Recursive Gaussian Process Regression with Integrated Monotonicity Assumptions for Control Applications

Ricus Husmann[a], Sven Weishaupt[b] and Harald Aschemann[c]
Chair of Mechatronics, University of Rostock, Rostock, Germany

Keywords: Machine Learning in Control Applications.

Abstract: In this paper, we present an extension to the recursive Gaussian Process (RGP) regression that enables the satisfaction of inequality constraints and is well suited for a real-time execution in control applications. The soft inequality constraints are integrated by introducing an additional extended Kalman Filter (EKF) update step using pseudo-measurements. The sequential formulation of the algorithm and several developed heuristics ensure both the performance and a low computational effort of the algorithm. A special focus lies on an efficient consideration of monotonicity assumptions for GPs in the form of inequality constraints. The algorithm is statistically validated in simulations, where the possible advantages in comparison with the standard RGP algorithm become obvious. The paper is concluded with a successful experimental validation of the developed algorithm for the monotonicity-preserving learning of heat transfer values for the control of a vapor compression cycle evaporator, leveraging a previously published partial input output linearization (IOL).

1 INTRODUCTION

A lot of progress can be noticed regarding the online identification of system models. An example for a basic method is given by the definition of parametric functions like polynomial ansatz functions and the subsequent use of a recursive least-squares regression as described in (Blum, 1957). If models with non-measurable states or parameters are to be identified, this method can be extended to linear Kalman Filters (KF) or Unscented/Extended Kalman Filters (UKF/EKF) as proposed in (Kalman, 1960), (Julier and Uhlmann, 1997). In the presence of additional inequality constraints, especially Moving Horizon Estimation (MHE) techniques are suitable (Haseltine and Rawlings, 2005). If a more general approach is envisaged, there exist online-capable training methods of neural networks as discussed in (Jain et al., 2014).

Gaussian Processes (GP) have been established as a popular non-parametric alternative to neural networks (NNs). They are usually more data-efficient than neural networks, robust to overfitting and – as main advantage in comparison to NNs – provide an uncertainty quantification for the predicted value, see (Schürch et al., 2020). The non-parametric character and the $O(n^3)$ rise of computational load in dependency of the utilized data points, however, poses a big challenge for their online implementation. Nevertheless, the literature offers suitable methods to address this problem, like active-set methods as described in (Quiñonero-Candela and Rasmussen, 2005), which limit the number of utilized measurement points. Furthermore, a promising algorithm was presented by (Huber, 2013) with the recursive Gaussian Process regression (RGP). The idea here is to define the GPs as parametric functions based on user-defined basis vectors. This preserves many benefits of the GP regression while maintaining a small computational load.

For many modeling tasks, a certain previous knowledge is available. This may include bounds on model outputs or monotonicity assumptions. This knowledge might come from physical properties or in the form of stability-conserving constraints in a control setting. Nevertheless, the application of this knowledge during learning has the potential to provide superior models with far less data. In neural networks, such assumptions might be considered by an altering of the reward function as in (Desai et al., 2021). For standard GPs, a number of methods are available for considering assumptions, e.g. regarding the system structure as presented in (Beckers et al., 2022) or for inequality constraints as shown in (Veiga

[a] https://orcid.org/0009-0006-0480-8877
[b] https://orcid.org/0009-0007-0601-4605
[c] https://orcid.org/0000-0001-7789-5699

Husmann, R., Weishaupt, S. and Aschemann, H.
Recursive Gaussian Process Regression with Integrated Monotonicity Assumptions for Control Applications.
DOI: 10.5220/0013783100003982

In *Proceedings of the 22nd International Conference on Informatics in Control, Automation and Robotics (ICINCO 2025) - Volume 1*, pages 340-349
ISBN: 978-989-758-770-2; ISSN: 2184-2809

and Marrel, 2020). To the best of our knowledge, however, the integration of inequality constraints for recursive GPs represents a new development.

In (Husmann et al., 2024b), a partial IOL for Vapor Compression Cycle (VCC) Control including an integral feedback regarding the heat transfer value was presented. As mentioned in that paper, the usage of the steady state of the integrator to derive a data-based model seems a promising idea. Since monotonicity assumptions hold for the input dependencies of such a heat transfer value model, this application represents an interesting candidate for the validation of the RGP algorithm with the inclusion of monotonicity assumptions.

The main contributions of the paper are:

- Computationally efficient consideration of inequality constraints w.r.t. Gaussian variables in a sequential EKF update
- Integrating monotonicity assumptions into RGP regression by means of inequality constraints
- Real-time implementation of the presented online-learning algorithm regarding a data driven RGP model for the heat-transfer value of a VCC

The paper is structured as follows: First, we recapitulate the RGP in Sec. 2. Then, the generalized approach for considering inequality constraints is presented in Subsec. 3.1. Afterwards, we introduce the special case of formulating GP monotonicity as such a constraint in Subsec. 3.4, summarize and discuss the complete algorithm. As a first validation in Sec. 4, we statistically analyze the validity of the algorithm for a simulation example. Finally, an experimental validation is provided in Sec. 5 with an implementation of the real-time algorithm on a VCC test rig. The paper finishes with a conclusion and an outlook.

2 RECURSIVE GAUSSIAN PROCESS REGRESSION

In this chapter, we briefly describe recursive Gaussian Process regression (RGP) as presented in (Huber, 2013) and extended in (Huber, 2014). As usual, we utilize a Squared Exponential (SE) kernel

$$k(\boldsymbol{x},\boldsymbol{x}') = \sigma_K^2 \cdot \exp(-(\boldsymbol{x}-\boldsymbol{x}')^T(\boldsymbol{x}-\boldsymbol{x}')(2L)^{-1}) \quad (1)$$

with a zero mean function $m(\boldsymbol{X}) = 0$. In our application, the hyperparameters L and σ_K are user-defined. As elaborated in Subsec. 2.1, a joint length scale L is defined for all input dimensions, and the possibly different input ranges are addressed by an extra normalization step. For each GP, we consider only one measurement y_k per time step k, with a corresponding measurement variance σ_y^2. As detailed in Subsec. 2.1, $\boldsymbol{X}$ refers to the constant basis vectors, which are defined during initialization, and $\boldsymbol{X}_k$ denotes the current test input. The mean values $\boldsymbol{\mu}_k^g$ and the covariance matrix $\boldsymbol{C}_k^g$ of the kernels, which are updated recursively, give the RGP algorithm a KF-like structure.

