Lehrstuhl für
Prozeßtechnik
Prof. Dr.-Ing. W. Marquardt
RWTH Aachen

Technical Report LPT–1999–14

Adaptive Multiscale Approach to Real–Time Dynamic Optimization

T. Binder[1], L. Blank[2], W. Dahmen[2], W. Marquardt[1]

October 1999

Internal Report, October 1999.

[1] Lehrstuhl für Prozeßtechnik
RWTH Aachen, D–52056 Aachen

[2] Institut für Geometrie und Praktische Mathematik
RWTH Aachen, D–52056 Aachen

Enquiries should be addressed to:

Lehrstuhl für Prozeßtechnik
RWTH Aachen
Templergraben 55
D–52056 Aachen

Tel.: ++49 / 241 / 80 46 68
Fax: ++49 / 241 / 88 88 326
E–Mail: secretary@lfpt.rwth–aachen.de

RWTH

An Adaptive Multiscale Approach to Real - Time Dynamic Optimization

*Thomas Binder[1], Luise Blank[2], Wolfgang Dahmen[2], Wolfgang Marquardt,[1†]
[1]Lehrstuhl für Prozeßtechnik,
[2]Institut für Geometrie und Praktische Mathematik,
RWTH Aachen University of Technology,
D-52056 Aachen, Germany

Abstract

In the present work we explore an adaptive discretization scheme for real–time dynamic optimization problems applied to dynamic data reconciliation. The proposed method is embedded into a solution methodology where the dynamic optimization problem is approximated by a hierarchy of successively refined finite dimensional problems. Information on the solution of the coarser approximations is used to construct a fully adaptive, problem dependent discretization. We will show, that adaptive discretization is more efficient compared to equidistant problem treatment.

***K*eywords:** realtime dynamic optimization, adaptive discretization, wavelets, receding horizon estimation, dynamic data reconciliation

1 Introduction

The numerical solution of dynamic optimization problems is typically quite challenging for large–scale applications. The challenge becomes even more severe when real–time applications are envisaged, since the response time within which a valid solution has to be prompted is fixed. In on–line applications usually the numerical cost and the associated solution time is conservatively estimated and the optimization problem is solved with well developed numerical methods on a typically fixed discretization mesh. There are only few attempts to incorporate adaptivity into the discretization mesh. Biegler et al. [1, 10] suggest a decomposition approach which incorporates some form of adaptivity. In particular they discretize the optimization problem with orthogonal polynomials on finite elements and solve it on a fixed grid in an inner algorithm. The position and length of the finite elements are placed by an outer algorithm such that the approximation quality is improved. The number of finite elements as well as the order of the polynomials remain unchanged. To our knowledge all available methods compute the optimal solution with algorithms where the degree of freedoms and therefore the numerical complexity is fixed. In applications where the response time is given while the computational resources, i. e. the number of available floating point operations, are uncertain, classical approaches might not find a valid solution within the available time.

Dynamic data reconciliation formulated on a moving horizon of fixed length is an example of such demanding restrictions. A reconciled set of measurements and an estimate of unmeasured

[†]Correspondence author. Phone: +49-241-806712, Fax: +49-241-8888326, E-mail: marquardt@lfpt.rwth-aachen.de

[*]This work has been supported by the "Deutsche Forschungsgemeinschaft" under grant no. MA1188/6

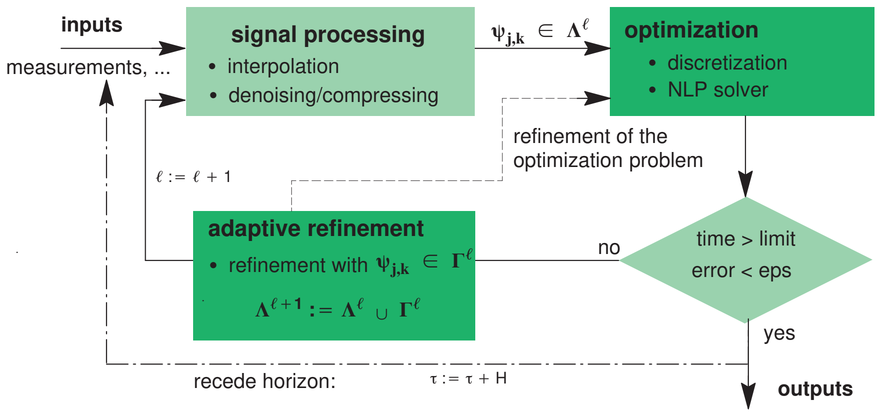

Figure 1: Schematic outline of the refinement strategy applied to dynamic data reconciliation

process quantities must be provided in real–time ideally with minimum delay to the last available measurement. The length of the time interval T employed in moving horizon estimation should be chosen as long as possible to improve the stability of the estimation [20, 21]. For a given problem, algorithm and hardware, the required solution time is governed by some interval h which may strongly vary over time depending on the availability of the process computer and on the frequencies included in the measured (and hence) estimated quantities. Despite these variations in h, the computations must *always* be accomplished in $h \leq H$ where H denotes the sampling interval.

Instead of keeping the degrees of freedom fixed for such problems we focus on a concept which *steadily refines* the optimization problem from a coarse approximation up to arbitrary degree of detail. The optimization problem is solved repeatedly on various discretization meshes with increasing number of decision variables and approximation quality. Such endeavor will prompt after each cycle a valid solution of some accuracy associated with the current discretization mesh. If the time limit is reached and the result on the approximation level ℓ is not yet available, we do still have the last solution of the approximation level $\ell - 1$. The principal outline of such a refinement strategy is shown in figure 1 for data reconciliation as sample application. The inner loop belongs to the refinement cycles whereas the outer loop refers to the moving of the time window. More details on the refinement concept applied to dynamic data reconciliation can be found in [3].

In this framework we have to solve optimization problems of increasing complexity repeatedly. Such a concept is expected to outperform a hierarchy of conventional discretizations provided only incremental computational work is needed for upgrading accuracy according to each refinement step. Several ways of exploiting previously spent computational work when progressing to the next level can be identified:

- adaptive mesh refinement based on information drawn from the previous solution.
- reuse of Hessian and Jacobian matrices (or their approximations),
- exploitation of the previous active set of inequalities,
- use of the previous solution as initial guess for the nonlinear (NLP) and/or quadratic (QP) programming solvers.

Multiscale concepts [8, 11, 13] seems to offer attractive properties which enable exploitation of the previous solution approximations.

The aim of this paper is to explore the potential use of the current solution for an adaptive refinement of the discretization. The goal of adaptive strategies is to achieve a high accuracy at the expense of possibly few degrees of freedom. In particular, we focus here in the sense of a first step towards a more comprehensive treatment on linear dynamic data reconciliation problems on a fixed time window. The outer loop in figure 1 is not treated. For a discussion of further warm start possibilities listed above in the framework of multiscale discretization, we refer to [4, 5].

The proposed adaptive refinement strategy has several ingredients. First, we have to ensure that there is sufficient improvement of approximation quality of the measurements during the refinement cycles. Second the approximation quality of the differential model equation has to be monitored. Third, the approximation quality of the a priori unknown inputs has to be suitably increased. Accurate approximations of the measurements during the refinement cycles are established employing simple thresholding techniques while estimates on the approximation quality of the differential equations is based on an error analysis using one step methods. Finally the input functions are refined employing a sensitivity analysis with respect to the cost functional.

