], where a higher value of w_d expresses higher importance of the regressor. Hyperparameter v expresses the overall scale of correlations and hyperparamter v_0 accounts for the influence of noise. Several possibilities of hyperparameter determination exist. A very rare possibility is that hyperparameters are known in advance as prior knowledge. Almost always, however, they must be determined from the training data, where different approaches are possible, *e.g.* [39]. Mostly the likelihood maximization (ML) approach is used as it gives good results despite its simplification, where any optimization method could be used to achieve ML [39].

3.4 Model validation

Validation concerns the level of agreement between the mathematical model and the system under investigation [2] and it is many times underemphasised despite its importance. Several features can represent the quality of the model. Their overview can be found *e.g.* in [2, 1]. The most important are model plausibility, model falseness and model purposiveness, explained as follows.

Model *plausibility* expresses the model's conformity with the prior process knowledge by answering two questions: whether the model "looks logical" and whether the model "behaves logical". The first question addresses the model structure, which in the case of GP models means mainly the plausibility of the hyperparameters. The second one is concerned with the responses of the model output to typical events on the input, which can be validated with visual inspection of the responses as is the case with other black-box models.

Model *falseness* reflects the agreement between the process and the model output or the process input and the output of the inverse model. The comparison can be done in two ways, both applicable to GP models: qualitatively, *i.e.* by visual inspection of differences in responses between the model and the process, or quantitatively, *i.e.* through evaluation of performance measures. Beside commonly used performance measures such as *e.g.* mean squared error MSE and mean relative square error (MRSE, which compares only the mean prediction of the model to the output of the process:

$$\mathrm{MSE} = \frac{1}{N}\sum_{i=1}^{N} e_i^2 \tag{12}$$

$$\mathrm{MRSE} = \sqrt{\frac{\sum_{i=1}^{N} e_i^2}{\sum_{i=1}^{N} y_i^2}} \tag{13}$$

where y_i and $e_i = \hat{y}_i - y_i$ are the system's output and prediction error in i-th step of simulation, the performance measures such as log predictive density error (LD, [39, 54]) can be used for evaluating GP models, taking into account not only mean prediction but the entire predicted distribution:

$$\mathrm{LD} = \frac{1}{2}\,log(2\pi) + \frac{1}{2N}\sum_{i=1}^{N}\left(log(\sigma_i^2) + \frac{e_i^2}{\sigma_i^2}\right) \tag{14}$$

where σ_i^2 is the prediction variance in i-th step of simulation. Performance measure LD weights the prediction error e_i more heavily when it is accompanied with smaller predicted variance σ_i^2, thus penalising overconfident predictions more than acknowledged bad predictions, indicated by higher variance. Another possible performance measure, applicable in the training procedure, is the negative log-likelihood of the training data (LL, [39]):

$$\mathrm{LL} = \frac{1}{2}\log\mid\mathbf{K}\mid + \frac{1}{2}\mathbf{y}^T\mathbf{K}^{-1}\mathbf{y} + \frac{N}{2}\log(2\pi), \tag{15}$$

where $\mathbf{K}$ is the covariance matrix, $\mathbf{y}$ is the vector of targets and N is the number of training points. LL is the measure inherent to the hyperparameter optimisation process, see (5), and gives the likelihood that the training data is generated by given, *i.e.* trained, model. The smaller the MRSE, LD and LL are, the better the model is.

Variance of the model predictions on a validation signal can be a validation measure itself, as it indicates whether the model operates in the region, where identification data were available. Nevertheless, it should be used carefully and in combinations with other validation tools, as predictions with small variance are not necessary good.

Model *purposiveness* or usefulness tells whether or not the model satisfies its purpose, which means the model is validated when the problem that motivated the modelling exercise can be solved using the obtained model. Here, again, the prediction variance can be used, *e.g.* when the prediction confidence is too low, the model can be labelled as not purposive.

4 Survey of publications on Gaussian process models of dynamic systems

The GP model was first used for solving a regression problem in the late 1970s, but it only gained popularity within the machine-learning community in the late 1990s. Furthermore, the results of a possible implementation of the GP model for the identification of dynamic systems were presented as recent as the last decade.

After what can be described as initial publications in year 1999 [4], year 2000 [5, 6] and year 2001 [7, 8], numbers of publications start to grow. Numerus publications on conferences and as internal, but publicly available publications occurred in years 2002 [9]-[19], 2003 [20]-[37] and 2004 [38]-[45]. After the first journal publication in year 2003 [24], publications in years 2005 [46]-[68], 2006 [69]-[81] and 2007 [82]-[97] contain more versatile publications including journal papers, book chapters and books mentioning use of GP models for the modelling of dynamic systems. In spite of efforts to be very thorough it is possible that the list of publications until year 2007 is not complete, but it certainly represents the majority of publications on Gaussian process models of dynamic systems.

These publications have explored use of Gaussian process models for various applications:

- dynamic systems modelling, e.g., [10],[11],[27],[65]
- time-series prediction, e.g., [7],[73],
- dynamic systems control, e.g., [12],[13],[18],[55],
- fault detection, e.g., [74],
- smoothing, e.g., [82],
- etc.

The utility to provide the information about the model prediction confidence made Gaussian process models attractive for modelling case studies in various domains like: chemical engineering [91] and process control [93], biomedical engineering [84], biological systems [83], environmental systems [73], power systems [43] and engineering [60], motion recognition [65], etc., to list just a few. It is worth noticing that the utility of Gaussian process modelling could be interesting also for use in other domains and applications therein.

5 Concluding remarks

In the first part it is explained how the Gaussian process model is used for dynamic systems identification with emphasis on some of its properties: model predictions containing the measure of confidence, low number of parameters and facilitated structure determination.

The prediction variance is one of the main differences between the GP model and other black box models. It can be effectively used in the usefulness validation, where the lack of confidence in the model prediction can serve as the grounds to reject the model as not useful. The prediction variance can also be used in falseness validation, whether via specific performance measures such as log-predictive density error, or through observation of confidence limits around the predicted output. Despite its usefulness in model validation, it should be accompanied with standard validation tools, as the small variance does not necessarily mean that the model is of good quality.

In the validation based regressor selection procedure the log-predictive density error and the log-likelihood of the training data can be useful in selecting model regressors. In the case of normalised inputs, the model hyperparameters indicate the influence of corresponding regressors and can be used as a tool for removal of non-influental regressors at the regressor selection stage of the model selection.

Small amounts of data relative to the number of selected regressors, data corrupted with noise and measurement errors and the need for the measure of model prediction confidence could be the reasons to select identification with the GP model. If there is not enough data or it is heavily corrupted with noise, even the GP model cannot perform well, but in that case the inadequacy of the model and the identification data is indicated through higher variance of the predictions.

The short survey and bibliography on Gaussian process models for dynamic systems shows that the interest in this modelling approach and its applications is growing. Published results have shown the GP model's potential for the identification of nonlinear dynamic systems and where the advantages of the GP model could be effectively used, e.g., for control design, diagnostic system design etc.

Literatura

[1] N. Hvala, S. Strmčnik, D. Šel, S. Milanič and B. Banko. Influence of model validation on proper selection of process models — an industrial case study. *Computers and Chemical Engineering*, 2005, **29**, 1507–1522.

[2] D.J. Murray-Smith. Methods for the external validation of continuous system simulation models: a review. *Mathematical and Computer Modelling of Dynamical Systems*, 1998, **4**, 5–31.

[3] J. Quiñonero-Candela and C.E. Rasmussen. Analysis of Some Methods for Reduced Rank Gaussian Process Regression. In: Murray-Smith, R. and Shorten R. (Eds.), *Switching and Learning in Feedback Systems*, Lecture Notes in Computer Science, Vol. 3355, 2005.

Published references on Gaussian process models of dynamic systems and their publications listed by publishing year

1999

[4] R. Murray-Smith, T. A. Johansen, R. Shorten. On transient dynamics, off-equilibrium behaviour and identification in blended multiple model structures. *In Proceedings of European Control Conference*, Paper BA-14, Karslruhe, 1999.

2000

[5] D. J. Leith, R. Murray-Smith, and W. E. Leithead. Nonlinear structure identification: A Gaussian process/velocity-based approach. *In Proceedings of the UKACC Control Conference*, Cambridge, 2000.

[6] W. E. Leithead, D. J. Leith, and R. Murray-Smith. A Gaussian Process prior/Velocity-based Framework for Nonlinear Modelling and Control. *In Irish Signals and Systems Conference*, Dublin, 2000.

2001

[7] V. Babovic and M. Keijzer. A Gaussian process model applied to prediction of the water levels in Venice lagoon. *In Proceedings of the XXIX Congress of International Association for Hydraulic Research*, 2001.

[8] R. Murray-Smith and A. Girard. Gaussian process priors with ARMA noise models. *In Proceedings of Irish Signals and Systems Conference*, Pages 147-152, Maynooth, 2001.

2002

[9] B. Banko and J. Kocijan. Uporaba Gaussovih procesov za identifikacijo nelinearnih sistemov. In B. Zajc, editor, *Zbornik enajste elektrotehniške in računalniške konference ERK*, Volume A, pages 323-326, Portorož, 2002. (in Slovene).

[10] A. Girard, C. E. Rasmussen, and R. Murray-Smith. Gaussian process priors with uncertain inputs: multiple-step-ahead prediction. Technical Report DCS TR-2002-119, University of Glasgow, Glasgow, 2002.