The following variables can be precalculated offline:

$$\text{Offline} \quad \begin{aligned} &\boldsymbol{K} = k(\boldsymbol{X},\boldsymbol{X}), \\ &\boldsymbol{K}_I = \boldsymbol{K}^{-1}, \\ &\boldsymbol{\mu}_0^g = \boldsymbol{0}\,, \\ &\boldsymbol{C}_0^g = \boldsymbol{K}\,. \end{aligned} \quad (2)$$

Given a zero mean function and a single measurement per time step, the inference step simplifies to:

$$\text{Inference} \quad \begin{aligned} &\boldsymbol{J}_k = k(\boldsymbol{X}_k,\boldsymbol{X}) \cdot \boldsymbol{K}_I\,, \\ &\mu_k^p = \boldsymbol{J}_k \cdot \boldsymbol{\mu}_k^g, \\ &C_k^p = \cancel{k(\boldsymbol{X}_k,\boldsymbol{X}_k)}^{\sigma_K^2} + \boldsymbol{J}_k(\boldsymbol{C}_k^g - \boldsymbol{K})\boldsymbol{J}_k^T\,, \end{aligned} \quad (3)$$

where the superscript p indicates the GP prediction for the test inputs $\boldsymbol{X}_k$. This prediction is used in the following update step:

$$\text{Update} \quad \begin{aligned} &\boldsymbol{G}_k = \boldsymbol{C}_k^g \boldsymbol{J}_k^T \cdot (C_k^p + \sigma_y^2)^{-1}\,, \\ &\boldsymbol{\mu}_{k+1}^g = \boldsymbol{\mu}_k^g + \boldsymbol{G}_k \cdot (y_k - \mu_k^p), \\ &\boldsymbol{C}_{k+1}^g = \boldsymbol{C}_k^g - \boldsymbol{G}_k \boldsymbol{J}_k \boldsymbol{C}_k^g\,. \end{aligned} \quad (4)$$

2.1 Input Normalization

We define the basis vectors for all input axes as an equidistant grid with step size 1. This leads to a $\left(\prod_{i=1}^{n_X} N_i\right) \times n_X$-dimensional matrix which contains all vertices of the grid, where n_X denotes the input dimension of the GP, and N_i the number of points in the respective dimension. An example for the basis vectors for $n_X = 2$, $N_1 = 2$ and $N_2 = 3$ is

$$\boldsymbol{X} = \begin{bmatrix} \boldsymbol{X}_1^T \\ \boldsymbol{X}_2^T \end{bmatrix}^T = \begin{bmatrix} 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 2 & 2 \end{bmatrix}^T . \quad (5)$$

To consider the ranges of the actual inputs ζ_i in each dimension, we introduce the normalization step

$$X_{i,k} = f_{norm}(\zeta_{k,i}) = (\zeta_{i,k} - \underline{\zeta}_i) \cdot \underbrace{\frac{N_i - 1}{\overline{\zeta}_i - \underline{\zeta}_i}}_{\beta_i}, \quad (6)$$

which is applied before each GP evaluation. Here $\underline{\zeta}_i$ and $\overline{\zeta}_i$ denote the corresponding lower and upper bounds of the input, and β_i is a constant factor, which is used in Subsec. 3.4.

The normalization and the use of a joint length L was originally introduced to handle numerical issues

that may occur for large L in the standard inversion-based RGP formulation applied in this paper. With the normalization step, a universal maximum L_{max} – independent of the system – can be determined to maintain numerical stability. Please note that an alternative decomposition-based online solution is available, which is described in (Husmann et al., 2025). An additional benefit, independent of the RGP formulation is given, however, by the reduction of free hyperparameters. Consequently, it has also been used in (Husmann et al., 2025).

3 MONOTONICITY CONSTRAINT INTEGRATION

In this chapter, we present our implementation to enforce (soft) monotonicity constraints for RGPs. The proposed method is based upon an EKF update for inequality constraints, which is described in the sequel after the precise problem formulation. Since the real-time capability is required, we put emphasis on a computational speedup of the algorithm by usage of reformulations and heuristics in the next subsections. Afterwards, we present the formulation of GP monotonicity as a constraint, summarize and discuss the complete algorithm.

We consider a hidden function $y_k = z_k(\boldsymbol{\zeta}_k)$ that is dependent on deterministic inputs $\boldsymbol{\zeta}_k$. The output y_k of the hidden function z_k is measurable, with zero-mean Gaussian measurement noise with a variance of σ_y^2. We assume previous knowledge of the monotonicity of z_k w.r.t. its inputs, which can be stated in an inequality constraint regarding the partial derivatives $\frac{\partial z_k}{\partial \zeta_i} \gtrless 0$. To enable safety margins, this is generalized to $\frac{\partial z_k}{\partial \zeta_i} \gtrless B_i$, where B_i is a constant characterizing the boundary of the constraint.

3.1 EKF Update for Inequality Constraints

The direct consideration of hard inequality constraints on Gaussian variables leads to truncated Gaussians, see (Tully et al., 2011). For univariate Gaussians, the resulting mean and covariance can be calculated efficiently. For multivariate Gaussians and inequality constraints that dependent on several Gaussian input variables, however, exact solutions usually necessitate numerical methods. Here, (Simon, 2006a) provides an overview. (Simon, 2006a) also discusses the use of equality constraints as exact pseudo-measurements within a KF update. This is related, however, to some numerical issues since exact measurements lead to rank-deficient updates in a KF. The alternative soft constraints, where the pseudo-measurement is considered with a small uncertainty, are not subject to this problem. In this paper, hence, we build upon this approach and extend it towards inequality constraints.

The basic idea is to implement inequality constraints as pseudo-measurements using a ReLU measurement function as well as an EKF-update. In the case of an inactive inequality constraint in the current step, the ReLU function in combination with the EKF "hides" the inequality constraint in the update, otherwise it is considered as an equality constraint. Here, some parallels to the active-set method for constrained optimization become obvious, see (Nocedal and Wright, 2006). These parallels are for example also drawn in (Gupta and Hauser, 2007). There however in combination with, projection and gain-limiting methods instead of pseudo-measurements. Of course, a truncated Gaussian may differ quite dramatically in shape from a Gaussian distribution. As a results, this linearization-based approach may cause large errors in the covariance. To rule out over-approximation errors by this effect, the overall covariance update by the EKF inequality constraint is discarded at the end, as described later. This measure contributes to the "softness" of the constraints.