The outline of the paper is as follows: In Section 2 we start with describing briefly the dynamic data reconciliation problem under consideration. To lay the foundation for the discretization in a multiscale setting we introduce in Section 3 wavelets and list their relevant properties. Section 4 treats the discretization of the dynamic optimization problem. The continuously formulated problem will be transformed into an algebraic quadratic programming problem using a Galerkin approach. The adaptive refinement strategy is proposed in Section 5. Refinement of known inputs, approximations to the model equations are developed independently in a preliminary step. In particular we derive formulas for the approximation of the known inputs, the evaluation of the error estimates of the model. Further, we develop an efficient method for the sensitivity evaluation needed to refine the unknown inputs. In a second step interactions among the different kind of variables are taken into account. A short summary on the algorithm is given in Section 6. The paper concludes with two small illustrative examples and a short summary is given.

2 Problem formulation

Dynamic data reconciliation rectifies the measurements taken from a process under transient conditions, estimates states and optimally adjusts potential uncertainties in the mathematical process model to render filtered measurements which are consistent with predicted process dynamics.

The process model under consideration is based on a formulation discussed in more detail in [3]. In this paper we consider a simplified version where the mathematical model of the process is assumed to be linear:

$$\begin{aligned} \dot{\boldsymbol{x}} &= \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{u} \quad , & (1)\\ \boldsymbol{y} &= \boldsymbol{C}\boldsymbol{x} + \boldsymbol{D}\boldsymbol{u} \quad , & (2)\\ \boldsymbol{x}(0) &= \boldsymbol{x}_0 \quad . & (3) \end{aligned}$$

We assume that the time domain of interest has been already transformed to the unity interval $I = [0, 1]$. It should be noted, that an arbitrarily long (but finite) horizon can be chosen in order to improve the stability properties of the estimator. In (1),(2) $\boldsymbol{x}(t) \in \mathbb{R}^{n_x}$ denote the state vector whereas $\boldsymbol{x}_0$ refers to the set of initial conditions. $\boldsymbol{y}(t) \in \mathbb{R}^{n_y}$ represents the model outputs and $\boldsymbol{u}(t) \in \mathbb{R}^{n_u}$ are known input functions. Further $\boldsymbol{A} \in \mathbb{R}^{n_x \times n_x}, \boldsymbol{B} \in \mathbb{R}^{n_x \times n_u}, \boldsymbol{C} \in \mathbb{R}^{n_y \times n_x}, \boldsymbol{D} \in \mathbb{R}^{n_y \times n_u}$ reflect the time–invariant model matrices.

The equations (1), (2) denote the nominal model of the process. Since the model is never perfect, there is always a mismatch between the real process and the model. The mismatch can be accounted for by inclusion of unstructured uncertainty into the model. The simplest model of unstructured uncertainty is an additive time–varying forcing function $\boldsymbol{w}(t) \in \mathbb{R}^{n_w}$, $n_w \leq n_x$, $\boldsymbol{v}(t) \in \mathbb{R}^{n_v}$, $n_v \leq n_y$, in the sequel referred to as error or input functions:

$$\dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{u} + \boldsymbol{W}\boldsymbol{w} := \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{w}) \quad , \tag{4}$$

$$\boldsymbol{y} = \boldsymbol{C}\boldsymbol{x} + \boldsymbol{D}\boldsymbol{u} + \boldsymbol{V}\boldsymbol{v} := \boldsymbol{g}(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{v}) \quad . \tag{5}$$

The rectangular matrices $\boldsymbol{W}, \boldsymbol{V}$ are indicator matrices of full row rank which often contain only one element per column. This formulation considers uncertainty in the model, i.e. the matrices $\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}, \boldsymbol{D}$ as well as in the inputs $\boldsymbol{u}$ and measurements $\boldsymbol{y}$ lumped into $\boldsymbol{v}(t)$ and $\boldsymbol{w}(t)$.

Besides the model equations, there is additional a–priori knowledge which should be stated and used in the estimation procedure. Following Kim et al. [18] or Robertson et al. [22], lower and upper bounds can be formulated for many of the process quantities such as the states $\boldsymbol{x}$, the measurements $\boldsymbol{y}$, and the model and measurement errors $\boldsymbol{v}$ and $\boldsymbol{w}$. These inequality constraints are of the type

$$\boldsymbol{l}_x \leq \boldsymbol{x} \leq \boldsymbol{s}_x \quad , \tag{6}$$

$$\boldsymbol{l}_y \leq \boldsymbol{y} \leq \boldsymbol{s}_y \quad , \tag{7}$$

$$\boldsymbol{l}_v \leq \boldsymbol{v} \leq \boldsymbol{s}_v \quad , \tag{8}$$

$$\boldsymbol{l}_w \leq \boldsymbol{w} \leq \boldsymbol{s}_w \quad , \tag{9}$$

where the $\boldsymbol{l}_i, \boldsymbol{s}_i$ refer to lower and upper bounds, respectively. The dimension and structure of the matrices $\boldsymbol{W}, \boldsymbol{V}$ as well as the formulation of the inequalities are modeling decisions and reflect the best possible knowledge available for the process under consideration [2].

For reconciliation of the measured data as well as for the best possible estimation of the unmeasured states we require that the degrees of freedom in the model (4),(5) are chosen in such a way, that the errors between the predictions $\boldsymbol{y}$ and the measurements $\boldsymbol{z}$ are minimized. Here $\boldsymbol{z}$ denotes an interpolated measurement function which is based upon denoised measurement samples $z(t_i)$. For the cost functional a least squares formulation is chosen as:

$$\Theta := \int_0^1 \Theta_0 \, d\tau := \int_0^1 \big((\boldsymbol{y}(\tau) - \boldsymbol{z}(\tau))^T \boldsymbol{Q} (\boldsymbol{y}(\tau) - \boldsymbol{z}(\tau)) + \boldsymbol{v}(\tau)^T \boldsymbol{R}_v \boldsymbol{v}(\tau) + \boldsymbol{w}(\tau)^T \boldsymbol{R}_w \boldsymbol{w}(\tau) \big) \, d\tau \quad , \tag{10}$$

such that the estimation is given by the optimization problem

$$\min \Theta \tag{11}$$

subject to the constraints (4)–(9). Note, that the initial condition (3) does not explicitly appear as constraint equation, since it is not needed for a correct statement of the estimation problem (see the Appendix for a more detailed discussion). In the most general case the weights $\boldsymbol{Q}, \boldsymbol{R}_w, \boldsymbol{R}_v$ can be operators. $\boldsymbol{Q}$ reflects the measurement covariance whereas $\boldsymbol{R}_w, \boldsymbol{R}_v$ have been introduced to regularize the problem. Typically the choice of the correct regularization is a nontrivial task [16] and has to reflect the tradeoff between bias of the solution on one side and the sensitivity to perturbations in the measurements on the other side.