[11] G. Gregorčič and G. Lightbody. Gaussian processes for modelling of dynamic non-linear systems. *In Proceedings of Irish Signals and Systems Conference*, Cork, Pages 141-147, Cork, June 2002.

[12] G. Gregorčič and G. Lightbody. Gaussian processes for internal model control. In A. Rakar, editor, *Proceedings of 3rd International PhD Workshop on Advances in Supervision and Control Systems*, A Young Generation Viewpoint, Pages 39-46, Strunjan, 2002.

[13] J. Kocijan. Gaussian process model based predictive control. Technical Report DP-8710, Institut Jožef Stefan, Ljubljana, 2002.

[14] J. Kocijan, B. Likar, B. Banko, A. Girard, R. Murray-Smith, and C. E. Rasmussen. Identification of pH neutralization process with neural networks and Gaussian process model: MAC project. Technical Report DP-8575, Institut Jožef Stefan, Ljubljana, 2002.

[15] D. Leith. Determining nonlinear structure in time series data. *In Proceedings of Workshop on Modern Methods for Data Intensive Modelling*, Maynooth, 2002. NUI Maynooth.

[16] D. J. Leith, W. E. Leithead, E. Solak, and R. Murray-Smith. Divide and conquer identification using Gaussian processes. *In Proceedings of the 41st Conference on Decision and Control*, Pages 624-629, Las Vegas, AZ, 2002.

[17] D. J. Leith, W. E. Leithead, E. Solak, and R. Murray-Smith. Divide and conquer identification using Gaussian processes. *In Proceedings of Workshop on non-linear and non-Gaussian signal processing*, Peebles, UK, 2002.

[18] R. Murray-Smith and D. Sbarbaro. Nonlinear adaptive control using nonparametric Gaussian process prior models. *In Proceedings of IFAC 15th World Congress*, Barcelona, 2002.

[19] R. Murray-Smith, R. Shorten, and D. Leith. Nonparametric models of dynamic systems. In C. Cowans, editor, *Proceedings of IEE Workshop on Nonlinear and Non-Gaussian signal processing - N2SP*, Peebles, UK, 2002.

2003

[20] A. Girard, C. E. Rasmussen, J. Quinonero-Candela, and R. Murray-Smith. Bayesian regression and Gaussian process priors with uncertain inputs - application to multiple-step ahead time series forecasting. In S. Becker, S. Thrun, and K. Obermayer, editors, *Advances in Neural Information Processing Systems conference*, Volume 15, Pages 529-536. MIT Press, 2003.

[21] G. Gray, R. Murray-Smith, K. Thompson, and D. J. Murray-Smith. Tutorial example of Gaussian process prior modelling applied to twin-tank system. Technical Report DCS TR-2003-151, University of Glasgow, Glasgow, 2003.

[22] G. Gregorčič and G. Lightbody. Internal model control based on Gaussian process prior model. *In Proceedings of the 2003 American Control Conference (ACC)*, pages 4981-4986, Denver, CO, June 2003.

[23] G. Gregorčič and G. Lightbody. From multiple model networks to the Gaussian processes prior model. *In Proceedings of IFAC ICONS conference*, Pages 149-154, Faro, 2003.

[24] G. Gregorčič and G. Lightbody. An afine Gaussian process approach for nonlinear system identification. *Systems Science Journal*, Volume 29, Issue 2, Pages 47-63, 2003.

[25] J. Hansen. Using Gaussian processes as a modelling tool in control systems. Technical Report DCS TR-2003, University of Glasgow, Glasgow, 2003.

[26] J. Kocijan, B. Banko, B. Likar, A. Girard, R. Murray-Smith, and C. E. Rasmussen. A case based comparison of identification with neural networks and Gaussian process models. *In Proceedings of IFAC ICONS conference*, Volume 1, Pages 137-142, Faro, 2003.

[27] J. Kocijan, A. Girard, B. Banko, and R. Murray-Smith. Dynamic systems identification with Gaussian processes. In I. Troch and F. Breitenecker, editors, *Proceedings of 4th IMACS Symposium on Mathematical Modelling (MathMod)*, pages 776-784, Vienna, 2003.

[28] J. Kocijan, A. Girard, and D. J. Leith. Incorporating linear local models in Gaussian process model. Technical Report DP-8895, Institut Jožef Stefan, Ljubljana, December 2003.

[29] J. Kocijan, R. Murray-Smith, C. E. Rasmussen, and B. Likar. Predictive control with Gaussian process models. In B. Zajc and M. Tkalcic, editors, *The IEEE Region 8 EUROCON 2003: computer as a tool*, Volume A, Pages 352-356, Ljubljana, 2003.

[30] D. J. Leith and W. E. Leithead. Nonlinear structure identification with application to Wiener-Hammerstein systems. *In Proceedings of 13th IFAC Symposium on System Identification*, Rotterdam, 2003.

[31] W. E. Leithead, E. Solak, and D. J. Leith. Direct identification of nonlinear structure using Gaussian process prior models. *In Proceedings of European Control Conference*, Cambridge, 2003.

[32] R. Murray-Smith, D. Sbarbaro, C. E. Rasmussen, and A. Girard. Adaptive, cautious, predictive control with Gaussian process priors. *In Proceedings of 13th IFAC Symposium on System Identification*, Pages 1195-1200, Rotterdam, 2003.

[33] J. Quinonero-Candela and A. Girard. Prediction at uncertain input for Gaussian processes and relevance vector machines - Application to multiple-step ahead time-series forecasting. Technical Report IMM-2003-18, Technical University Denmark, Informatics and Mathematical Modelling, Kongens Lyngby, 2003.

[34] J. Quinonero-Candela, A. Girard, J. Larsen, and C. E. Rasmussen. Propagation of uncertainty in Bayesian kernel models - Application to multiple-step ahead forecasting. *In Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Volume 2, Pages 701-704, 2003.

[35] C. E. Rasmussen and M. Kuss. Gaussian processes in reinforcement learning. In S. Thrun, L. K. Saul, and B. Schoelkopf, editors, *Advances in Neural Information Processing Systems conference*, Volume 16, Pages 751-759. MIT Press, 2004.

[36] D. Sbarbaro and R. Murray-Smith. Self-tuning control of nonlinear systems using Gaussian process prior models. Technical Report DCS TR-2003-143, University of Glasgow, Glasgow, 2003.

[37] E. Solak, R. Murray-Smith, W. E. Leithead, D. J. Leith, and C. E. Rasmussen. Derivative observations in Gaussian process models of dynamic systems. In S. Becker, S. Thrun, and K. Obermayer, editors, *Advances in Neural Information Processing Systems conference*, Volume 15, Pages 529-536. MIT Press, 2003.

2004

[38] K. Ažman. Identifikacija dinamičnih sistemov z Gaussovimi procesi z vključenimi lokalnimi modeli. Master's thesis, Univerza v Ljubljani, Ljubljana, September 2004. (in Slovene).

[39] A. Girard. Approximate methods for propagation of uncertainty with Gaussian process models. PhD thesis, University of Glasgow, Glasgow, 2004.

[40] G. Gregorčič. Data-based modelling of nonlinear systems for control. PhD thesis, University College Cork, National University of Ireland, Cork, 2004.

[41] J. Kocijan and D. J. Leith. Derivative observations used in predictive control. *In Proceedings of Melecon 2004*, Volume 1, Pages 379-382, Dubrovnik, 12.-15. May 2004.

[42] J. Kocijan, R. Murray-Smith, C. E. Rasmussen, and A. Girard. Gaussian process model based predictive control. *In Proceedings of the 2004 American Control Conference (ACC)*, Pages 2214-2218, Boston, MA, 30. June-2. July 2004.

[43] D. J. Leith, M. Heidl and J. Ringwood. Gaussian process prior models for electrical load forecasting. *In 2004 International Conference on Probabilistic Methods Applied to Power Systems*, Pages 112-117. 2004.

[44] B. Likar. Prediktivno vodenje nelinearnih sistemov na osnovi Gaussovih procesov. Master's thesis, Univerza v Ljubljani, Ljubljana, September 2004. (in Slovene).

[45] D. Sbarbaro, R. Murray-Smith, and A. Valdes. Multivariable generalized minimum variance control based on artificial neural networks and Gaussian process models. *In International Symposium on Neural Networks*. Springer Verlag, 2004.

2005

[46] K. Ažman. Incorporating prior knowledge into Gaussian process model. *In Proceedings of 6th International PhD Workshop on Systems and Control* - A Young Generation Viewpoint, Volume A, Pages 253-256, Izola, 2005.

[47] K. Ažman and J. Kocijan. An example of Gaussian process model identification. In L. Budin and S. Ribarić, editors, *Proceedings of 28th International conference MIPRO*, CIS - Inteligent Systems, Pages 79-84, Opatija, maj 2005.

[48] K. Ažman and J. Kocijan. Identifikacija dinamičnega sistema s histerezo z modelom na osnovi Gaussovih procesov. In B. Zajc and A. Trost, editors, *Zbornik štirinajste elektrotehniške in računalniške konference ERK*, Volume A, Pages 253-256, Portorož, 2005. (in Slovene).

[49] K. Ažman and J. Kocijan. Comprising prior knowledge in dynamic Gaussian process models. *In Proceedings of the International Conference on Computer Systems and Technologies* - CompSysTech, Pages IIIB.2-1 – IIIB.2-6, Varna, 2005.