To simplify the implementation, we standardize all inequalities j by means of the sign indicator variable s_j: $\hat{y}_{IC,1} < B_1 \equiv s_1 \cdot (\hat{y}_{IC,1} - B_1) < 0$ with $s_1 = 1$, $\hat{y}_{IC,2} > B_2 \equiv s_2 \cdot (\hat{y}_{IC,2} - B_2) < 0$ with $s_2 = -1$. This corresponds to linear inequalities of the type $s_j(h_{IC,j}^T \boldsymbol{x}_k - B_j) < 0$, where $\boldsymbol{x}_k$ denotes the state vector. Now, we introduce the nonlinear measurement function, which is evaluated with the mean values

$$\hat{y}_{IC,j} = \tilde{h}_{IC,j}(\boldsymbol{x}_k = \boldsymbol{\mu}_{k+1}^g) = \mathrm{ReLU}(s_j(h_{IC,j}^T \boldsymbol{\mu}_{k+1}^g - B_j)). \tag{7}$$

Due to the standardization of the inequalities, all the pseudo-measurements become $y_{IC,j} = 0$. The measurement functions $\tilde{\boldsymbol{h}}_{IC} = \left[\tilde{h}_{IC,1}, \tilde{h}_{IC,2}, ..\right]^T$ can be concatenated in a vector.

The partial derivative of the measurement, necessary for the EKF update, is given by

$$\hat{\boldsymbol{h}}_{IC,j,k}^T = \left(\frac{\partial \tilde{h}_{IC,j}(\boldsymbol{x}_k)}{\partial \boldsymbol{x}_k}\bigg|_{\boldsymbol{x}_k = \boldsymbol{\mu}_{k+1}^g} \right)^T = \begin{cases} s_j h_{IC,j}^T & \text{if} \quad s_j(h_{IC,j}^T \boldsymbol{\mu}_{k+1}^g - B_j) > 0 \\ [0,0,..] & \text{else,} \end{cases} \tag{8}$$

which can be concatenated as well to the linearized measurement matrix $\hat{\boldsymbol{H}}_{IC,k} = \left[\hat{\boldsymbol{h}}_{IC,1,k}, \hat{\boldsymbol{h}}_{IC,2,k}, ..\right]^T$. The update can then be computed according to a stan-

dard EKF, with the pseudo-measurements $y_{IC,j} = 0$

$$\begin{aligned}\tilde{\boldsymbol{G}}_k &= \boldsymbol{C}^g_{k+1}\hat{\boldsymbol{H}}^T_{IC,k}(\hat{\boldsymbol{H}}_{IC,k}\boldsymbol{C}^g_{k+1}\hat{\boldsymbol{H}}^T_{IC,k}+\boldsymbol{R}_{IC})^{-1}\,, \quad (9)\\ \boldsymbol{\mu}^c_{k+1} &= \boldsymbol{\mu}^g_{k+1} - \tilde{\boldsymbol{G}}_k\cdot\tilde{\boldsymbol{h}}_{IC}(\boldsymbol{\mu}^g_{k+1})\,,\\ \boldsymbol{C}^c_{k+1} &= \boldsymbol{C}^g_{k+1} - \tilde{\boldsymbol{G}}_k\hat{\boldsymbol{H}}_{IC,k}\boldsymbol{C}^g_{k+1},\end{aligned}$$

where $\boldsymbol{R}_{IC}$ is the small diagonal pseudo-measurement noise matrix, with all $\boldsymbol{R}_{IC}(j,j) = r_{IC}$. The superscript c denotes the constrained mean values and covariance.

3.2 Reduction of the Computational Load

Since we strive for real-time applicability of the algorithm, computational efficiency is crucial. Thus, instead of one batchwise EKF update, we perform n_{IC} sequential EKF updates for each pseudo-measurement. The sequential KF is theoretically identical to the batchwise one – as long as the (pseudo-)measurements are uncorrelated, see (Simon, 2006b). This assumption is met in our case because $\boldsymbol{R}_{IC}$ is diagonal. For the EKF, additional conditions have to be taken into account as discussed in Subsec. 3.7. Please note that numerical differences between the batchwise and sequential KF update may occur for large dimensions n_{IC}. Consequently, the exact reformulation of the batchwise EKF update to a sequential EKF update can be stated as follows:

Set $\boldsymbol{C}^c_{k+1,1} = \boldsymbol{C}^p_{k+1}$ and $\boldsymbol{\mu}^c_{k+1,1} = \boldsymbol{\mu}^g_{k+1}$.
For $j = 1,\ldots,n_{IC}$:

$$\begin{aligned}\tilde{\boldsymbol{G}}_{k,j} &= \boldsymbol{C}^c_{k+1,j}\hat{\boldsymbol{h}}^T_{IC,k,j}(\hat{\boldsymbol{h}}_{IC,k,j}\boldsymbol{C}^c_{k+1,j}\hat{\boldsymbol{h}}^T_{IC,k,j}+r_{IC})^{-1}\,,\\ \boldsymbol{\mu}^c_{k+1,j+1} &= \boldsymbol{\mu}^c_{k+1,j} - \tilde{\boldsymbol{G}}_{k,j}\tilde{h}_{IC,j}(\boldsymbol{\mu}^g_{k+1})\,, \quad (10)\\ \boldsymbol{C}^c_{k+1,j+1} &= \boldsymbol{C}^c_{k+1,j} - \tilde{\boldsymbol{G}}_{k,j}\hat{\boldsymbol{h}}_{IC,k,j}\boldsymbol{C}^c_{k+1,j}\,.\end{aligned}$$

As discussed in the previous subsection, the rows $\hat{\boldsymbol{h}}_{IC,k,j}$ of $\hat{\boldsymbol{H}}_{IC,k}$ are zero if the respective constraint j is not active. Since an EKF update with a zero measurement vector has no effect, another exact reformulation can be applied, which leads to a dramatic speedup.

Set $\boldsymbol{C}^c_{k+1,1} = \boldsymbol{C}^p_{k+1}$ and $\boldsymbol{\mu}^c_{k+1,1} = \boldsymbol{\mu}^g_{k+1}$.
Evaluate $\boldsymbol{t}_k = \boldsymbol{H}_{IC}\boldsymbol{\mu}^g_{k+1}$ with $\boldsymbol{H}_{IC} = [\boldsymbol{h}_{IC,1}, \boldsymbol{h}_{IC,2}, ..]^T$.
For $j = 1,\ldots,n_{IC}$:
If $s_j(\boldsymbol{t}_k(j) - B_j) > 0$:

$$\begin{aligned}\tilde{\boldsymbol{G}}_{k,j} &= \boldsymbol{C}^c_{k+1,j}\boldsymbol{h}_{IC,j}(\boldsymbol{h}^T_{IC,j}\boldsymbol{C}^c_{k+1,j}\boldsymbol{h}_{IC,j}+r_{IC})^{-1}\,,\\ \boldsymbol{\mu}^c_{k+1,j+1} &= \boldsymbol{\mu}^c_{k+1,j} - \tilde{\boldsymbol{G}}_{k,j}s_j(\boldsymbol{t}_k(j) - B_j)\,, \quad (11)\\ \boldsymbol{C}^c_{k+1,j+1} &= \boldsymbol{C}^c_{k+1,j} - \tilde{\boldsymbol{G}}_{k,j}\boldsymbol{h}^T_{IC,j}\boldsymbol{C}^c_{k+1,j}\,.\end{aligned}$$

Here, we also used the property $s_j^2 = 1$.