3 Wavelets

3.1 Wavelet Prerequisites

The central element of the proposed refinement strategy is the use of hierarchically organized bases. In particular, our approach is based on wavelets which offer attractive properties for the construction of an adaptive algorithm to solve the estimation problem. A detailed description of the wavelet methodology is far beyond the scope of this paper. For a more thorough treatment the reader is referred to [8, 11, 13]. Here we introduce only the notation and illuminate the basic properties and their significance for the refinement approach. For simplicity, the case of scalar functions is treated, which naturally extends to vector functions. We define for function vectors $\boldsymbol{f}, \boldsymbol{g}$ the inner product in L_2 by $\langle \boldsymbol{f}, \boldsymbol{g} \rangle := \langle \boldsymbol{f}, \boldsymbol{g} \rangle_{L_2(I)} = \int_0^1 \boldsymbol{f}(t)^T \boldsymbol{g}(t) dt$ with $\boldsymbol{f}, \boldsymbol{g} \in L_2(I)^n$ and the notation $\langle\langle \boldsymbol{f}, \boldsymbol{g} \rangle\rangle := \int_0^1 \boldsymbol{f}, \boldsymbol{g}^T dt$ for the matrix of all possible scalar products $(\langle f_i, g_j \rangle)_{(i,j)}$.

We are interested in wavelets on the interval $I = [0, 1]$. So called multiresolution sequences of nested spaces $S_{j0} \subset \ldots \subset S_j \subset S_{j+1} \subset \ldots$ whose union is dense in $L_2(I)$ usually form the starting point for their construction. Here j_0 stands for some coarsest discretization level. The spaces S_j are originally spanned by *scaling functions* $S_j := span\{\varphi_{j,k} : k \in \mathcal{I}_j\}$ where diam supp $\varphi_{j,k} \approx 2^{-j}$ can be viewed as the mesh size of S_j. Except for certain adjustments near the end points of the interval [12] the basis functions $\varphi_{j,k}$ are generated by translation and dilation of a single function $\varphi^{(m)}$. At this point we employ *B-splines* of order m [14] as the generating function $\varphi^{(m)}$. There exits a dual generator $\tilde{\varphi}^{(m,\tilde{m})}$, $m, \tilde{m} \in \mathbb{N}$ such that

$$\langle \varphi(\cdot)^{(m)}, \tilde{\varphi}^{(m,\tilde{m})}(\cdot - k) \rangle = \delta_{0,k} \qquad k \in \mathbb{Z} \quad ,$$

Let us assume we approximate a function $f \in L_2(I)$ on a coarse scale with $f_j \in S_j$ and refine the approximation to $f_{j+1} \in S_{j+1}$. Then it is very convenient for the representation of f_{j+1} to keep the representation of f_j and just add additional information. The detail $f_{j+1} - f_j$ lies in the difference or detail space W_j with $S_{j+1} = S_j \oplus W_j$. If the chosen basis functions $\{\psi_{j,k}\}$ of W_j meet certain properties they are called wavelets. A function $f_J \in S_J$ has two representations in S_J and $S_{j_0} \oplus_{k=j_0}^{J-1} W_k$ namely

$$f_J \quad = \quad \sum_k c_{J,k} \varphi_{J,k} = \sum_{j=j_0-1}^{J-1} \sum_k d_{j,k} \psi_{j,k}$$

with $\psi_{j_0-1,1} := \varphi_{j_0,k}$. To keep the notation simple we express the above *single* and *multiscale* representation of f_J in vector form according to

$$f_J = \boldsymbol{c}^T \boldsymbol{\Phi}_J = \boldsymbol{d}^T \boldsymbol{\Psi}_J \quad . \tag{12}$$

Both representations are related by the fast wavelet transformation (FWT), $\boldsymbol{c} = \boldsymbol{T}_J \boldsymbol{d}$, where $\boldsymbol{T}_J : \boldsymbol{d} \rightarrow \boldsymbol{c}$ is typically executed in cascadic fashion requiring only $O(N_J)$ operations. Here N_J denotes the dimension of the space S_J.

The wavelet concept is based on the fact that there exist dual scaling and difference spaces $\tilde{S}_j$ and $\tilde{W}_J$ which are constructed in a similar way based on a scaling function $\tilde{\varphi}^{(m,\tilde{m})}$. For the sake of brevity we do not explicitly give the definitions for the dual function relations, instead we refer to [9] where the dual generators are constructed for B-splines $\varphi^{(m)}$. The wavelets $\psi, \tilde{\psi}$ depend on the choice of $m, \tilde{m}$ and should therefore be consequently denoted by $\psi^{(m,\tilde{m})}, \tilde{\psi}^{(m,\tilde{m})}$. To ease the notation we omit the subscript but keep this dependency in mind.

The dual spaces are related to the primal spaces through the following biorthogonality conditions:

$$\begin{aligned} &\langle\langle \boldsymbol{\Psi_j}, \tilde{\boldsymbol{\Psi}}_{\boldsymbol{j}} \rangle\rangle = \boldsymbol{I}\,, \quad \langle\langle \boldsymbol{\Phi}_j, \tilde{\boldsymbol{\Phi}}_j \rangle\rangle = \boldsymbol{I}\,, \quad j \in \mathbb{N}, \text{and as consequence} \\ &\tilde{W}_j \perp S_j\,, \quad W_j \perp \tilde{S}_j \quad . \end{aligned}$$

Here $\boldsymbol{I}$ denotes the identity matrix. These biorthogonality properties form essential preconditions for the important stability property of the basis:

$$c_1 \parallel \boldsymbol{d}_f \parallel_{l_2} \leq \parallel f \parallel_{L_2} \leq c_2 \parallel \boldsymbol{d}_f \parallel_{l_2} \quad , c_1 \leq c_2 \quad , \tag{13}$$

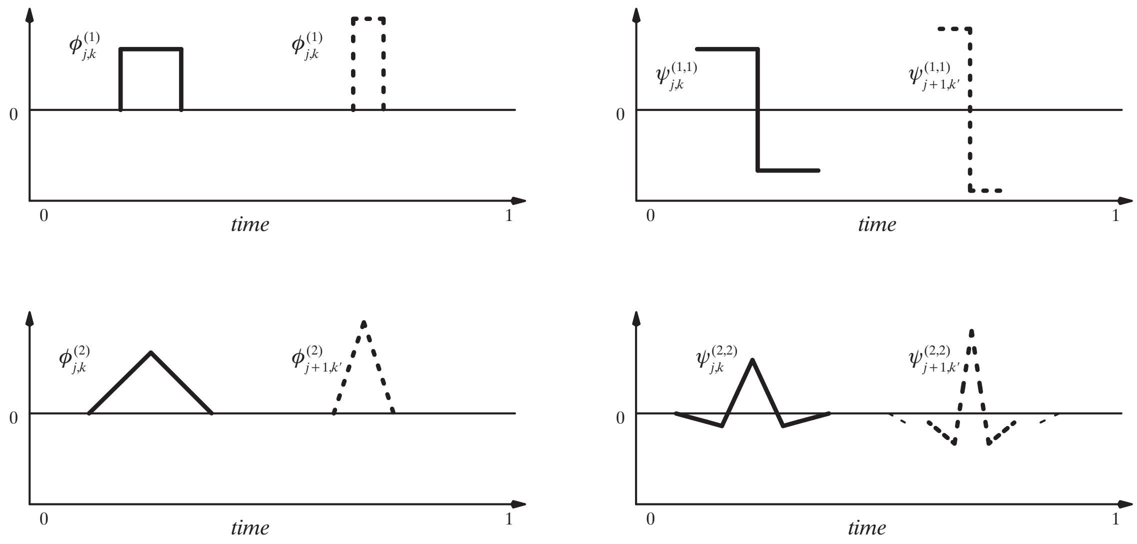

Figure 2: Scaling functions $\varphi_{j,k}^{(m)}, m = 1, 2$ and their corresponding wavelets for the choice $\tilde{m} = 1, 2$ are displayed. The latter are denoted conveniently as Haar and hat wavelets, respectively.

with the multiscale coefficients $\boldsymbol{d}_f$ of f. Thus small variations in the coefficients result in small variations of the function and vice versa – a crucial property for numerical calculations. One further essential property is the local support of wavelets. Wavelet coefficients reflect local changes of the function. Furthermore, the local support, translation and dilation of ψ lead to nearly sparse matrices for the inner products $\langle\langle \boldsymbol{\Psi}, \boldsymbol{\Psi} \rangle\rangle$ which reveal a distinctive structure, the so called finger structure. The number of nonzero elements in the inner product matrix is of $O(k \log k)$ where k reflects the number of wavelets used.