[50] B. Grašič. Napovedovanje povišanih koncentracij ozona z uporabo umetnih nevronskih mrež, Gaussovih procesov in mehke logike. Master's thesis, Univerza v Ljubljani, Ljubljana, 2005. (in Slovene).

[51] G. Gregorčič and G. Lightbody. Gaussian process approaches to nonlinear modelling and control. In A. Ruano, editor, *Intelligent control systems using computational intelligence techniques*, IEE Intelligent Control Series. IEE, 2005.

[52] J. Hansen, R. Murray-Smith, and T. A. Johansen. Nonparametric identification of linearizations and uncertainty using Gaussian process models - application to robust wheel slip control. *In Joint 44th IEEE conference on decision and control and European control conference CDC-ECC 2005*, Pages 7994-7999, Sevilla, 2005.

[53] J. Kocijan and A. Girard. Incorporating linear local models in Gaussian process model. *In Proceedings of IFAC 16th World Congress*, Praga, 2005.

[54] J. Kocijan, A. Girard, B. Banko, and R. Murray-Smith. Dynamic systems identification with Gaussian processes. *Mathematical and Computer Modelling of Dynamic Systems*, Volume 11, Issue 4, Pages 411-424, December 2005.

[55] J. Kocijan and R. Murray-Smith. Nonlinear predictive control with Gaussian process model. In *Switching and Learning in Feedback Systems*, volume 3355 of Lecture Notes in Computer Science, Pages 185-200. Springer, Heidelberg, 2005.

[56] W. E. Leithead. Identification of nonlinear dynamic systems by combining equilibrium and off-equilibrium information. *In Proceedings of International Conference on Industrial Electronics and Control Applications (ICIECA)*, Quito, 2005.

[57] W. E. Leithead, K. S. Neo, and D. J. Leith. Gaussian regression based on models with two stochastic processes. *In Proceedings of IFAC 16th World Congress*, Praga, 2005.

[58] W. E. Leithead, Y. Zhang, and D. J. Leith. Eficient hyperparameter estimation of Gaussian process regression based on quasi-Newton BFGS update and power series approximation. *In Proceedings of IFAC 16th World Congress*, Praga, 2005.

[59] W. E. Leithead, Y. Zhang, and D. J. Leith. Time-series Gaussian process regresion based on Toeplitz computation of O(N2) operations and O(N) level storage. *In Joint 44th IEEE conference on decision and control and European control conference CDC-ECC 2005*, Sevilla, 2005.

[60] W. E. Leithead, Y. Zhang, and K.S. Neo. Wind turbine rotor acceleration: Identification using Gaussian regression. *In Proceedings of International conference on informatics in control automation and robotics (ICINCO)*, Barcelona, 2005.

[61] R. Murray-Smith, B. A. Pearlmutter. Transformations of Gaussian Process priors. In *Deterministic and Statistical Methods in Machine Learning*, Volume 3536 of Lecture Notes in Artificial Intelligence, Pages 110-123. Springer, Heidelberg, 2005.

[62] R. Palm. Multi-step-ahead prediction with Gaussian processes and TS-fuzzy models. *In Proceedings of 14th IEEE International Conference on Fuzzy Systems*, Pages 945-950, 2005.

[63] D. Sbarbaro and R. Murray-Smith. Self-tuning control of nonlinear systems using Gaussian process prior models. In *Switching and Learning in Feedback Systems*, Volume 3355 of Lecture Notes in Computer Science, Pages 140-157. Springer, Heidelberg, 2005.

[64] J. Q. Shi, R. Murray-Smith, and D. M. Titterington. Hierarchical Gaussian process mixtures for regression. *Statistics and Computing*, Volume 15, Pages 31-41, 2005.

[65] J. M.Wang, D. J. Fleet, and A. Hertzmann. Gaussian process dynamical models. *In Advances in Neural Information Processing Systems*, Volume 18, Pages 1441-1448. MIT Press, 2005.

[66] Z.-H. Xiong, W.-Q. Zhang, Y. Zhao, H.-H. Shao. Thermal parameter soft sensor based on the mixture of Gaussian processes. *Zhongguo Dianji Gongcheng Xuebao (Proc. Chin. Soc. Electr. Eng.)*, Volume 25, Issue 7, Pages 30-33, 2005.

[67] Z.-H. Xiong, H.-B. Yang, Y.-F. Wu, H.-H. Shao. Sparse GP-based soft sensor applied to the power plant. *Zhongguo Dianji Gongcheng Xuebao (Proc. Chin. Soc. Electr. Eng.)*, Volume 25, Issue 8, Pages 130-133, 2005.

[68] Y. Zhang and W. E. Leithead. Exploiting Hessian matrix and trust region algorithm in hyperparameters estimation of Gaussian process. *Applied Mathematics and Computation*, Volume 171, Issue 2, Pages 1264 - 1281, 2005.

2006

[69] K. Ažman and J. Kocijan. Gaussian process model validation: biotechnological case studies. In I. Troch and F. Breitenecker, editors, *Proceedings of the 5th Vienna Symposium on Mathematical Modeling - MathMod*, Vienna, 2006.

[70] K. Ažman and J. Kocijan. Identifikacija dinamičnega sistema z znanim modelom šuma z modelom na osnovi Gaussovih procesov. In B. Zajc and A. Trost, editors, *Zbornik petnajste elektrotehniške in računalniške konference ERK*, Volume A, Pages 289-292, Portorož, 2006. (in Slovene).

[71] K. Ažman and J. Kocijan. An application of Gaussian process models for control design. *In UKACC International Control Conference*, Glasgow, 2006.

[72] P. Boyle. Gaussian processes for regression and optimisation. PhD thesis, Victoria University of Wellington, Wellington, New Zealand, 2006.

[73] B. Grašič, P. Mlakar, and M. Z. Božnar. Ozone prediction based on neural networks and Gaussian processes. *Nuovo Cimento della Societa Italiana di Fisica, Sect. C*, Volume 29, Issue 6, Pages 651-662, 2006.

[74] Dj. Juričić and J. Kocijan. Fault detection based on Gaussian process model. In I. Troch and F. Breitenecker, editors, *Proceedings of the 5th Vienna Symposium on Mathematical Modeling - MathMod*, Vienna, 2006.

[75] M. Kuss. Gaussian process models for robust regression, classification and reinforcement learning. PhD thesis, Technische Universitaet Darmstadt, Darmstadt, 2006.

[76] D. J. Leith, R. Murray-Smith, and W. E. Leithead. Inference of disjoint linear and nonlinear subdomains of a nonlinear mapping. *Automatica*, Volume 42, Issue 5, Pages 849-858, May 2006.

[77] K. Moon, V. Pavlović. Impact of dynamics on subspace embedding and tracking of sequences. *In Proceedings - 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, CVPR 2006 Volume 1, Pages 198-205, 2006.

[78] K. S. Neo, W. E. Leithead, and Y. Zhang. Multi frequency scale Gaussian regression for noisy time-series data. *In UKACC International Control Conference*, Glasgow, 2006.

[79] C.E. Rasmussen and C.K.I. Williams, *Gaussian Processes for machine learning.* The MIT Press, Cambridge, MA, 2006.

[80] K. Thompson and D. J. Murray-Smith. Implementation of Gaussian process models for nonlinear system identification. In I. Troch and F. Breitenecker, editors, *Proceedings of the 5th Vienna Symposium on Mathematical Modeling - MathMod*, Vienna, 2006.

[81] R. Urtasun, D. J. Fleet, P. Fua. 3D people tracking with Gaussian process dynamical models. *In Proceedings - 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, CVPR 2006 Volume 1, Pages 238-245, 2006.

2007

[82] K. Ažman. Identifikacija dinamičnih sistemov z Gaussovimi procesi. PhD thesis, Univerza v Ljubljani, Ljubljana, 2007. (in Slovene).

[83] K. Ažman and J. Kocijan. Application of Gaussian processes for black-box modelling of biosystems. *ISA Transactions*, Volume 46, Issue 4, Pages 443-457, 2007.

[84] S. Faul, G. Gregorčič, G. Boylan, W. Marnane, S. Lightbody, G. Connolly. Gaussian process modelling of EEG for the detection of neonatal seizures. *IEEE Transactions on Biomedical Engineering*, Volume 54, Issue 12, Pages: 2151 – 2162, 2007.

[85] A. Grancharova, J. Kocijan and T. A. Johansen. Explicit stohastic nonlinear predictive control based on Gaussian process models. *In Proceedings of the European Control Conference 2007*, Pages 2340-2347, Kos, 2007.

[86] G. Gregorčič and G. Lightbody. Local model identification with Gaussian processes. *IEEE Transactions on neural networks*, Volume 18, Issue 5, Pages 1404-1423, 2007.

[87] Hachino, T. Kadirkamanathan, V. Time Series Forecasting Using Multiple Gaussian Process Prior Model. *In IEEE Symposium on Computational Intelligence and Data Mining*, 2007. CIDM 2007. Pages 604-609, 2007.

[88] J. Kocijan. Identifikacija nelinearnih sistemov z Gaussovimi procesi. *Modeliranje dinamičnih sistemov z umetnimi nevronskimi mrežami in sorodnimi metodami.* Univerza v Novi Gorici, 2007, Pages 73-86. (in Slovene).