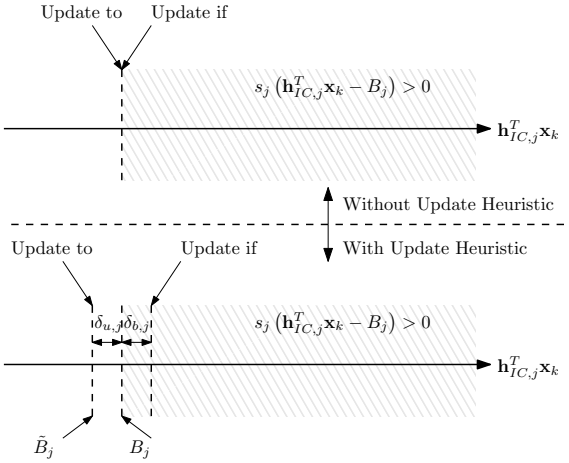

Figure 1: Schematic illustration of the update heuristic.

3.3 Upper-Bounding the Computational Load

Since the algorithm is utilized for model learning, we can expect that our measurements, in majority, do not violate our constraints, but the constraint integration rather improves the speed at which learning takes place. To further reduce the maximum computational load per time step, we upper-bound the sequential pseudo-measurement updates per time step to $\tilde{n}_{IC} \leq n_{IC}$. Practice has shown that in cases where the actual hidden function is close to the bounds, noise and numerical errors may lead to repetitive updates w.r.t. certain constraints. In combination with the upper-bounding of the pseudo-measurement updates per time step, this may lead to cases where certain constraints are not considered altogether. Thus, we introduce an additional heuristic to enable a hysteresis-like behavior: As illustrated in Fig. 1, we only update if $s_j(\boldsymbol{t}_k(j) - B_j) > \delta_{b,j}$. Moreover, we update to the value $\tilde{B}_j = B_j - \delta_{u,j}s_j$, where $\delta_{b,j} \geq 0$ and $\delta_{u,j} \geq 0$ are both small non-negative constants.

3.4 GP Monotonicity Assumptions as Constraints

Monotonicity assumptions on a real, continuously differentiable function $z_k(\boldsymbol{\zeta})$ can be stated as inequality constraints of its partial derivatives w.r.t. its inputs, e.g. $\frac{\partial z_k}{\partial \zeta_i} \gtrless 0$. According to the derivation in (McHutchon, 2015), the mean values of the partial derivative of a GP with SE kernels in the test grid $\boldsymbol{X}_k$

w.r.t. the input $\zeta_{i,k}$ is given by

$$\frac{\partial \boldsymbol{\mu}_k^p}{\partial \zeta_{i,k}} = \underbrace{\left(-\frac{\beta_i}{L}\left(\left(\boldsymbol{X}_{i,k} - \boldsymbol{X}_i^T\right) \odot k(\boldsymbol{X}_k, \boldsymbol{X})\right) \cdot \boldsymbol{K}_I\right)}_{\boldsymbol{H}_{IC,l,k}} \cdot \boldsymbol{\mu}_k^g, \tag{12}$$

where $\odot$ denotes the Schur- or Hadamard product. This function is linear in the mean values of the RGP. For a constant test grid $\tilde{\boldsymbol{X}}$, the linear gain matrix $\boldsymbol{H}_{IC,l,k}$ is also time-invariant. For simplicity, we only consider one monotonicity assumption per dimension, where $l = 1, \ldots, n_l$ with $n_l \leq n_\zeta$ indicate the input dimensions for which monotonicity assumptions exist. Consequently, each dimension is characterized by only one set of values s_l, B_l, $\delta_{B,l}$ and $\delta_{U,l}$. As a constant test grid is considered for the monotonicity, we can state the mean values of the partial derivatives at the test vector grid using linear pseudo-measurement matrices. They can be calculated by means of

$$\frac{\partial \boldsymbol{\mu}_k^p}{\partial \zeta_l} = \underbrace{\left(-\frac{\beta_l}{L}\left(\left(\tilde{\boldsymbol{X}}_l - \boldsymbol{X}_l^T\right) \odot k(\tilde{\boldsymbol{X}}, \boldsymbol{X})\right) \cdot \boldsymbol{K}_I\right)}_{\boldsymbol{H}_{IC,l}} \cdot \boldsymbol{\mu}_k^g \tag{13}$$

for all dimensions l that comply with monotonicity assumptions. Here, the matrices $\boldsymbol{H}_{IC,l}$ can be calculated offline.

As confirmed in a comparison with the exact covariance of the partial derivative, see (McHutchon, 2015), the usage of these measurement functions for the covariance prediction of the partial derivative leads to further linearization errors. This error vanishes if $\tilde{\boldsymbol{X}} = \boldsymbol{X}$ holds. In the following, we therefore choose the monotonicity test vector grid equal to the basis vector grid – which is reasonable for most applications anyway.

3.5 Inequality Constraints on the GP Output

Even though it is not employed in this paper, the consideration of inequality constraints on GP outputs is included here for completeness. Like the monotonicity constraints, output constraints are evaluated on a grid. If the chosen test grid is also identical to the basis vector grid, the corresponding output equation becomes $\boldsymbol{\mu}_{k+1}^p = \boldsymbol{J}_k(\boldsymbol{X})\boldsymbol{\mu}_{k+1}^g$. Since $\boldsymbol{J}_k(\boldsymbol{X}) = k(\boldsymbol{X}, \boldsymbol{X})\boldsymbol{K}_I = \boldsymbol{I}$ holds, the respective pseudo-measurement matrix becomes the identity matrix $\boldsymbol{H}_{IC} = \boldsymbol{I}$ and leads to the simplification $\boldsymbol{t}_k = \boldsymbol{\mu}_{k+1}^g$. Please note that for soft GP output constraints, other measures, like a bounding of the measurements y_k or the RGP mean values $\boldsymbol{\mu}_k^g$, might be more efficient and effective.

3.6 Summary of GP Monotonicity Constraints

This section summarizes the pseudo-measurement updates for constraints related to monotonicity assumptions in the RGP. The extended RGP algorithm will in the following referred to as RGPm. Please note that the index k denoting the time step is omitted for simplicity. In addition to the offline precomputations (2) for the standard RGP in Sec. 2, the following measurement matrices need to be calculated for each input dimension with a monotonicity assumption:

For $l = 1, \ldots, n_l$:

$$\boldsymbol{H}_{IC,l} = \left(-\tfrac{\beta_l}{L}\left(\left(\boldsymbol{X}_l - \boldsymbol{X}_l^T\right) \odot \boldsymbol{K}\right) \cdot \boldsymbol{K}_I\right). \tag{14}$$

Also the boundaries $\tilde{B}_l = B_l - \delta_{u,l} s_l$ and $\delta_{b,l}$ for the update heuristic as well as the upper limits per dimension $\tilde{n}_{IC,l}$ have to be defined.