Additionally the wavelets are constructed such that they fulfill the moment conditions

$$\begin{aligned} \int_0^1 x^r \psi_{j,k}(x) dx &= 0\,, \qquad r = 0, \ldots, \tilde{m} - 1\,, \\ \int_0^1 x^r \tilde{\psi}_{j,k}(x) dx &= 0\,, \qquad r = 0, \ldots, m - 1\,. \end{aligned} \tag{14}$$

An immediate consequence of (14) is that one can subtract any polynomial P of order m from a function without changing its wavelet coefficients. If f_J is smooth on the support of $\tilde{\psi}_{j.k}$ the coefficients $\mid d_{j,k} \mid$ will be quite small. Moreover, on account of the norm equivalence (13), discarding small coefficients causes only a small error in the function. Therefore such functions may be approximated very accurately by linear combinations of only relatively very few wavelet basis functions corresponding to the large coefficients. In fact, this number may be much smaller than the dimension of the full space S_J corresponding to the highest refinement scale among the significant coefficients. The possibly much sparser wavelet representation determined by the significant coefficients of the approximated object means to employ an individually adapted trial space which could be much smaller than S_J and thus leads to a much more economic discretization concept. Such a possibly *sparse basis* is denoted by

$$\boldsymbol{\Psi}_\Lambda := \{\psi_{j,k} : (j,k) \in \Lambda\} \tag{15}$$

where $\Lambda \subseteq \Lambda^J := \{(j,k) \mid j_0 - 1 \leq j \leq J - 1, k \in \mathcal{I}_j\}$ is an arbitrary subset of the *fully refined wavelet basis* Λ^J containing the coarsest level S_{j_0}. The space spanned by $\boldsymbol{\Psi}_\Lambda$ will be denoted by W_Λ. Note that working with the adapted multiscale space W_Λ can be viewed as working on a non–uniform mesh.

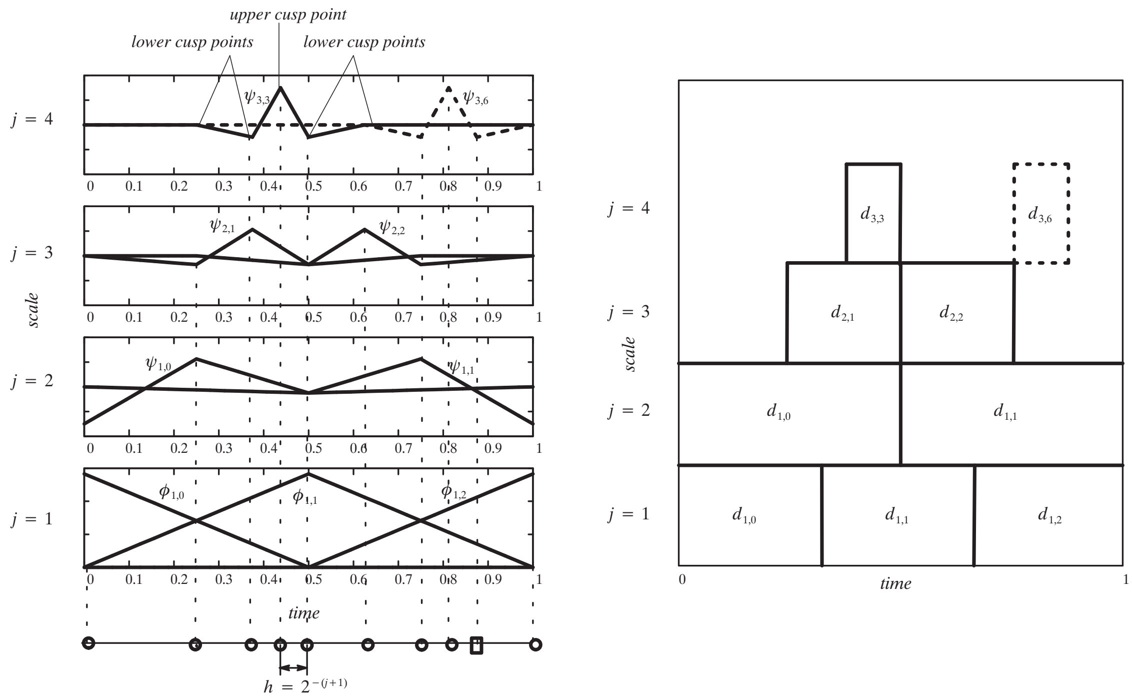

Figure 3: Sparse wavelet basis in time frequency representation. The wavelet basis is linked to a unique, potentially non–equidistant mesh which is shown in the lower left corner.

3.2 Employed wavelet spaces

In this paper we consider only two choices of multiscale spaces: One is the biorthogonal wavelet spaces spanned by the so called *hat* function $\varphi^{(m)}$ with $m = 2$ and corresponding dual scaling function $\tilde{\varphi}^{(2,2)}$. The other choice $m = 1, \tilde{m} = 1$ gives rise to the so called *Haar* wavelets. The two choices lead to *piecewise linear* or *piecewise constant* approximations respectively. For illustration, representatives of the primal scaling functions as well as the corresponding wavelets are shown in figure 2. To distinguish both types of scaling functions and wavelets we denote the Haar wavelet and the corresponding scaling function basis by $\boldsymbol{\Psi}^H$ and $\boldsymbol{\Phi}^H$ respectively and we omit the superscript for the hat wavelet and scaling function. Since Haar wavelets are orthogonormal we have $\langle \boldsymbol{\Psi}^H, \boldsymbol{\Psi}^H \rangle = \boldsymbol{I}$. Both choices of wavelet spaces are dense in $L_2(I)$, but the multiscale space corresponding to the hat function belongs to the Sobolev space $H^1(I)$ which is necessary for the weak formulation of the constraints on the states $\boldsymbol{x}$.

Next, we introduce a common way to graphically illustrate the sparse basis and illuminate the consequences for the underlying discretization mesh. The example under consideration involves the hat wavelet but everything extends in a similar manner to other wavelets. Assume that f is approximated in sparse wavelet representation $\boldsymbol{d}_f^T \boldsymbol{\Psi}_\Lambda$. An arbitrarily chosen example of a sparse basis is shown in figure 3 where the functions are scalewise displayed. The coefficients $d_{j,k}, (j,k) \in \Lambda$ can be displayed in the so called *time–frequency plane* where a specific patch is assigned to each coefficient $d_{j,k}$ (see figure 3, right part).