[89] J. Kocijan and K. Ažman. Gaussian Process Model Identification: A Process Engineering Case Study. *In Proceedings of the 16th International Conference on Systems Science*, Volume 1, Pages 418 - 427, Wroclaw, 2007.

[90] J. Kocijan, K. Ažman and A. Grancharova. The Concept for Gaussian Process Model Based System Identification Toolbox. *In Proceedings of the InternationalConference on Computer Systems and Technologies* - CompSysTech, Pages IIIA.23-1 - IIIA.23-6, Rousse, 2007.

[91] J. Kocijan, B. Likar. Gas-Liquid Separator Modelling and Simulation with Gaussian Process Models. *In Proceedings of the 6th EUROSIM Congress on Modelling and Simulation - EUROSIM 2007*, 7 pages, Ljubljana, 2007.

[92] W. E. Leithead, Y. Zhang. O(N-2)-operation approximation of covariance matrix inverse in Gaussian process regression based on quasi-Netwon BFGS method. *Communications In Statistics-Simulation And Computation*, Volume 36, Issue 2, Pages 367-380, 2007.

[93] B. Likar and J. Kocijan. Predictive control of a gas-liquid separation plant based on a Gaussian process model. *Computers and Chemical Engineering*, Volume 31, Issue 3, Pages 142-152, 2007.

[94] M. Neve, G. De Nicolao, and L. Marchesi. Nonparametric identification of population models via Gaussian processes. *Automatica*, Volume 43, Issue 7, Pages 1134-1144, 2007.

[95] R. Palm. Multiple-step-ahead prediction in control systems with Gaussian process models and TS-fuzzy models. *Engineering Applications of Artificial Intelligence*, Volume 20, Issue 8, Pages 1023-1035, 2007.

[96] J. M. Wang, D. J. Fleet, and A. Hertzmann. Multifactor Gaussian Process models for style-content separation. *International Conference on Machine Learning (ICML)*, Oregon, 2007.

[97] Y. Zhang, W. E. Leithead. Approximate implementation of the logarithm of the matrix determinant in Gaussian process regression. *Journal Of Statistical Computation And Simulation*, Volume 77, Issue 4, Pages 329-348, 2007.

5.1 Acknowledgements

A lot of the work has been done previously. Biggest credits go to C. E. Rasmussen, C. K. I. Williams, R. Murray-Smith and many others.

6 Gaussian Process Model Dynamic System Identification Toolbox for Matlab

6.1 Prerequisites

As this toolbox is intended to use within Matlab environment the user should have Matlab installed. It works on Matlab 7 and later, but there should be no problems using the toolbox on previous versions of Matlab, *e.g.* 6 or 5.

It is also assumed that the GPML toolbox[2], general purpose GP modelling toolbox for Matlab, is installed, as GPdyn toolbox serves as its extension.

If the functions for data preprocessing (`premnmx`) and data postprocessing (`postmnmx`) are to be used, the Matlab Neural Network has to be installed.

The user should also be (at least a little) familiar with Matlab structure and programming.

6.2 Installing GPdyn toolbox

Unzip the file GPdyn into chosen directory and add path (with subdirectories) to Matlab path.

6.3 Overview of the GPdyn toolbox

GPdyn files are contained in several directories, depending on their purpose:

basic functions, used by higher-level functions;

simulation functions, used for simulating the dynamical GP model;

lmgp functions, which are used when modelling the system with the GP model with incorporated local models (LMGP model);

supporting functions, which do not belong into other groups, but make some tasks easier;

[2]It can be obtained from *http://www.gaussianprocess.org/gpml.*

function templates, which allow user to write his own functions; they include fault detection with GP models and incorporating knowledge about coloured noise on the output of the system and hysteresis;

demo functions, which demonstrate the use of the toolbox for identification of dynamical systems.

The list of included functions, demos and one model is given in following tables.

Basic GP model functions		p.
`gpr`	- (statical) GP model prediction - data likelihood and partial derivatives	†
`traingp`	GP model training	33
`gpr_simul`	GP model regression for simulation	34
`minimize`	local minima search of the differentiable function used by `traingp`	†

Covariance functions
included and explained in GPML toolbox

GP model simulation		p.
`simul02naive`	GP model simulation without propagation of uncertainty	36
`simul00exact`	GP model simulation with analytical propagation of uncertainty	37
`gpr_SEard_exact`	GP model prediction with probabilistic inputs	38
`simul02mcmc`	GP model simulation with numerical propagation of uncertainty	??
`mcmc_getsamplesgaussmix`	sampling Gaussian mixtures	40

Note:
† . . . GPML toolbox routines, references can be found in its manual.

LMGP model simulation		p.
gpSD00	- LMGP model prediction - data likelihood and its derivatives	41
trainlmgp	LMGP model training	43
simullmgp00naive	LMGP model simulation without propagation of uncertainty	44
simullmgp00exact	LMGP model simulation with analytical propagation of uncertainty	45
gpSD00ran	LMGP model prediction with uncertain inputs	46
simullmgp00mcmc	LMGP model simulation with numerical propagation of uncertainty	48

Supporting functions		p.
add_noise_to_vector	adding white noise to noise-free simulation results	50
construct_simul_input	construction of the input for the simulation routines from system's input	51
construct_train_input	construction of the input for the training routines from system's input	52
loss2	performance measures	53
mcmc_test_pdfs	testing sampled probability distributions	54
plotgp	plot results (output and error) of the GP model prediction	55
plotgpe	plot error of the GP model prediction	56
plotgpy	plot output of the GP model prediction	57
premnmx	preprocessing of data	‡
postmnmx	postprocessing of data	‡
postmnmxvar	postprocessing of predicted variance	58
sig_prbs	generating pseudo-random binary signal	59
sig_prs_minmax	generating pseudo-random signal	60

Note:
‡ . . . part of Matlab Neural Network toolbox, references can be found there.

Function templates		p.
`get_K_clrd_noise`	construction of noise part of covariance matrix for known coloured noise	61
`simul02naive_hystOut`	simulation of the GP model with incorporated hysteresis	63
`detect_fault`	fault detection with the use of GP model	64
`detect_fault_validity`	estimate the validity of the FD	65

Demos		p.
`demo_example_present`	present the system used in demos	66
`demo_example_gp_data`	generate data for the identification and validation of the GP model	68
`demo_example_gp_training`	training of the GP model	67
`demo_example_gp_simulation`	validation with simulation of the GP model	69
`demo_example_lmgp_data`	generate data for the identification and validation of the LMGP model	70
`demo_example_lmgp_training`	training of the LMGP model	71
`demo_example_lmgp_simulation`	simulation of the LMGP model	72
`example`	system simulation	73
`example_derivative`	obtaining system's derivatives	74
`example_LM_ident`	identification of system's local models	75

6.4 How to use this toolbox

6.4.1 Demos

A simple nonlinear dynamical system is used to demonstrate the identification and simulation of the GP models:

$$y(k+1) = \frac{y(k)}{1+y^2(k)} + u^3(k) \tag{16}$$

The system was used as an example of dynamical system identification with artificial neural networks in:
K.S. Narendra and K. Parthasarathy. Identification and Control of Dynamical Systems Using Neural Networks, IEEE Transactions on NN, Vol.1 No. 1, 4-27, 1990.

demo_example_present, presents this system.

Following three demos present the identification of dynamical systems with the GP model:

demo_example_gp_data, which presents how to obtain and assemble data for identification;

demo_example_gp_training, which demonstrates the training (=identifying) the GP model;

demo_example_gp_simulation, which shows how to simulate the GP model.

The use of the GP model with incorporated local models is presented with demos:

demo_example_lmgp_data, which presents how to obtain and assemble data for identification;

demo_example_lmgp_training, which demonstrates the training (=identifying) the LMGP model;

demo_example_lmgp_simulation, which shows how to simulate the LMGP model.

6.4.2 Templates

Templates are used to facilitate the incorporation of different prior knowledge into the GP model and to present how the GP model can be used for fault detection.

Function `get_K_clrd_noise` returns the "noise part" of covariance matrix, when the system output is corrupt with known coloured Gaussian noise and not usually assumed white Gaussian noise. The returned noise part of covariance matrix must be added to the "function part" of covariance matrix.

Function `simul00naive_hystOut` demonstrates how to simulate the GP model, where the D-th regressor represent the state of the hysteresis on the output.

In Dj. Juričić and J. Kocijan. Fault detection based on Gaussian process model. In I. Troch and F. Breitenecker, editors, *Proceedings of the 5th Vienna Symposium on Mathematical Modeling – MathMod*, Dunaj, 2006,
it has been shown that the GP models can be used for fault detection. The function `detect_fault` is used for fault detection and the function `detect_fault_validity` is used to estimate the confidence we can have in the prediction.

6.5 Notes

The GPML package[3] is relatively new and introduces a new way of calling basic prediction routine `gpr`. In previous version the routines used for dynamical GP modelling had only the possibility to use Gaussian covariance function (now `covSEard`) with white noise on the output of the system (now `covNoise`) and the necessary calculations were done directly in the prediction routine (now `gpr`). The basic functions have already been recoded to employ the new change, but in more complex (i.e. simulation with analytical propagation of variance `simul00exact`, LMGP functions, "coloured noise" covariance function `covClrdNoise`) this change still has to be made.

6.6 Future work

In future work the needed improvements, listed in previous subsection, have to be considered.