For each time step k, after the evaluation of the RGP inference (3) and update steps (4) from Sec. 2, the pseudo-measurement update is computed as follows:

Set $\boldsymbol{C}_{1,1}^c = \boldsymbol{C}_{k+1}^g$ and $\boldsymbol{\mu}_{1,1}^c = \boldsymbol{\mu}_{k+1}^g$.
For $l = 1, \ldots, n_l$:
  Evaluate $\boldsymbol{t}_l = \boldsymbol{H}_{IC,l}\boldsymbol{C}_{l,1}^c$, with $\boldsymbol{H}_{IC,l} = [\boldsymbol{h}_{IC,l,1}, \boldsymbol{h}_{IC,l,2}, ..]^T$ and set counter $m = 1$.
  For $j = 1, \ldots, n_{IC,l}$:
    If $s_l(\boldsymbol{t}_l(j) - \tilde{B}_l) > \delta_{b,l}$ and $m \leq \tilde{n}_{IC,l}$: , (15)
      $\tilde{\boldsymbol{G}}_{l,j} = \boldsymbol{C}_{l,j}^c \boldsymbol{h}_{IC,l,j} (\boldsymbol{h}_{IC,l,j}^T \boldsymbol{C}_{l,j}^c \boldsymbol{h}_{IC,l,j} + r_{IC})^{-1}$
      $\boldsymbol{\mu}_{l,j+1}^c = \boldsymbol{\mu}_{l,j}^c - \tilde{\boldsymbol{G}}_{l,j} s_l (\boldsymbol{h}_{IC,l,j} \boldsymbol{\mu}_{l,1}^c - \tilde{B}_l)$
      $\boldsymbol{C}_{l,j+1}^c = \boldsymbol{C}_{l,j}^c - \tilde{\boldsymbol{G}}_{l,j} \boldsymbol{h}_{IC,l,j} \boldsymbol{C}_{l,j}^c$
      $m = m + 1$
  $\boldsymbol{\mu}_{l+1,1}^g = \boldsymbol{\mu}_{l,\tilde{n}_{IC,l}}^c$ and $\boldsymbol{C}_{l+1,1}^c = \boldsymbol{C}_{l,\tilde{n}_{IC,l}}^c$
Set $\boldsymbol{\mu}_{k+1}^g = \boldsymbol{\mu}_{n_l,\tilde{n}_{IC,l}}^c$, but reset $\boldsymbol{C}_{k+1}^g = \boldsymbol{C}_{1,1}^c$.

3.7 Discussion

In general, a consideration of monotonicity constraints with a test vector grid unequal to the basis vector grid is possible and was successfully implemented. It tends to be, however, numerically unstable, most likely due to linearization errors w.r.t. the covariance prediction.

As pointed out, the implemented monotonicity constraints are only soft constraints. This means in practice that the GP will adhere to the measurements if these consistently contradict the monotonicity assumptions. This scenario would be caused by wrong

assumptions regarding either the monotonicity itself or the measurement quality. Although still not guaranteeing constraint satisfaction, a safety margin can be added by changing B_l accordingly. For safety-critical constraints, an additional evaluation on a finer grid as well as a subsequent output correction may be useful.

A UKF may be more suited for handling the nonlinearities in the presented pseudo-measurement update. Nevertheless, it would prevent most of the reformulations which facilitate the envisaged real-time implementation. Consequently, it was not considered further.

Please note that piece-wise monotonicity can also be addressed as long as the regions are definable through sets of basis vector points. This variation and the direct output constraints in Subsec. 3.5 have already been successfully validated but are not described for the sake of brevity.

Please note that the sequential update of the EKF involves an additional condition for the equivalence to a batchwise update. Here, the linearization point has to be identical. This condition is fulfilled within each input dimension by introducing the intermediary variables $\boldsymbol{t}_k$ and $\boldsymbol{t}_l$ in Subsecs. 3.2 and 3.4, respectively. It is, however, not fulfilled over all input dimensions l in Subsec. 3.4. Thus, even without the bounded number of updates and the update heuristic, a slight deviation is present between the sequential EKF and the batchwise EKF if several monotonicity assumptions apply.

4 VALIDATION BASED ON SIMULATIONS

As a first validation example, the hidden static function $z_k = 2(1+0.1\zeta_k+\zeta_k^3)$ is learned. The input ζ_k is picked from a random uniform distribution over the complete input range $\zeta_k \in [-1,1]$. Zero-mean Gaussian white noise with a variance of $\sigma_y^2 = 1e-2$ is added to the measured output y_k.

The RGP hyperparameters remain constant and are chosen as follows: $L = 3$, $\sigma_K = 1e1$, and $N = 21$. Obviously, the hidden function is strictly monotonically increasing, so $\frac{\partial z}{\partial \zeta} > 0$, $B_1 = 0$, and $s_1 = -1$ hold. The pseudo-measurement noise is chosen as $r_{IC} = 1e-8$.

4.1 Simulation Results

In Fig. 2, the hidden function z and its learned reconstructions z_{RGPm} and z_{RGP} – with and without monotonicity assumptions – are depicted after five noisy

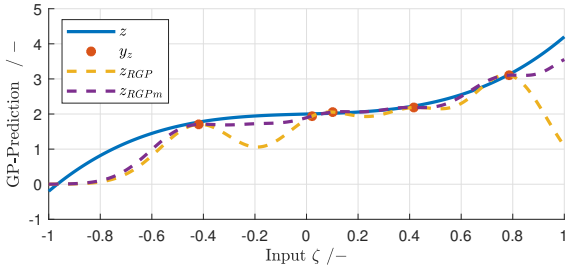

Figure 2: RGP and RGPm outputs in comparison with the hidden function z after 5 time steps.

measurements y_k. It becomes obvious that the RGPm algorithm works properly. In this example, it is capable of enforcing strict monotonicity over the whole input range, while the curve is not significantly altered in the vicinity of the actual measurement points. The result represents a clearly better fit in comparison with the classical RGP – especially in regions where still no measurements are available.