As already mentioned before, the space W_Λ can be related to a specific discretization mesh. Consider the example displayed in figure 3. Since the wavelets are piecewise linear functions we are able to formulate a possibly *non–equidistant mesh* $\triangle_\Lambda = \{t_i \in I\}$ collecting the knot points t_i. The smallest possible mesh size is given by $h = 2^{-(J+1)}$ where J denotes the largest scale which is contained in Λ. For arbitrary index sets the *cardinality* of mesh points $\triangle_\Lambda$ and basis functions Λ is usually not the same, i.e. $\sharp \triangle_\Lambda \neq \sharp\Lambda$. However, under the assumption, that the basis functions form a so called *graded* tree, $\sharp\ \triangle_\Lambda = \sharp\Lambda$ holds. A graded tree requires, that for all $\psi_{j,k} \in \Lambda$ the local neighbors on the scale $j-1$ are contained in Λ, i. e. $\psi_{j-1,k/2}, \psi_{j-1,k/2-1} \in \Lambda,\ k$ even, $\psi_{j-1,(k-1)/2}, \psi_{j-1,(k+1)/2-1} \in \Lambda,\ k$ odd. As a consequence of the graded tree criteria the local mesh ratio is bounded, i.e. the mesh sizes next to each other can only be doubled, halvend or remain the same. Typically this is denoted as quasi uniform grid. The tree criteria can easily be checked by inspection of the time–frequency representation. The example shown in figure 3 clearly violates the criteria since $\psi_{2,3} \notin \Lambda$ which is necessary for $\psi_{3,6}$. As a consequence the mesh consisting of 10 mesh points is spanned by 9 basis functions and causes unwanted spikes in the function representation. For complete tree structure the lower cusp points of the wavelets coincide with upper cusp points which belong to wavelets of lower scale. Therefore it is possible to relate the wavelets directly to mesh points by a one to one correlation.

4 Discretization

4.1 Weak problem formulation

The optimization problem (11) is discretized in a way which supports the concept of adaptivity. A multiscale formulation of the process model which permits model representation of various resolutions is obtained by the following weak formulation:

$$\min \int_0^1 (\boldsymbol{y}-\boldsymbol{z})^T \boldsymbol{Q} (\boldsymbol{y}-\boldsymbol{z}) + \boldsymbol{w}^T \boldsymbol{R}_w \boldsymbol{w} + \boldsymbol{v}^T \boldsymbol{R}_v \boldsymbol{v}\ d\tau \tag{16}$$

subject to the constraints

$$\begin{aligned}
\langle \dot{\boldsymbol{x}} - \boldsymbol{f}(\boldsymbol{x},\boldsymbol{u},\boldsymbol{w}), \boldsymbol{\zeta}_x \rangle &= \boldsymbol{0} \quad \forall\, \boldsymbol{\zeta}_x \in (L_2(I))^{n_x}\,, && (17)\\
\langle \boldsymbol{y} - \boldsymbol{g}(\boldsymbol{x},\boldsymbol{u},\boldsymbol{v}), \boldsymbol{\zeta}_y \rangle &= \boldsymbol{0} \quad \forall\, \boldsymbol{\zeta}_y \in (L_2(I))^{n_y}\,, && (18)\\
\langle \boldsymbol{x} - \boldsymbol{s}_x, \boldsymbol{\nu}_{(x,s)} \rangle &\le \boldsymbol{0} \quad \forall\, \boldsymbol{\nu}_{(x,s)} \in (L_2^+(I))^{n_x}\,, && (19)\\
\langle \boldsymbol{y} - \boldsymbol{s}_y, \boldsymbol{\nu}_{(y,s)} \rangle &\le \boldsymbol{0} \quad \forall\, \boldsymbol{\nu}_{(y,s)} \in (L_2^+(I))^{n_y}\,, && (20)\\
\langle \boldsymbol{w} - \boldsymbol{s}_w, \boldsymbol{\nu}_{(w,s)} \rangle &\le \boldsymbol{0} \quad \forall\, \boldsymbol{\nu}_{(w,s)} \in (L_2^+(I))^{n_w}\,, && (21)\\
\langle \boldsymbol{v} - \boldsymbol{s}_v, \boldsymbol{\nu}_{(v,s)} \rangle &\le \boldsymbol{0} \quad \forall\, \boldsymbol{\nu}_{(v,s)} \in (L_2^+(I))^{n_v}\,, && (22)\\
\langle \boldsymbol{l}_x - \boldsymbol{x}, \boldsymbol{\nu}_{(x,l)} \rangle &\le \boldsymbol{0} \quad \forall\, \boldsymbol{\nu}_{(x,l)} \in (L_2^+(I))^{n_x}\,, && (23)\\
\langle \boldsymbol{l}_y - \boldsymbol{y}, \boldsymbol{\nu}_{(y,l)} \rangle &\le \boldsymbol{0} \quad \forall\, \boldsymbol{\nu}_{(y,l)} \in (L_2^+(I))^{n_y}\,, && (24)\\
\langle \boldsymbol{l}_w - \boldsymbol{w}, \boldsymbol{\nu}_{(w,l)} \rangle &\le \boldsymbol{0} \quad \forall\, \boldsymbol{\nu}_{(w,l)} \in (L_2^+(I))^{n_w}\,, && (25)\\
\langle \boldsymbol{l}_v - \boldsymbol{v}, \boldsymbol{\nu}_{(v,s)} \rangle &\le \boldsymbol{0} \quad \forall\, \boldsymbol{\nu}_{(v,l)} \in (L_2^+(I))^{n_v}\,, && (26)
\end{aligned}$$

where $L_2^+(I) = \{s \in L_2(I) : s \ge 0 a.e.on I\}$ refers to positive function spaces. For convenience we gather the unknows in $\boldsymbol{\Upsilon} := (\boldsymbol{x}^T, \boldsymbol{y}^T, \boldsymbol{w}^T, \boldsymbol{v}^T)^T \in \boldsymbol{\Upsilon}$ and the test functions in $\boldsymbol{\zeta} := (\boldsymbol{\zeta}_x^T, \boldsymbol{\zeta}_y^T, \boldsymbol{\nu}_{(x,s)}^T, \boldsymbol{\nu}_{(y,s)}^T, \boldsymbol{\nu}_{(w,s)}^T, \boldsymbol{\nu}_{(v,s)}^T, \boldsymbol{\nu}_{(x,l)}^T, \boldsymbol{\nu}_{(y,l)}^T, \boldsymbol{\nu}_{(w,l)}^T, \boldsymbol{\nu}_{(v,l)}^T) \in \mathcal{M}$ where

$$\boldsymbol{\Upsilon} := (H^1(I))^{n_x} \times (L_2(I))^{n_y+n_w+n_v}, \quad \mathcal{M} := (L_2(I))^{n_x+n_y} \times (L_2^+(I))^{2(n_x+n_y+n_w+n_v)} \quad .$$

Finite dimensional spaces of $\boldsymbol{\Upsilon}, \mathcal{M}$ will be chosen in the following section.