[3]Rasmussen and Williams, *http://www.gaussianprocess.org/gpml.*

7 Toolbox reference

This section gives a list of all functions, templates and demos constituting the model with their names, syntax and basic instructions.

traingp

```
function [logtheta, flogtheta, i] = traingp(covfunc, input, target, logtheta0)
```

Syntax

function [logtheta, flogtheta, i] = traingp(covfunc, input, target, logtheta0)

Description

Function for optimization (training) of the GP model hyperparameters based on the training data via Maximum Likelihood (ML).
Uses routines gpr and minimize.
Based on the work of C.E.Rasmussen.

Inputs:
covfunc .. specified covariance function, see help covFun for more info
input .. input part of the training data, NxD matrix
target .. output part of the training data (ie. target), Nx1 vector
logtheta0 .. intial values of hyperparameters (optional)
Outputs:
logtheta .. optimized hyperparameters
flogtheta .. minus log likelihood for the different runs (init. to 0)
i .. number of iterations needed for the last optimization

Examples

demo_example_gp_training.m

See Also

GPR, MINIMIZE, COVFUN, TRAINLMGP

gpr_simul

Syntax

```
function [out1, out2, out3, out4] = gpr_simul(logtheta, covfunc, x, y, xstar,
alpha, L)
```

Description

gpr_simul - Gaussian process regression, with a named covariance function. Two modes are possible: training and prediction: if no test data are given, the function returns minus the log likelihood and its partial derivatives with respect to the hyperparameters; this mode is used to fit the hyperparameters. If test data are given, then (marginal) Gaussian predictions are computed, whose mean and variance are returned. Note that in cases where the covariance function has noise contributions, the variance returned in S2 is for noisy test targets; if you want the variance of the noise-free latent function, you must substract the noise variance. The rutine is modified to calculate the inverse covariance matrix only once to speed-up simulation.

usage: [nlml dnlml] = gpr(logtheta, covfunc, x, y)
or: [mu S2 alpha L] = gpr(logtheta, covfunc, x, y, xstar, alpha, L)

where:

logtheta is a (column) vector of log hyperparameters
covfunc is the covariance function
x is a n by D matrix of training inputs
y is a (column) vector (of size n) of targets
xstar is a nn by D matrix of test inputs
alpha is inv(covariance matrix)y
L is cholesky(covariance matrix)
nlml is the returned value of the negative log marginal likelihood
dnlml is a (column) vector of partial derivatives of the negative
log marginal likelihood wrt each log hyperparameter
mu is a (column) vector (of size nn) of prediced means
S2 is a (column) vector (of size nn) of predicted variances

For more help on covariance functions, see "help covFunctions".

simul02naive

Syntax

```
function [y, s2] = simul02naive(logtheta, covfunc, input, target, xt,lag)
```

Description

"Naive"(i.e. without propagation of variance) simulation of the GP model.
Uses routine gpr.

Inputs:
loghteta .. optimized hyperparameters
covfunc .. specified covariance function, see help covFun for more info
input .. input part of the training data, NxD matrix
target .. output part of the training data (ie. target), Nx1 vector
xt .. input matrix for simulation, kxD vector, see
construct_simul_input.m for more info
lag .. the order of the model (number of used lagged outputs)
Outputs:
y .. mean predicted output
s2 .. associated variances

Examples

demo_example_gp_simulation.m

See Also

GPR_SIMUL, SIMUL00EXACT, SIMUL02MCMC

simul00exact

Syntax

```
[mu, sig2, m, s2] = simul00exact(logtheta, covfunc, input, target, xt, lag)
```

Description

Simulation of the GP model, where the output variance is propagated using analytical approximation, see A. Girard, Approximate Methods for Propagation of Uncertainty with Gaussian Process Models, PhD thesis, 2004.
Notes:
Currently it can be used only with Gaussian covariance function and with white noise model (sum of covSEard and covNoise).

Uses routine gpr_SEard_exact.

Inputs:
loghteta .. optimized hyperparameters
covfunc .. specified covariance function, see help covFun for more info
input .. input part of the training data, NxD matrix
target .. output part of the training data (ie. target), Nx1 vector
xt .. input matrix for simulation, kxD vector, see construct_simul_input.m for more info
lag .. the order of the model (number of used lagged outputs)
Outputs:
mu .. predictive mean using "naive"approach (doesn't propagate the uncertainty)
sig2 .. predictive variance using "naive"approach
m .. predictive mean when propagating the uncertainty
s2 .. predictive variance when propagating the uncertainty
The form of the covariance function is
$C(x^p, x^q) = v_1\ exp\left(-\frac{1}{2}\sum_{d=1}^{D} w_d(x_d^p - x_d^q)^2\right) + v_0\ \delta_{pq}$
(computed using cov2.m)
Based on the work of R. Murray-Smith and A. Girard.

Examples

demo_example_gp_simulation.m

See Also

SIMUL02NAIVE, GPR_SEARD_EXACT

gpr_SEard_exact

Syntax

```
[m, S2] = gpr_SEard_exact(logtheta, covfunc, invQ, input, target, muX, vy, SigX,
lag)
```

Description

This function computes the predictive mean and variance at test input If SigX: consider random input, with covariance SigX. Predictions computed using the exact equations. The form of the covariance function is $C(x^p, x^q) = v_1\ exp\left(-\frac{1}{2}\sum_{d=1}^{D} w_d(x_d^p - x_d^q)^2\right) + v_0\ \delta_{pq}$ (computed using cov2.m)

Inputs:
loghteta .. optimized hyperparameters
covfunc .. dummy, used for (eventual) future compatibility
invQ .. inverse of the data covariance matrix
input .. input part of the training data, NxD matrix
target .. output part of the training data (ie. target), Nx1 vector
muX .. D by 1 test input
vy .. output noise variance (either known or learnt - then "[]")
SigX .. covariance of the test input (OPTIONAL)
lag .. the order of the model (number of used lagged outputs)
Outputs:
m ..predicted mean
S2 .. predicted variance (noise free)

Based on the work of J. Quinonero-Candela and A. Girard.

Examples

demo_example_gp_simulation.m

See Also

SIMUL00EXACT

simul02mcmc

Syntax

```
function [y, s2] =
   simul02mcmc(logtheta, covfunc, input, target, xt,lag)
```

Description

Simulation of the GP model, where the output variance is propagated using simple MCMC method, see A. Girard, Approximate Methods for Propagation of Uncertainty with Gaussian Process Models, PhD thesis, 2004.
Idea:
at every time step the output of GP model is approximated with Nsamples samples, which are used as the future inputs of the GP model. Samples are re-used if necessary (ie. y(k-1) for y(k-2) if lag=2 etc.)
Uses routines gpr and mcmc_getsamplesgaussianmix.

Inputs:
loghteta .. optimized hyperparameters
covfunc .. specified covariance function, see help covFun for more info
input .. input part of the training data, NxD matrix
target .. output part of the training data (ie. target), Nx1 vector
xt .. input matrix for simulation, kxD vector, see construct_input_matrix.m for more info
lag .. the order of the model (number of used lagged outputs)
Nsamples .. number of samples used in algorithm (ie. runs of simulation)
Outputs:
mu .. mean predicted output
s2 .. asociated variances
MU .. matrix of all predicted means, kxNsamples
SIG2 .. associated predicted variances
indexes_warning .. sampling problems, see mcmc_getsamplesgaussianmix.m

Written by K.Azman, 31.05.2005
Based on the work of C.E. Rasmussen and A. Girard.

Examples

demo_narendra_gp_simulation.m

See Also

GPR_SIMUL, MCMC_GETSAMPLESGAUSSIANMIX, SIMUL02NAIVE

mcmc_getsamplesgaussmix

Syntax

```
function [x,y,indexes,samples,xsamples,fwarning] =
   mcmc_getsamplesgaussmix(x,Mi,Sig,N)
```

Description

Function returns the samples from the mixtures of Gaussian distributions, given by Mi and Sig and also estimates the quality of the sampled distribution. First the cumulative distribution is estimated, then the inverse transform method with the use of Matlab random ganerator is used to obtain individual samples.
Inputs:
x .. partitioned input space
Mi .. vector of Gaussian mixture means
Sig .. vectors of Gaussian mixture standard deviations
N ... number of returned samples
Outputs:
x .. partitioned inpout space
y .. probabitily density function (pdf) of Gaussian mixture
indexes .. indexes of chosen samples, should be applied to x to get the values
samples .. number of samples at particular elements of x - for testing the quality of the approximated distribution
xsamples .. indexes applied to x (samples)
fwarning .. warning flag, risen if less than 98% of distribution is covered with samples

Function uses rand (internal Matlab command) and mcmc_getgaussmix,
defined at the end of the function. It is used by the function
simul00mcmcm.