For a statistical validation of the algorithm and an assessment of the impact of the adaptations, we investigate six different variants of the algorithm:

S0 Pure RGP

S1 unlimited $\tilde{n}_{IC} = n_{IC}$ RGPm with $\delta_b = 0$ and $\delta_u = 0$

S2 $\tilde{n}_{IC} = 2$ limited RGPm with $\delta_b = 0$ and $\delta_u = 0$

S3 $\tilde{n}_{IC} = 2$ limited RGPm with boundary heuristic $\delta_b = 1e-1$ and $\delta_u = 1e-1$

S4 $\tilde{n}_{IC} = 5$ limited RGPm with $\delta_b = 0$ and $\delta_u = 0$

S5 $\tilde{n}_{IC} = 5$ limited RGPm with boundary heuristic $\delta_b = 1e-1$ and $\delta_u = 1e-1$

For each scenario, 500 simulation runs are conducted with the described uniform random input and the noisy output. After $k = 1,2,5,\ldots,1000$ steps, the root mean-squares error (RMSE) between the learned function of each variation, compared to the hidden function over the complete input range, is calculated. This RMSE is again averaged over all 500 simulations and depicted in Fig. 3. Fig. 4 shows the average cumulative number of pseudo-measurement updates (CPMU) that were conducted for the alternatives S1,...,S5.

All RGPm variants perform better than the classical RGP. This advantage vanishes after a certain number of steps, and all the RMSE converge to the same value. This could be expected because the monotonicity assumptions are valid and should be represented on average in the measurements.

It can be seen that a tighter limit on the number of pseudo-measurement updates per time step is generally a trade-off w.r.t. the performance. Accordingly, S1 is better than S4 and S2. In the current application, the improvement from S4 to S1 is however quite

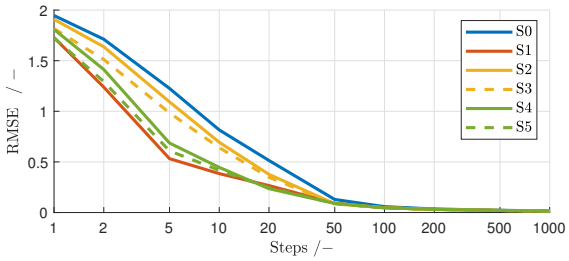

Figure 3: Average RMSE for 500 simulation runs of the different RGP and RGPm variants.

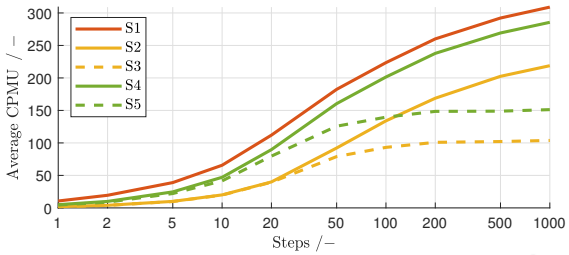

Figure 4: Average cumulative pseudo-measurement updates (CPMU) for 500 simulation runs of the different RGPm variants.

small, and reverses slightly around 20 steps. This last effect may stem from numerical errors that could be related to the higher number of pseudo-measurement updates.

The heuristic approach of including outer/inner boundaries resulted in significant improvements in the RMSE. The reduced repeated pseudo-measurement updates w.r.t. certain constraints are clearly visible in Fig. 4. While S1, S2 and S4 still perform pseudo-measurement updates after 1000 steps, the update heuristic leads to a converging CPMU for S3 and S5, which means that no more pseudo-measurement updates are performed after a certain point. Whereas the real-time capability is only influenced by the maximum number of updates per time step, the reduced computational load can be seen as an additional benefit.

5 EXPERIMENTAL RESULTS FOR A VAPOR COMPRESSION CYCLE

In this chapter, the functionality of the proposed method is investigated in a real-time application – a counterflow evaporator as a subsystem of a VCC. As discussed in detail in (Husmann et al., 2024a) and (Husmann et al., 2024b), heat transfer values, especially in the presence of phase changes, depend on process parameters and are hard to determine. In (Husmann et al., 2024b), integral feedback was applied directly affecting the heat transfer value and combined with a partial IOL. In addition to a stability proof also the convergence of integrator state towards the actual heat transfer value was proven. In this paper we want to leverage the integrator steady state to learn a RGP model for the heat transfer value that depends on several process parameters. The aim is to use the RGP model in parallel to the integrator feedback to further improve the transient control behavior.

The algorithm was implemented on a test rig at the Chair of Mechatronics at the University of Rostock. The utilized hardware, a Bachmann PLC (CPU:MH230), runs at a sampling time of 10ms. The test rig is fully equipped with sensors, an electronic expansion valve as well as a compressor with a variable frequency drive. Please refer to (Husmann et al., 2024a) and (Husmann et al., 2024b) for further details about the test rig.

5.1 Partial IOL Control

As the partial IOL is described in detail in (Husmann et al., 2024b), we only want to give a brief overview in this paper. The envisioned control structure is depicted in Fig. 5. As shown, the control inputs are given by the expansion valve opening u_E and the compressor speed u_C. The control outputs are chosen as the evaporator mean temperature $y_1 = T_{Evap}$ and the superheating enthalpy $y_2 = \Delta h_{SH}$ after the evaporator. The control-oriented evaporator model, first derived in (Husmann et al., 2023), is governed by the following two ODEs

$$\dot{h}_{Evap} = \frac{\dot{m}_E h_{Evap,in} - \dot{m}_C h_{Evap,out} + \dot{Q}_{Evap} - (\dot{m}_E - \dot{m}_C) h_{Evap}}{V_{Evap} \rho_{Evap}} \tag{16}$$

and

$$\dot{\rho}_{Evap} = \frac{1}{V_{Evap}} (\dot{m}_E - \dot{m}_C) \ . \tag{17}$$

Here, the first state equation is affected by an uncertain convective heat flow

$$\dot{Q}_{Evap} = \alpha \cdot \alpha_{norm} \cdot A_{Evap} \cdot (T_U - T_{Evap}) \tag{18}$$

with α_{norm} as a normalization factor. The input-dependent mass flows are given by

$$\dot{m}_E = \zeta_E \cdot \sqrt{\rho_{Cond,out} (p_{Cond} - p_{Evap})} \cdot u_E \tag{19}$$

and

$$\dot{m}_C = \hat{m}_{Rev} u_C \eta_V \ . \tag{20}$$

Both outputs can be expressed in terms of the state variables. The second output is given by

$$y_{lin,2} = \Delta h_{SH} = \frac{h_{Evap} - (1 - w_\rho) h_{Evap,in}}{w_\rho} - h_{Evap,sat}(T_{Evap}) \, , \tag{21}$$

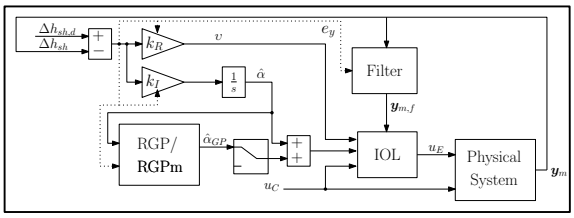

Figure 5: Nonlinear tracking control structure.