4.2 Discrete problem formulation

We chose appropriate finite dimensional wavelet spaces $\boldsymbol{\Upsilon}_\Lambda$ and $\boldsymbol{\mathcal{M}}_\Lambda$ as subsets of $\boldsymbol{\Upsilon}$ and $\boldsymbol{\mathcal{M}}$ and obtain a finite dimensional, discretized problem. More precisely, the components of the trial space $\boldsymbol{\Upsilon}_\Lambda \subseteq \boldsymbol{\Upsilon}$ corresponding to the variables $\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{v}$ are spanned by collections of hat wavelets $\psi_{j,k}$ while $\boldsymbol{w}, \boldsymbol{u}$ are spanned by collections of Haar wavelets $\psi^H_{j,k}$. Specifically, the test space $\boldsymbol{\mathcal{M}}_\Lambda \subseteq \boldsymbol{\mathcal{M}}$ are chosen such that simple and sparse structures of the resulting discretization matrices are obtained. The state and output equations (17),(18) are tested by Haar wavelets $\psi^H_{j,k}$ and dual wavelets $\tilde{\psi}^{(2,2)}$, respectively and yield discretization submatrices which exhibit finger and diagonal structure. Since the trial functions are piecewise constant or linear, it is sufficient to evaluate the inequalities (19)-(26) pointwise on the unique mesh Δ_Λ which is associated with the finite index set Λ (see Section 3.2). Due to the low order of the trial functions the constraints are never violated by an overshoot between the mesh points.

To ensure stability of the discretization, the spaces $\boldsymbol{\Upsilon}_\Lambda$, $\boldsymbol{\mathcal{M}}_\Lambda$ have to satisfy the LBB-condition. This means that $\boldsymbol{\mathcal{M}}_\Lambda$ may not be too rich relative to $\boldsymbol{\Upsilon}_\Lambda$. Specifically this requires, that the index set Λ'_x, -determining the test functions $\psi^H_{j,k}$ for the state equations, has to be a subset of the index set Λ_x, which corresponds to the trial space of the states $\boldsymbol{x}$. Since the space spanned by the Haar wavelets has one scaling function less than the space of hat wavelets, we set Λ'_x as Λ_x and exclude the index of the missing scaling function. The dimension of Λ'_x is therefore always lower than the dimension of Λ_x.

The discretized version of (11) reads now as

$$\min \ (\boldsymbol{d}_y - \boldsymbol{d}_z)^T \boldsymbol{Q}_\Lambda (\boldsymbol{d}_y - \boldsymbol{d}_z) + \boldsymbol{d}_w^T \boldsymbol{W}_\Lambda \boldsymbol{d}_w + \boldsymbol{d}_v^T \boldsymbol{V}_\Lambda \boldsymbol{d}_v \tag{27}$$

subject to the constraints

$$\begin{aligned}
\boldsymbol{X}_\Lambda \boldsymbol{d}_x &= \boldsymbol{B}_\Lambda \boldsymbol{d}_u + \boldsymbol{W}_\Lambda \boldsymbol{d}_w \quad , && (28)\\
\boldsymbol{I}_\Lambda \boldsymbol{d}_y &= \boldsymbol{C}_\Lambda \boldsymbol{d}_x + \boldsymbol{D}_\Lambda \boldsymbol{d}_u \quad , && (29)\\
l_{x_i}\boldsymbol{e} &\leq \boldsymbol{T}_{\Delta_{x,i}} \boldsymbol{d}_{x_i} \leq s_{x_i}\boldsymbol{e} \, , \quad \forall i = 1, \dots, n_x \quad , && (30)\\
l_{y_i}\boldsymbol{e} &\leq \boldsymbol{T}_{\Delta_{yi}} \boldsymbol{d}_{y_i} \leq s_{y_i}\boldsymbol{e} \, , \quad \forall i = 1, \dots, n_y \quad , && (31)\\
l_{w_i}\boldsymbol{e} &\leq \boldsymbol{T}_{\Delta_{w,i}} \boldsymbol{d}_{w_i} \leq s_{w_i}\boldsymbol{e} \, , \quad \forall i = 1, \dots, n_w \quad , && (32)\\
l_{v_i}\boldsymbol{e} &\leq \boldsymbol{T}_{\Delta_{v,i}} \boldsymbol{d}_{v_i} \leq s_{v_i}\boldsymbol{e} \, , \quad \forall i = 1, \dots, n_v \quad . && (33)
\end{aligned}$$

In this notation the vector $\boldsymbol{d}_\xi = [\boldsymbol{d}_{\xi_1}^T, \dots, \boldsymbol{d}_{\xi_{n_\xi}}^T]$ comprises all the wavelet coefficients of the approximations $\xi_i = \boldsymbol{d}_{\xi_i}^T \Psi_{\Lambda_{\xi_i}}, i = 1, \dots, n_\xi$, whereas $\boldsymbol{T}_{\Delta_{\xi,i}}$ denotes a transformation matrix which evaluates the multiscale representation pointwise at meshpoints given by $\Delta_{\Lambda_{\xi_i}}$. $\boldsymbol{e}$ refers to the unit vector $\boldsymbol{e} = [1, \dots, 1]$. The discretization matrices in problem (27) reflect the inner products of trial and test functions. The discretization matrices are defined by

$$\begin{aligned}
(\boldsymbol{Q}_\Lambda)_{i,j} &= \boldsymbol{Q}_{i,j} \langle\langle \boldsymbol{\Psi}_{\Lambda_{y_i}}, \boldsymbol{\Psi}_{\Lambda_{y_j}} \rangle\rangle \quad ,\\
(\boldsymbol{R}_{v_\Lambda})_{i,j} &= \boldsymbol{R}_{v_{i,j}} \langle\langle \boldsymbol{\Psi}_{\Lambda_{v_i}}, \boldsymbol{\Psi}_{\Lambda_{v_j}} \rangle\rangle \quad ,\\
(\boldsymbol{R}_{w_\Lambda})_{i,j} &= \boldsymbol{R}_{w_{i,j}} \langle\langle \boldsymbol{\Psi}_{\Lambda_{w_i}}, \boldsymbol{\Psi}_{\Lambda_{w_j}} \rangle\rangle \quad ,\\
(\boldsymbol{W}_\Lambda)_{i,j} &= \boldsymbol{W}_{i,j} \langle\langle \boldsymbol{\Psi}^H_{\Lambda'_{x_i}}, \boldsymbol{\Psi}^H_{\Lambda_{w_j}} \rangle\rangle \quad ,\\
(\boldsymbol{V}_\Lambda)_{i,j} &= \boldsymbol{V}_{i,j} \langle\langle \tilde{\boldsymbol{\Psi}}_{\Lambda_{y_i}}, \boldsymbol{\Psi}_{\Lambda_{v_j}} \rangle\rangle \quad ,\\
(\boldsymbol{C}_\Lambda)_{i,j} &= \boldsymbol{C}_{i,j} \langle\langle \tilde{\boldsymbol{\Psi}}_{\Lambda_{y_i}}, \boldsymbol{\Psi}_{\Lambda_{x_j}} \rangle\rangle \quad ,\\
(\boldsymbol{I}_\Lambda)_{i,j} &= \boldsymbol{I}_{i,j} \langle\langle \tilde{\boldsymbol{\Psi}}_{\Lambda_{y_i}}, \boldsymbol{\Psi}_{\Lambda_{y_j}} \rangle\rangle \quad ,\\
(\boldsymbol{X}_\Lambda)_{i,j} &= \boldsymbol{I}_{i,j} \langle\langle \boldsymbol{\Psi}^H_{\Lambda'_{x_i}}, \dot{\boldsymbol{\Psi}}_{\Lambda_{x_j}} \rangle\rangle - \boldsymbol{A}_{i,j} \langle\langle \boldsymbol{\Psi}^H_{\Lambda'_{x_i}}, \boldsymbol{\Psi}_{\Lambda_{x_j}} \rangle\rangle \quad ,
\end{aligned}$$

where $(\boldsymbol{\Xi})_{i,j} = \boldsymbol{\Xi}_{i,j} \langle\langle \boldsymbol{\Psi}_{\Lambda_{\xi_i}}, \boldsymbol{\Psi}_{\Lambda_{\xi_j}} \rangle\rangle$ refers to the (i,j) block of dimension $\sharp\Lambda_i \times \sharp\Lambda_j$ of the matrix $\boldsymbol{\Xi}_\Lambda$. Note, that the discretization is formulated for *any* refinement level ℓ which is omitted here and below to ease notation.