See Also

RAND, SIMUL00MCMC

gpSD00

Syntax `function [out1, out2, out3, out4] = gpSD00(X, input, target, targetvariance, derivinput, derivtarget, derivvariance, test)`

Description

This function computes the predictive mean and variance at test input for the LMGP model with the covariance function as sum of covSEard and covNoise.
Usage:
[fX, dfX] = gpSD00(X, input, target, derivinput,derivtarget, derivvariance)
or:
[mu, S2, muderiv, S2deriv] = gpSD00(X, input, target, derivinput, derivtarget, derivvariance, test)

Inputs:
X .. a (column) vector (of size D+2) of hyperparameters
input .. a n by D matrix of training inputs
target .. a (column) vector (of size n) of targets
targetvariance .. a (column) vector (of size n) of variances of targets. Unknown variances are indicated by NaN - these are then replaced by the noise term in the covariance function
derivinput .. an n by D matrix of training inputs at which we have derivative information (not necessarily the same as 'input').
derivtarget .. an n by D matrix of partial derivatives at 'derivinput', w.r.t. each input
derivvariance .. an n by D^2 matrix, where each row is the elements of the covariance matrix associated with the appropriate derivtarget
test .. a nn by D matrix of test inputs
Outputs:
fX .. the returned value of minus log likelihood
dfX .. a (column) vector (of size D+2) of partial derivatives of minus the log likelihood wrt each of the hyperparameters
mu .. a (column) vector (of size nn) of prediced means
S2 .. a (column) vector (of size nn) of predicted variances
where D is the dimension of the input. The form of the covariance function is
$C(x^p, x^q) = v^2 \; exp\left(-(x^p - x^q)^T P^{-1}(x^p - x^q)/2\right) + u^2 \; \delta_{pq}$
where the first term is the squared negative exponential and the second term with the kronecker delta is the noise contribution. The P matrix is diagonal with "Automatic

Relevance Determination"(ARD) or "input length scale"parameters $w_1^2, ..., w_D^2$; The hyperparameter v is the "signal std dev" and u is the "noise std dev". All hyperparameters are collected in the vector X as follows: $X = [\log(w_1) \log(w_2) \ldots \log(w_D) \log(v) log(u)]^T$

Note: the reason why the log of the parameters are used in X is that this often leads to a better conditioned (and unconstrained) optimization problem than using the raw hyperparameters themselves.

This function can conveniently be used with the "minimize" function to train a Gaussian process:

(C) Copyright 1999 - 2003, Carl Edward Rasmussen (2003-07-24). Derivative adaptation, Roderick Murray-Smith (2003-07-25). Can now cope with a number of derivative observations independent of the number of function observations. It has separate noise level for the derivative observations.

Examples
demo_example_lmgp_simulation.m

See Also
SIMULLMGP00NAIVE, SIMULLMGP00MCMC, MINIMIZE, GPSD00RAN

trainlmgp

Syntax

```
function [logtheta, flogtheta, i] = trainlmgp(covfunc, input, target, inputDer,
targetDer, deriveVar, logtheta0)
```

Description

Function for optimization (training) of the LMGP model (GP model v local information) hyperparameters based on the training data via Maximum Likelihood (ML).
Note:
currently it can be used only with Gaussian covariance function and with white noise model (sum of covSEard and covNoise). Uses routines gpSD00 and minimize.

Based on the work of C.E.Rasmussen and R. Murray-Smith.

Inputs:
covfunc .. specified covariance function, see help covFun for more info
input .. input part of the training data, NxD matrix
target .. output part of the training data (ie. target), Nx1 vector
derivinput ..input part of the derivative training data, NEQxD matrix
derivtarget .. target derivatives, NEQxD matrix
derivevariance .. variances of the local model prameters, NEQxD matrix
logtheta0 .. intial values of hyperparameters (optional)
Outputs:
loghteta .. optimized hyperparameters
flogtheta .. minus log likelihood for the different runs (init. to 0)
i .. number of iterations needed for the last optimization

Examples

demo_example_lmgp_training.m

See Also

MINIMIZE, GPSD00, TRAINGP

simullmgp00naive

Syntax `function [mu, s2] = simullmgp00naive(logtheta, covfunc, input, target, targetvariance,... derivinput, derivtarget, derivvariance, xt, lag)`

Description
"Naive"(i.e. without propagation of variance) simulation of the GP model with the incorporated local models (LM).
Note:
Currently it can be used only with Gaussian covariance function and with white noise model (sum of covSEard and covNoise).
Uses routine gpSD00.

Inputs:
loghteta .. optimized hyperparameters
covfunc .. specified covariance function, see help covFun for more info
input .. input part of the training data, NxD matrix
target .. output part of the training data (ie. target), Nx1 vector
targetvariance .. target variance, use NaN where not known
derivinput .. input part of the derivative training data, NEQxD matrix
derivtarget .. target derivatives, NEQxD matrix
derivvariance .. variances of the local model prameters, NEQxD matrix
xt .. input matrix for simulation, kxD vector, see
construct_simul_input.m for more info
lag .. the order of the model (number of used lagged outputs)
Outputs:
mu .. mean predicted output
s2 .. asociated variances

Based on the work of R. Murray-Smith and A. Girard.

Examples
demo_example_lmgp_simulation.m

See Also
GPSD00, SIMULLMGP00EXACT, SIMULLMGP00MCMC, SIMUL02NAIVE

simullmgp00exact

Syntax `[mu, s2] = simullmgp00exact(logtheta, covfunc, input, target, targetvar, derivinput, derivtarget, derivvariance, xt, lag)`

Description
Simulation of the GP model with incorporated local models (LMGP models), where the output variance is propagated using analytical approximation, see J. Kocijan, A. Girard, and D. J. Leith. Incorporating linear local models in Gaussian process model. Technical Report DP-8895, Institut Jožef Stefan, Ljubljana, December 2003.
A. Girard, Approximate Methods for Propagation of Uncertainty with Gaussian Process Models, PhD thesis, 2004.
Notes:
Currently it can be used only with Gaussian covariance function and with white noise model (sum of covSEard and covNoise).

Uses routine gpSD00ran.

Inputs:
loghteta .. optimized hyperparameters
covfunc .. specified covariance function, see help covFun for more info
input .. input part of the training data, NxD matrix
target .. output part of the training data (ie. target), Nx1 vector
targetvariance .. target variance, use NaN where not known
derivinput .. input part of the derivative training data, NEQxD matrix
derivtarget .. target derivatives, NEQxD matrix
derivvariance .. variances of the local model prameters, NEQxD matrix
xt .. input matrix for simulation, kxD vector, see construct_simul_input.m for more info
lag .. the order of the model (number of used lagged outputs)
Outputs:
mu .. mean predicted output
s2 .. asociated variances

Examples
demo_example_lmgp_simulation.m
See Also
GPSD00RAN, SIMULLMGP00NAIVE, SIMULLMGP00MCMC, SIMUL00EXACT

gpSD00ran

Syntax `function [mu, S2, invQ] = gpSD00ran(X, input, target, targetvariance, derivinput, derivtarget, derivvariance, test, SigmaX)`

Description
This function computes the predictive mean and variance at test input for the LMGP model with the covariance function as sum of covSEard and covNoise. The prediction and propagation of variance is done by this routine. When test data are given, then (marginal) Gaussian predictions are computed, whose mean and (noise free) variance are returned.

Inputs:
X .. a (column) vector (of size D+2) of hyperparameters
input .. a n by D matrix of training inputs
target .. a (column) vector (of size n) of targets
targetvariance .. a (column) vector (of size n) of variances of targets unknown variances are indicated by NaN – these are then replaced by the noise term in the covariance function
derivinput .. an n by D matrix of training inputs at which we have
derivative information (not necessarily the same as 'input').
derivtarget .. an n by D matrix of partial derivatives at 'derivinput', w.r.t. each input
derivvariance .. an n by $D\hat{2}$ matrix, where each row is the elements of the covariance matrix associated with the appropriate derivtarget
test .. a nn by D matrix of test inputs
SigmaX .. covariance of the test input
Outputs:
mu .. a (column) vector (of size nn) of prediced means
S2 .. a (column) vector (of size nn) of predicted variances
invQ .. the inverse or covariance matrix.

derivative adaptation, Roderick Murray-Smith (2003-07-25). Can now cope
with a number of derivative observations independent of the number of
function observations. It has seperate noise level for the derivative
observations.
Variance propagation upgrade, Jus Kocijan, 2003

Examples

demo_example_lmgp_simulation.m

See Also

SIMULLMGP00NAIVE SIMULLMGP00MCMC, MINIMIZE, GPSD00

simullmgp00mcmc

Syntax `[mu, s2, MM, VV] = simullmgp00mcmc(logtheta, covfunc, input, target, targetvariance,... derivinput, derivtarget, derivvariance, xt, lag, Nsamples)`

Description
Simulation of the GP model with incorporated local models (LMGP models), where the output variance is propagated using numerical MCMC approximation, see K. Ažman. Identifikacija dinamičnih sistemov z Gaussovimi procesi. PhD thesis, Univerza v Ljubljani, Ljubljana, 2007. (in Slovene).
Idea:
at every time step the output of GP model is approximated with the Gaussian distribution, from which we sample one value. We repeat the procedure Nsamples-times. Nsamples samples, which are used as the future inputs of the GP model. Samples are re-used if necessary (ie. y(k-1) for y(k-2) if lag=2 etc.)
Notes:
(1) Currently it can be used only with Gaussian covariance function and with white noise model (sum of covSEard and covNoise) due to the gpSD00.
Uses routine gpSD00.

Inputs:
loghteta .. optimized hyperparameters
covfunc .. specified covariance function, see help covFun for more info
input .. input part of the training data, NxD matrix
target .. output part of the training data (ie. target), Nx1 vector
targetvariance .. target variance, use NaN where not known
derivinput .. input part of the derivative training data, NEQxD matrix
derivtarget .. target derivatives, NEQxD matrix
derivvariance .. variances of the local model prameters, NEQxD matrix
xt .. input matrix for simulation, kxD vector, see construct_simul_input.m for more info
lag .. the order of the model (number of used lagged outputs)
Nsamples .. number of samples used in algorithm (ie. runs of simulation)
Outputs:
mu .. mean predicted output
s2 .. associated variances
MM.. matrix of all predicted means, kxNsamples

VV .. associated predicted variances

Written by K.Azman, 31.05.2005
Based on the work of C.E. Rasmussen and A. Girard.