and its first time derivative becomes

$$\dot{y}_{lin,2} = \frac{1}{w_\rho}\dot{h}_{Evap} - \frac{1-w_\rho}{w_\rho}\frac{dh_{Evap,in}}{dt} - \left.\frac{\partial h_{sat}}{\partial T}\right|_{T_{Evap}} \dot{T}_{Evap}\,, \tag{22}$$

which lead to a relative degree of one. Whereas the inverse dynamics is determined for both control inputs in the standard IOL proposed in (Husmann et al., 2023), the partial IOL used here leverages the inverse dynamics of the control input u_E

$$u_E = b_3(.)^{-1}\left[\upsilon_2 - a_2(.) - b_4(.)u_C\right], \tag{23}$$

and considers the second input u_C as a measurable disturbance. Please note that a secondary controller needs to be implemented for the first control output $y_1 = T_{Evap}$, see (Husmann et al., 2024b). In this paper, however, the control input u_C is assumed as given because it has no relevance for the presented algorithm. The partial IOL is completed with a stabilizing feedback $\upsilon_2 = k_R(e_y)e_y$, with $e_y = y_{2,d} - y_2$. Here the nonlinear feedback from (Husmann et al., 2024b) is utilized. The introduction of a parameter update law for the heat transfer value $\dot{\hat{\alpha}} = -k_I(y_{2,d} - y_2)$, which corresponds to integral control action w.r.t. $\hat{\alpha}$, dramatically increases the performance and guarantees steady-state accuracy as shown in (Husmann et al., 2024b). Since the integrator state $\hat{\alpha}$ was proven to converge to an exact representation of the lumped heat transfer value in the control-oriented model, it is suitable for the training of a model.

5.2 RGPm Implementation

As input variables for the GP model, the compressor speed u_C and the superheating enthalpy $y_2 = \Delta h_{SH}$ are chosen, i.e., $\hat{\alpha}_{GP} = z(u_C, \Delta h_{SH})$. As known from our own experiments and from the literature, the inequalities $\frac{\partial \alpha}{\partial u_C} > 0$ and $\frac{\partial \alpha}{\partial \Delta h_{SH}} < 0$ hold. Consequently, they are included as monotonicity assumptions in the RGPm algorithm. The evaluation of the GP model is conducted with the reference value $y_{d,2}$ instead of the current one, i.e., $\hat{\alpha}_{GP} = z(u_C, \Delta h_{SH,d})$. Thereby, any additional feedback is avoided, and the GP model obtains a pure feedforward structure w.r.t. its evaluation. This prevents stability issues.

The RGP and RGPm algorithms are implemented with basis vector dimensions of $N_1 = 5$ and $N_2 = 5$ for the respective GP inputs, which results in $N_1 N_2 = 25$ basis vector points. The monotonicity update is bounded to $n_1 = n_2 = 5$ steps per dimension and per time step. The parameters for the update heuristics are defined as follows: $\delta_{u,1} = \delta_{u,2} = 0$ and $\delta_{b,1} = \delta_{b,2} = 2e-6$. The pseudo-measurement noise is chosen as $r_1 = r_2 = 1e-8$, and the length scale is set to $L = 1.5$. The signing variables for the respective monotonicity assumptions are given by $s_1 = -1$ and $s_2 = 1$ and the safety margins are disabled, so $B_1 = B_2 = 0$.

To ensure that the integrator state is only utilized in the vicinity of it's equilibrium state, we employ a heuristic which only enables the RGP update if $|e_y| < 1$ holds for 10 consecutive seconds. The numerical values in this heuristic mark a trade-off between learning speed and accuracy. With a larger error margin or a shorter time horizon, more data qualifies for training. However, the considered data might also relate to system states further away from the equilibrium.

5.3 Results

To illustrate the functionality and advantages of the RGPm algorithm, two consecutive experiments with the same step-wise u_C-trajectory are performed: the first one for $y_{2,d} = 14$, the other one for $y_{2,d} = 7$. The RGPm algorithm is trained online during both experiments – without applying its output value to the controller. For a comparison, the RGP algorithm is later trained offline on the same data. The results after the first and second experiment are depicted for both algorithms in Fig. 6. Since the monotonicity updates for the covariance matrix are discarded at the end of each time step, both algorithms lead to the same covariance, so that only one is depicted per experiment. In the upper right picture of Fig. 6, the training region during the experiment becomes obvious, because the covariance is small there. When comparing the mean values for the RGPm (upper left) and RGP (upper middle), it becomes evident that the learned model within the input region used for the training is nearly identical. Outside of this region, however, the shapes vary drastically. While the monotonicity assumptions in the RGPm model lead to a physically viable model shape, this cannot be stated for the standard RGP model. Here, the overall shape is only defined by the limited amount of data available in the non-trained region. As shown in the comparison of the trained models after the second experiment for both RGPm (lower left) and RGP (lower middle), the differences between the models decline with the availability of data. This reinforces the findings from the simulation study.

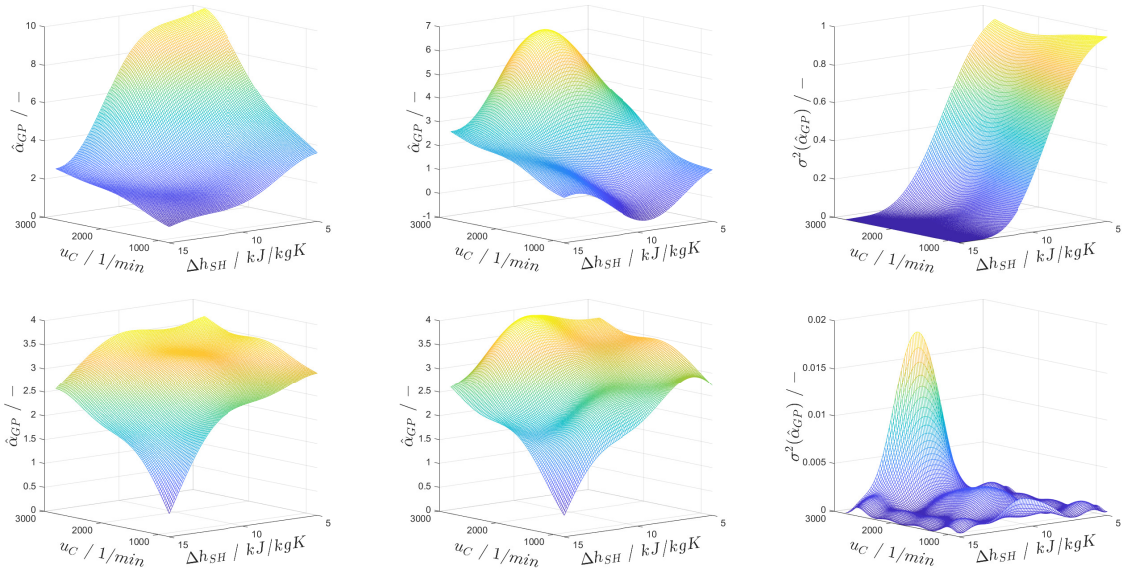

Figure 6: GP model predictions from left to right: RGPm (online), RGP (offline comparison for same data) and covariances of both variants. From top to bottom: after the first u_C-trajectory for $y_{2,d} = 14$, and after the second u_C-trajectory for $y_{2,d} = 7$.