The above discretization results in a possibly large number of simple inequalities (30)-(33) and a large–scale linear equation system (28)-(29) constraining a quadratic cost functional. In principle the QP problem can be solved with any available quadratic programming technique. However the large–scale QP reveals a lot of structure which must be exploited to render highest possible efficiency. Such a tailored strategy is discussed in more detail in [5] but is not of concern for the presented adaptive strategy.

4.3 Stability of differential equation discretization

Before we proceed with the development of the adaptive refinement strategy we will comment on stability properties inherent to the discretization scheme presented. The discussion is restricted to the differential equation (17) only. Stability and approximation properties of the optimization problem are not considered. If the trial basis functions in $\boldsymbol{\Psi}_{\Lambda_x}$

- correspond to a complete tree,
- induce the same mesh for each state $\boldsymbol{x}_i$, i. e. $\Lambda_x = \Lambda_{x_i}\ \forall i = 1, \ldots, n_x$,

then we are able to show that the above discretization scheme on a single level is equivalent to the trapezoidal integration formula. Instead of the recursive evaluation usually employed in time integration, we evaluate all equations simultaneously For the autonomous system

$$\dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x}$$

the trapezoidal rule is given by

$$(\boldsymbol{I} - \frac{1}{2}h\boldsymbol{A})x(t_{k+1}) = (\boldsymbol{I} + \frac{1}{2}h\boldsymbol{A})x(t_k)$$

where $h = t_{k+1} - t_k$ and $t_k \in \Delta_\Lambda$. The integration scheme is A-stable and of local approximation order $O(h^3)$. Furthermore, it is also known, that the stepsize h has to be chosen small enough, to capture the largest absolute eigenvalue of the system. Otherwise, although being A-stable, the trapezoidal rule exhibits oscillatory behaviour, which means that during the recursion the solution changes its sign from one recursion step to the other.

Under the above stated assumptions, the same properties hold also for the multiscale discretization of the differential equation.

5 Refinement

The discretization setting given in Section 4 covers, in principle, non–uniform meshes for each quantities where the exactness of the solution is governed by the choice of the trial and test spaces $\Upsilon_\Lambda, \mathcal{M}_\Lambda$. Recall that Λ_ℓ denotes the "old" index set which gives rise to the trial space Υ_{Λ_ℓ}, while "new" trial functions actually added in the refinement step are given by the set Γ_ℓ. Hence the resulting trial space after one step is $\Upsilon_{\Lambda_{\ell+1}}$ where $\Lambda_{\ell+1} = \Lambda_\ell \cup \Gamma_\ell$.

In the refinement strategy we first group the quantities into known functions $\boldsymbol{z}, \boldsymbol{u}$, state and output variables $\boldsymbol{x}, \boldsymbol{y}$ and input functions $\boldsymbol{w}, \boldsymbol{v}$. Separate strategies to derive (preliminary) refinement sets Γ'_ℓ are developed for each class of the variables. In a second step we take interactions and coupling into account and construct from the preliminary sets Γ'_ℓ the actually used sets Γ_ℓ. In particular, the sets $\Gamma'_{z_\ell}, \Gamma'_{u_\ell}$ of the known functions $\boldsymbol{z}, \boldsymbol{u}$ are determined with the aid of the wavelet norm equivalences. The refinement sets $\Gamma'_{x_\ell}, \Gamma'_{y_\ell}$ of $\boldsymbol{x}, \boldsymbol{y}$ are obtained through local error estimates while the construction of $\Gamma'_{w_\ell}, \Gamma'_{v_\ell}$ is based on a sensitivity analysis.

5.1 Refinement of $\boldsymbol{z}, \boldsymbol{u}$

The functions z_i, u_i are measurement functions, which have been obtained by linear interpolation of possibly non-equidistant discrete measurements. The measurement functions are first expanded into single scale representation S_J where J is chosen sufficiently large to capture all measurement samples. The representation is then transformed into the equivalent multiscale representation by applying the fast wavelet transformation such that $\boldsymbol{z}_i^J = \boldsymbol{d}_{z_i}^T \boldsymbol{\Psi}^J, \boldsymbol{u}_i^J = \boldsymbol{d}_{u_i}^T \boldsymbol{\Psi}^J$. In practice the measurements are usually corrupted by noise and measurement outliers. The continuous representation will then also contain such deficiencies. However, appropriate pre–processing can be applied to denoise the measurements noting that wavelets offer powerful tools for that task as well. Efficient treatment of the measurement deficiencies is an important topic by itself but is not treated in this paper. Instead we refer to the signal processing literature, e.g. [23].

Keeping the refinement strategy in mind we are interested in representing $\boldsymbol{z}^J, \boldsymbol{u}^J$ on various levels of resolutions. We outline the refinement only for $\boldsymbol{z}$. Analogous results hold also for $\boldsymbol{u}$.

Let us seek for compressed approximations $\boldsymbol{z}_{\Lambda_\ell}$ of $\boldsymbol{z}^J$ in a sense that a specified relative accuracy of the approximation is achieved according to

$$\frac{\| \boldsymbol{z}^J - \boldsymbol{z}_{\Lambda_\ell} \|_{L_2}}{\| \boldsymbol{z}^J \|_{L_2}} \leq \varepsilon'_\ell \quad . \tag{34}$$

On account of the norm equivalences (13) this relation holds provided that

$$\| \boldsymbol{d}_z^J - \boldsymbol{d}_{z_{\Lambda_\ell}} \|_{l_2} \leq \frac{c_2}{c_1} \| \boldsymbol{d}_z^J \|_{l_2} \varepsilon'_\ell := \epsilon_\ell \tag{35}$$

where c_1, c_2 are positive constants, which are dependent on the function basis only. To ensure (35) we simply have to order the vector $\boldsymbol{d}_z^J$ by magnitude and successively sum up the smallest elements until the limit ϵ_ℓ is reached. The remaining indices of the coefficients which have not been collected are the significant coefficients and form the index set Λ_ℓ. Typically the number of significant wavelet coefficients is far less than the number of discrete measurement samples. During the refinement sequence the approximation quality is increased $\varepsilon'_{\ell+1} \leq \varepsilon'_\ell$. The trial functions which are needed to have a refined approximation are then collected in the set Γ'_{z_ℓ}.