Examples
demo_example_lmgp_simulation.m

See Also
GPSD00, SIMULLMGP00NAIVE, SIMULLMGP00EXACT, SIMUL02MCMC

add_noise_to_vector

Syntax

```
function [x_noise,x_no_noise] = add_noise_to_vector(x_no_noise, noise_std)
```

Description

Function adds white Gaussian noise to vector given as a parameter. It is used to get noisy data from simulation noise-free data.

Inputs:
x_no_noise .. vector of noise-free data
noise_std .. standard deviation of white Gaussian noise
Outputs:
x_noise .. noisy data
x_no_noise .. original data

Function uses randn (internal Matlab command) to obtain sampled Gaussian distribution.

Examples

demo_example_gp_data.m

See Also

RANDN

construct_simul_input

Syntax

`function xt = construct_simul_input(lag,u,x0)`

Description

Function contructs matrix for GP model simulation from input signal and starting values of output of the system. Shape of the matrix:
xt = [y(1)... y(lag) u(1) ... u(lag);
y(2)... 0 u(2) ... u(lag+1);
⋮
0 ... 0 u(end-lag)... u(end-1)];
Note:
first u(-lag+1)=u(0) and the matrix is correspondingly shorter!
Inputs:
lag .. order of the system
u .. system input signal u, u = [u(1) ... u(end)]'
x0 .. starting values of system state (=output), x0 = [y(1) ... y(lag)]
Outputs:
xt .. matrix used as the input to simulation rutines

Examples

demo_example_gp_simulation.m

See Also

SIMUL02NAIVE, SIMUL00EXACT, etc.

construct_train_input

Syntax

`function [target,tinput] = construct_train_input(lag,u,x0)`

Description

Function contructs matrix of regressors for GP model training from input and output signals of the system. Shape of the matrix:
tinput = [y(1) ... y(lag) u(1) ... u(lag);
y(2) ... y(lag+1) u(2) ... u(lag+1);
⋮
y(end-lag) ... y(end-1) u(end-lag) ... u(end-1)];

target = [y(lag+1) y(lag+2) ... y(end)]';

Inputs:
lag .. order of the system
u .. system input signal u, u = [u(1) ... u(end)]'
y .. system output signal y, y = [y(1) ... y(end)]'
Outputs:
target.. target vector for the training routines
tinput .. matrix of regressors for the training rutines

Examples

demo_example_gp_simulation.m

See Also

SIMUL02NAIVE, SIMUL00EXACT, etc.

loss2

Syntax

```
function [ae, se, lpd, mrse] = loss2(tt, mu, s2)
```

Description

Function computes several frequently used performance values:
- mean absolute error AE
- mean squared error SE
- minus log-predicted density error LPD
- mean relative squared error MRSE

Notes:
(1) It can be used with one-step-ahead prediction or simulation results.
(2) Can be expanded with other performance values.

Inputs:
tt .. target output values (=system output)
y .. predicted output values (=model output)
s2 .. predicted output variance

Outputs:
ae .. mean absolute error
se .. mean squared error
lpd .. minus log-predicted density error
mrse .. mean relative square error

mcmc_test_pdfs

Syntax

`function [] = mcmc_test_pdfs(MM,VV,desired_steps)`

Description

Function test and plots the Gaussian mixture given my MM and VV in desired simulation steps

Inputs:
MM .. mean values of predicted distributions, matrix ksteps times Nsamples
VV .. associated variances of predicted distributions
desired_steps .. vector with the indices of steps, where we want to test the distributions
Outputs:
.. plots the figures of the distributions in the desired steps

See Also

SIMUL00EXACT

plotgp

Syntax

```
function [i] = plotgp(i, t, sys, y, std)
```

Description

Function plots the results of the simulation or one-step-ahead prediction. In the upper 2/3 of the figure the output together with 95% confidence band and target is plotted, in the lower third of the figure the error with corresponding 95% confidence band is plotted. Can be used to plot in new or currently active figure.

Inputs:
i .. figure handle, if i==0 function plots in current figure
t .. time vector (x-axis)
sys .. target output
y .. predicted output
std .. predicted standard deviation
Output:
i .. figure handle

Examples

demo_example_gp_simulation.m

See Also

PLOTGPY, PLOTGPE

plotgpe

Syntax

```
function [i] = plotgpe(i, t, sys, y, std)
```

Description

Function plots the error with corresponding 95% confidence band.
Can be used to plot in new or currently active figure.

Inputs:
i .. figure handle, if i==0 function plots in current figure
t .. time vector (x-axis)
sys .. target output
y .. predicted output
std .. predicted standard deviation
Output:
i .. figure handle

Examples

demo_example_gp_simulation.m

See Also

PLOTGP, PLOTGPY

plotgpy

Syntax

```
function [i] = plotgpy(i, t, sys, y, std)
```

Description

Function plots the results of the simulation or one-step-ahead prediction: target and output together with 95% confidence band.

Inputs:
i .. figure handle, if i==0 function plots in current figure
t .. time vector (x-axis)
sys .. target output
y .. predicted output
std .. predicted standard deviation
Output:
i .. figure handle

Examples

demo_example_gp_simulation.m

See Also

PLOTGP, PLOTGPE

postmnmxvar

Syntax

```
function s2 = postmnmxvar(s2n,min,max)
```

Description

Postprocesses predicted variance for data which has been preprocessed by PREMNMX.

Inputs:
s2n .. normalized predicted variance
min .. minimum of preprocessed data
max .. maximum of preprocessed data
Output:
s2 .. postprocessed predicted variance

See Also

POSTMNMX, PREMNMMX

sig_prbs

Syntax

```
function y = sig_prbs(n,m)
```

Description

Function generates pseudo-random binary signal (PRBS) with uniform amplitude, see D. Matko: Identifikacije, 1998, FE Ljubljana, p. 43-45

Inputs:
n .. legth of the register
m .. number of samples per tact of PRBS (the width of the narrowest impulse at sampling steps)
Output:
y .. signal values

sig_prs_minmax

Syntax

```
function y = sig_prs_minmax(n, ksw, min, max)
```

Description

Function, which generates “random” signal with length n and uniform distribution of amplitudes between min and max. ksw is the minimum width (duration in steps) of the signal at particular amplitude.

Inputs:
n .. length of the signal
ksw .. tact (minimum duration of signal at constant value)
min .. minimum amplitude
max .. maximum amplitude
Output:
y .. signal values

Examples

demo_example_gp_data.m

get_K_clrd_noise

Syntax

```
function [Cn, Cn_vec] = get_K_clrd_noise(NN, a, b, noisevar)
```

Description

Function returns the noise part of covariance matrix, when the coloured noise on the output of the system is known. This covariance matrix should be added to its function part. The function folows the derivation in [1] and differs a bit from [2].
Idea:
AR part (ai-s) :
first na elements of the covariance matrix calculated using Yule-Walker
equations, the rest of them (na+1 .. NN) computed iteratively
MA part (bi-s) :
the elements of covariance matrix up to bn are calculated, the rest of
them equals 0
Notes:
(1) K(tau) = K(i,j), where $|i-j| = |j-i| = tau$
(2) Function is not compatible with the current version of the GPML
toolbox.
(3) Function uses the function TOEPLITZ.

Inputs:
NN .. size of the covariance matrix (number of training data)
a ... vector of AR elements of correlated-noise (1 a_1 a_2 .. a_na)
b ... vector of MA elements of correlated-noise (b_0 b_1 b_2 .. b_nb)
usually b_0=0
noisevar ... noise variance sigma_e^2
Outputs:
Cn .. noise part of the covariance matrix
Cn_vec .. elements of the covariance matrix in a vector

See Also

[1] K. Azman, J. Kocijan, Identifikacija dinamičnega sistema z znanim modelom
šuma z modelom na osnovi Gaussovih procesov, n: B. Zajc, A. Trost (eds.)
Zbornik petnajste mednarodne Elektrotehniške in računalniške konference
ERK 2006, 25.-27.09., Portorož, Slovenija, 2006, Vol. A, pp. 289–292,

2006. (in Slovene)
[2] R. Murray-Smith and A. Girard. Gaussian Process Priors with ARMA noise models. In Proceedings: Irish Signals and Systems Conference, 147-153, Maynooth, 2001.

simul02naive_hystOut

Syntax

```
function [y, s2, hyst] = simul02naive_hystOut(logtheta, covfunc, input, target,
   xt,lag, hystProp)
```

Description

Template for function for "naive"(i.e. without propagation of variance) simulation of the GP model with hysteresis on the output. Uses routine gpr and works similar to SIMUL00NAIVE.
See K. Azman, J. Kocijan, Identifikacija dinamičnega sistema s histerezo z modelom na osnovi Gaussovih procesov. In: B. Zajc, A. Trost (eds.) Zbornik štirinajste mednarodne Elektrotehniške in računalniške konference ERK 2005, 26.-28.09., Portorož, Slovenija, 2005, Vol. A, pp. 253–256. (in Slovene)