The softness of the constraints in the RGPm algorithm becomes clear for low superheating and large compressor speeds. Here, the monotonicity constraints are slightly violated, however much less than in the standard RGP algorithm. The violation may be fixed by safety margins and a more involved heuristic for the RGP update.

The evaluation of the overall control performance with the trained model is outside of the scope of this paper, since a more detailed comparison would need additional concepts and algorithms, e.g. suitable training strategies. A quick validation of the RGPm model after the second experiment, depicted in Fig. 6 (lower left corner), can be performed by deactivating the integrator feedback control and running a u_C test trajectory together with a learned $\hat{\alpha}_{GP}$ and with a constant α. Here, for a reference value of $y_d = 13$, which wasn't used for the training, a 28 % reduction in RMSE could be achieved, which represents a promising first result for the application of the trained RGPm model.

6 CONCLUSIONS AND OUTLOOK

This paper presents an extension of the recursive Gaussian Process regression (RGP) algorithm to enforce (soft) constraints during an online training. Here, the consideration of monotonicity assumptions in the GP model is emphasized as well as their implementation as constraints. The algorithm is statistically validated in simulations. Moreover, different simplifications are introduced to enable a real-time use, and their effect on the performance is investigated. A real-time implementation on a test rig is presented afterwards, where the learning of heat transfer value models for an evaporator is considered. The evaporator is a main component of a vapor compression cycle operated by a previously published partial IOL. Here, the extension of a standard RGP regression by monotonicity constraints results in a significantly better modeling – especially if few data is available.

A combination of the extension presented in this paper with the Kalman Filter integration of the RGP (KF-dRGP), is straightforward and allows for an RGP training with constraints if the output of the hidden function is not directly measurable. For systems where a hard constraint satisfaction is crucial, further investigations are still necessary. As discussed earlier, an additional safety step with a calculation of the constraint satisfaction on a finer grid could be suitable. As mentioned in Sec. 5, the full integration of the GP model within the partial IOL for the VCC control necessitates further work. It is obviously desirable, for example, to simultaneously learn and apply the learned model. This does, however, introduce an additional feedback loop into the system, which may affect the stability. Here, the presented output and monotonicity bounds mark important tools. Given promising first results, steps in this direction will be topics of future work.

REFERENCES

Beckers, T., Seidman, J., Perdikaris, P., and Pappas, G. J. (2022). Gaussian process port-hamiltonian systems: Bayesian learning with physics prior. In *2022 IEEE 61st Conference on Decision and Control (CDC)*, pages 1447–1453.

Blum, M. (1957). Fixed memory least squares filters using recursion methods. *IRE Transactions on Information Theory*, 3(3):178–182.

Desai, S. A., Mattheakis, M., Sondak, D., Protopapas, P., and Roberts, S. J. (2021). Port-hamiltonian neural networks for learning explicit time-dependent dynamical systems. *Physics Review E*, 104:034312.

Gupta, N. and Hauser, R. (2007). Kalman filtering with equality and inequality state constraints.

Haseltine, E. L. and Rawlings, J. B. (2005). Critical evaluation of extended Kalman filtering and moving-horizon estimation. *Industrial & Engineering Chemistry Research*, 44(8):2451–2460.

Huber, M. F. (2013). Recursive Gaussian process regression. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 3362–3366.

Huber, M. F. (2014). Recursive Gaussian process: On-line regression and learning. *Pattern Recognition Letters*, 45:85–91.

Husmann, R., Weishaupt, S., and Aschemann, H. (2023). Nonlinear control of a vapor compression cycle by input-output linearisation. In *27th International Conference on Methods and Models in Automation and Robotics (MMAR)*, pages 193–198.

Husmann, R., Weishaupt, S., and Aschemann, H. (2024a). Control of a vapor compression cycle based on a moving-boundary model. In *2024 28th International Conference on System Theory, Control and Computing (ICSTCC)*, pages 438–444.

Husmann, R., Weishaupt, S., and Aschemann, H. (2024b). Nonlinear control of a vapor compression cycle based on a partial IOL. In *IECON 2024 - 50th Annual Conference of the IEEE Industrial Electronics Society*, pages 1–6.

Husmann, R., Weishaupt, S., and Aschemann, H. (2025). Direct integration of recursive Gaussian process regression into extended Kalman filters with application to vapor compression cycle control. In *13th IFAC Symposium on Nonlinear Control Systems NOLCOS 2025*.

Jain, L. C., Seera, M., Lim, C. P., and Balasubramaniam, P. (2014). A review of online learning in supervised neural networks. *Neural computing and applications*, 25:491–509.

Julier, S. J. and Uhlmann, J. K. (1997). A new extension of the Kalman filter to nonlinear systems. In *Proc. of AeroSense: The 11th Int. Symp. on Aerospace/Defence Sensing Simulation and Controls*.

Kalman, R. E. (1960). A new approach to linear filtering and prediction problems. *Transactions of the ASME–Journal of Basic Engineering*, 82(Series D):35–45.

McHutchon, A. J. (2015). *Nonlinear modelling and control using Gaussian processes*, chapter A2, pages 185–192.

Nocedal, J. and Wright, S. J. (2006). *Numerical Optimization*, chapter 16.5, pages 467–480. Springer New York, NY.

Quiñonero-Candela, J. and Rasmussen, C. E. (2005). A unifying view of sparse approximate Gaussian process regression. *Journal of Machine Learning Research*, 6:1939–1959.

Schürch, M., Azzimonti, D., Benavoli, A., and Zaffalon, M. (2020). Recursive estimation for sparse Gaussian process regression. *Automatica*, 120.

Simon, D. (2006a). *Optimal State Estimation*, chapter 7.5, pages 212–222. John Wiley & Sons, Ltd.

Simon, D. (2006b). *Optimal State Estimation*, chapter 6.1, pages 150–155. John Wiley & Sons, Ltd.

Tully, S., Kantor, G., and Choset, H. (2011). Inequality constrained Kalman filtering for the localization and registration of a surgical robot. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5147–5152.

Veiga, S. D. and Marrel, A. (2020). Gaussian process regression with linear inequality constraints. *Reliability Engineering & System Safety*, 195:106732.