Inputs:
loghteta .. optimized hyperparameters
covfunc .. specified covariance function, see help covFun for more info
input .. input part of the training data, NxD matrix, D-th regressor represents the state of hysteresis
target .. output part of the training data (ie. target), Nx1 vector
xt .. input matrix for simulation, kxD vector, see construct_simul_input.m for more info
lag .. the order of the model (number of used lagged outputs)
hystProp .. hysteresis properties:
  .limits .. [xmin xmax] values, where the state of hysteris changes
  .DYH .. change of the output due to hysteresis
  .state .. initial state of hysteresis
Outputs:
y .. mean predicted output
s2 .. corresponding variances with added white noise v0
hyst .. corresponding states of the hysteresis

See Also

GPR_SIMUL, SIMUL02NAIVE

detect_fault

Syntax

```
function [index_bool, index_val] = detect_fault(residuals, var_residuals, NW,
    c_max)
```

Description

Function for detecting the fault, where the system is modelled with the GP model. It takes into account predicted mean as well as the predicted variance. Greater window size can be selected to increase robustness on the account of lower sensibility.
Note:
Function is not compatible with the current version of the GPML toolbox.
[1] Dj.Juricic, J.Kocijan,Fault detection based on Gaussian process model,
I.Troch, F.Breitenecker (eds.), Proceedings of the 5th Vienna
Symposium on Mathematical Modelling – MathMod, Wien, 2006

Inputs:
residuals .. [eps(1) .. eps(n)], where eps(k) = $y(k) - \hat{y}(k) = y(k) - m(k)$
var_residuals .. [v(1) .. v(n)], where w(k) = var(eps(k)) predicted variance associated with $\hat{y}(k)$, i.e. GP prediction at time step k
NW .. size of window, used for calculating the index and deciding about fault
c_max .. maximal significance level still reflecting in deciding for H0 (no fault)
Outputs:
index_bool: 1 for H1 (fault) and 0 for H0 (no fault)
index_val: significance levels, corresponding to index_bool

See Also

DETECT_FAULT_VALIDITY, SIMUL00MCMC

detect_fault_validity

Syntax

```
function [index_validity, index_validity_window] =
    detect_fault_validity(X, input, target, NW, var_residuals)
```

Description

Function, which estimates the degree at which we can trust the results of the function DETECT_FAULT. To measure the quality of the identified model particular region, the GP model prediction variances are used.

Note:

(1) Function is not compatible with the current version of the GPML toolbox.

(2) The simulation methods which propagate the variance are recommended, eg. simul00mcmc etc.

[1] Dj.Juricic, J.Kocijan, Fault detection based on Gaussian process model, I.Troch, F.Breitenecker (eds.), Proceedings of the 5th Vienna Symposium on Mathematical Modelling – MathMod, Wien, 2006

Inputs:

X, input, target.. standard GP model inputs

NW .. window size

var_residuals .. vector of predicted variances

var_residuals .. [v(1) .. v(n)], where w(k) = var(eps(k)) predicted variance associated with $\hat{y}(k)$, i.e. GP prediction at time step k

NW .. size of window, used for calculating the index and deciding about fault

c_max .. maximal significance level still reflecting in deciding for H0 (no fault)

Outputs:

index_validity: validity index for time steps Ik

(Ik=0 $\rightarrow$ not valid, Ik=1 $\rightarrow$ valid)

index_validity_window: validity index Ik averaged over window NW

See Also

DETECT_FAULT, SIMUL00MCMC

demo_example_present

Description

Demo to present the dynamics of the chosen nonlinear dynamic system:
$y(k+1) = \frac{y(k)}{1+y^2(k)} + u^3(k)$
which is used to demonstrate the use of this toolbox.
[1] K.S. Narendra and K. Parthasarathy. Identification and Control of Dynamical Systems Using Neural Networks, IEEE Transactions on NN, Vol.1 No. 1, 4-27, 1990.

See Also

EXAMPLE, DEMO_EXAMPLE_GP_DATA, DEMO_EXAMPLE_GP_TRAINING, DEMO_EXAMPLE_GP_SIMULATION

demo_example_gp_training

Description

Demo to present the how to train (=identify) the GP model, which describes the nonlinear dynamic system.

See Also

EXAMPLE, DEMO_EXAMPLE_GP_DATA, DEMO_EXAMPLE_GP_SIMULATION, TRAINGP, GPR

demo_example_gp_data

Description

Demo to present how to obtain and compose data for the identification of the GP model.

See Also

EXAMPLE, DEMO_EXAMPLE_PRESENT, DEMO_EXAMPLE_GP_TRAINING, DEMO_EXAMPLE_GP_SIMULATION

demo_example_gp_simulation

Description

Demo to present the simulation of the GP model, describing dynamic system. Three different simulations are presented:
(1) “naive” simulation (without propagation of uncertainty)
(2) “exact” simulation (anayltical propagation of uncertainty)
(3) simulation with numerical (MCMC) propagation of uncertainty.
See:
A. Girard, Approximate Methods for Propagation of Uncertainty with Gaussian Process Models, PhD thesis, 2004.
for more info.

Set flags accordingly.

demo_example_lmgp_data

Description

Demo to present how to obtain and compose data for the identification of the GP model with incorporated local models (LMGP model).

There are two options how to obtain local model data:
a. analyticaly (set fIdentifyLm to 0)
b. with identification (set fIdentifyLm to 1)

Note:
Currently it can be used only with Gaussian covariance function and with white noise model (sum of covSEard and covNoise).

See Also

EXAMPLE, DEMO_EXAMPLE_LMGP_TRAINING,
DEMO_EXAMPLE_LMGP_SIMULATION, EXAMPLE_LM_IDENT,
EXAMPLE_DERIVATIVE

demo_example_lmgp_training

Description

Demo to present the how to train (=identify) the LMGP model.
Note:
Currently it can be used only with Gaussian covariance function and with white noise model (sum of covSEard and covNoise).

See Also

DEMO_EXAMPLE_LMGP_DATA, DEMO_EXAMPLE_LMGP_SIMULATION, TRAINLMGP, GPSD00

demo_example_lmgp_simulation

Description

Demo to present the simulation of the LMGP model of the chosen dynamic system. Three different simulations are presented:
(1) "naive" simulation (without propagation of uncertainty)
(2) "exact" simulation (anayltical propagation of uncertainty)
(3) simulation with numerical (MCMC) propagation of uncertainty.
See:
A. Girard, Approximate Methods for Propagation of Uncertainty with Gaussian Process Models, PhD thesis, 2004.
for more info.

Set flags accordingly.

Note:
Currently it can be used only with Gaussian covariance function and
with white noise model (sum of covSEard and covNoise).

See Also

EXAMPLE, DEMO_EXAMPLE_LMGP_DATA, DEMO_EXAMPLE_LMGP_TRAINING, SIMULLMGP00NAIVE, SIMULLMGP00EXACT, SIMULLMGP00MCMC, GPSD00, GPSD00RAN

example

Syntax

`function [Y, X, U] = example(U, varargin)`

Description

Function simulating the nonlinear dynamic system:
$y(k+1) = \frac{y(k)}{1+y^2(k)} + u^3(k)$
which is used to demonstrate the use of this toolbox.
[1] K.S. Narendra and K. Parthasarathy. Identification
and Control of Dynamical Systems Using Neural Networks,
IEEE Transactions on NN, Vol.1 No. 1, 4-27, 1990.

Usage:
(1) [Y, X, U] = example(u, y0), u is vector, y0 is optional
(2) [Y, X, U] = example(U, y0), U is Nx2 matrix
Inputs:
u .. vector of inputs
U .. matrix U=[x u], where x is vector of past states nad u is vector of inputs
y0 .. intial state of the system, optional
Outputs:
Y .. vector of outputs
X .. vector of past outputs: X(k) = Y(k-1)
U .. vector of inputs

Examples

demo_example_gp_data.m

See Also

EXAMPLE_DERIVATIVE, EXAMPLE_LM_IDENT

example_derivative

Syntax

`[dFdY, dFdU, Y, U] = example_derivative(U)`

Description

Function which returns partial derivatives df/du and df/dy of the nonlinear dynamic system:
$y(k+1) = \frac{y(k)}{1+y^2(k)} + u^3(k)$
in equilibrium points.

Input:
U .. vector, which defines the equilibrium points with the input u.
Outputs:
dFdY .. partial derivatives df(u,y)/dy
dFdU .. partial derivatives df(u,y)/du
Y .. corresponding f(y) in derivative points
U .. corresponding u in derivative points

Examples

demo_example_gp_data.m

See Also

EXAMPLE, EXAMPLE_LM_IDENT

example_LM_ident

Syntax

`function [lm, Yeq] = example_LM_ident(Ueq, dU, noiseStd, test_flag)`

Description

Function which identifies local models for example system (see EXAMPLE.m) in given equilibrium points using Instrumented Variables (IV4) algortihm.

Inputs:
Ueq .. equilibrium points where we would like to identify local models
dU .. amplitude of the PRBS signal used for the perturbation of the system in working points
noiseStd .. noise standard deviation of the system's modelled output
test_flag .. flag for ploting the identification results,
flag=i → plot figure(i), flag=0 → no figure
Outputs:
lm .. structure of local models, each model has the Matlab identification toolbox structure
Yeq .. values of the output in corresponding equilibrium points

Examples

demo_example_gp_data.m

See Also

EXAMPLE, EXAMPLE_DERIVATIVE