5.2 Refinement of $\boldsymbol{x}, \boldsymbol{y}$

The refinement of the model equation is based on the local error estimation where the input functions are kept at their current values $\boldsymbol{w}_{\Lambda_\ell}, \boldsymbol{v}_{\Lambda_\ell}, \boldsymbol{u}_{\Lambda_\ell}$

Although the discretization structure permits individual non-equidistant meshes for $\boldsymbol{x}, \boldsymbol{y}$ we restrict ourself in this paper to the simpler case where one mesh is used to discretize all x_i, y_i. The assumption that we use the same mesh for the outputs and states is reasonable since the output functions are a linear combination of some of the states. Therefore if all the states live on the same mesh it makes little sense to separate the mesh of states and outputs. The approximation error of the outputs $\boldsymbol{y}$ depend linearly on $\boldsymbol{x}$ and it is therefore sufficient to base the refinement of $\boldsymbol{x}, \boldsymbol{y}$ only on the error evaluation of the differential equation.

To lay the foundation of the refinement of $\boldsymbol{x}$, we revisit known results concerning the integration of differential equation systems. Let us consider the constraints (4) of the optimization problem (11) where we introduce in addition an initial condition:

$$\dot{\boldsymbol{x}} \;=\; \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{w}) \quad , \boldsymbol{x}(0) = \boldsymbol{x}_0 \quad .$$

Furthermore assume that $\boldsymbol{x}_0, \boldsymbol{u}, \boldsymbol{w}$ are known and fixed. The approximation error of an integration formula can be estimated by comparing two approximations computed with different step sizes. In particular, for the case of step size halving an error estimate is given by [15]:

$$\epsilon_{x,h}(t_i) \quad := \quad \boldsymbol{x}(t_i) - \boldsymbol{x}_h(t_i) \approx \frac{\boldsymbol{x}_h(t_i) - \boldsymbol{x}_{2h}(t_i)}{2^p - 1} \quad ,$$

where p denotes the approximation order of the integration formula. In our specific example we compare the previously computed solutions $\boldsymbol{x}_\Lambda(t_i)$ with a locally refined solution. The latter is obtained by two successively executed integration steps with halved step size starting with the initial value $\boldsymbol{x}_\Lambda(t_{i-1})$. As has been already pointed out in Section 4.3 the proposed discretization scheme is under the assumption, that the trial functions form a complete tree structure, equivalent to the trapezoidal integration formula. Therefore, it is appropriate to compute the locally refined solution with the trapezoidal integration formula. The evaluation requires the execution of $2 \times \sharp(\Delta_\Lambda - 1)$ integration steps. We have to solve in each step a linear equation system which is of the same size as the model dimensions. At first glance this seems to be a substantial numerical effort. But one has to realize that the local solutions are computed by solving a sequence of decoupled equation systems of relatively small size and is therefore much cheaper then resolving the complete refined optimization problem. Furthermore, the numerical burden can be substantially lowered for linear models if the mesh Δ_Λ involves equal step size. Since in such cases the matrices in the integration formula only depend on the step size for a given problem, we only have to factor such matrices once for each occurring step size. The numerical effort can be even lowered further if the local error is evaluated at the midpoints rather than at the mesh points. Since the trial functions interpolate linearly between the mesh points we do have the values of the coarse approximation at the midpoints. Thus this reduces the numerical cost by a factor of 2. Instead of using absolute error estimates we are interested in the relative ones such that the error estimates on the mesh midpoints are given by

$$\varepsilon_{x,h}((t_i + t_{i+1})/2) \quad := \quad \frac{\boldsymbol{x}_h((t_i + t_{i+1})/2) - \boldsymbol{x}_\Lambda((t_i + t_{i+1})/2)}{(2^p - 1)(\max(\boldsymbol{x}_\Lambda((t_i + t_{i+1})/2), \text{thresh})} \quad , \tag{36}$$

where 'thresh' denotes a threshold value to take care of values close to zero.

Once the local errors have been estimated at the grid midpoints we collect the trial functions into the refinement set Γ'_{x_ℓ}, which are associated with those points yielding large absolute values of the local errors. The number of additional trial functions can be either fixed or can be kept dependent on a chosen threshold. The former ensures that the numerical complexity of the refinement step can be controlled, since the number of additional trial functions in each refinement is determined a priori. The latter allows us to realize a prespecified approximation accuracy. All local errors are considered which are above the chosen threshold.

5.3 Refinement of the input $\boldsymbol{w}, \boldsymbol{v}$

The central idea in the refinement of the input variables $\boldsymbol{w}, \boldsymbol{v}$ lies in the evaluation of the directional sensitivities of the cost functional, with respect to potential new trial functions which are currently not used in the discretization. The sensitivities are taken as an approximation quality indicator. Trial functions with large local sensitivities are likely to have large impact on the approximation quality and therefore will be collected in the refinement sets $\Gamma'_{w_\ell}, \Gamma'_{v_\ell}$. Before we outline the refinement strategy we will first describe the mathematical framework for an efficient evaluation of the sensitivities.

5.3.1 Sensitivity evaluation of the discretized problem

With the choices of the trial and test spaces given in Section 4.2 we have obtained an algebraic optimization problem. For the moment we neglect the inequality constraints. The Lagrange function of the discretized problem is given by

$$\mathcal{L}_\Lambda = \Theta_\Lambda + \boldsymbol{d}_{\lambda_x}^T \boldsymbol{f}_\Lambda + \boldsymbol{d}_{\lambda_y}^T \boldsymbol{g}_\Lambda \quad , \tag{37}$$

with

$$\begin{aligned}
\Theta_\Lambda &:= (\boldsymbol{d}_y - \boldsymbol{d}_z)^T \boldsymbol{Q}_\Lambda (\boldsymbol{d}_y - \boldsymbol{d}_z) + \boldsymbol{d}_w^T \boldsymbol{R}_{w_\Lambda} \boldsymbol{d}_w + \boldsymbol{d}_v^T \boldsymbol{R}_{v_\Lambda} \boldsymbol{d}_v \quad , \\
\boldsymbol{f}_\Lambda &:= \boldsymbol{X}_\Lambda \boldsymbol{d}_x - \boldsymbol{B}_\Lambda \boldsymbol{d}_u - \boldsymbol{W}_\Lambda \boldsymbol{d}_w \quad , \\
\boldsymbol{g}_\Lambda &:= \boldsymbol{I}_\Lambda \boldsymbol{d}_y - \boldsymbol{C}_\Lambda \boldsymbol{d}_x - \boldsymbol{D}_\Lambda \boldsymbol{d}_u - \boldsymbol{V}_\Lambda \boldsymbol{d}_v \quad .
\end{aligned}$$

and $\boldsymbol{d}_{\lambda_x}, \boldsymbol{d}_{\lambda_y}$ denoting the discrete Lagrange multipliers. Differentiation with respect to the unknowns yields the *first order (Kuhn–Karush–Tucker) conditions*:

$$\begin{aligned}
\boldsymbol{X}_\Lambda^T \boldsymbol{d}_{\lambda_x} - \boldsymbol{C}_\Lambda^T \boldsymbol{d}_{\lambda_y} &= 0 \quad , & (38)\\
2\boldsymbol{Q}_\Lambda \boldsymbol{d}_y + \boldsymbol{I}_\Lambda^T \boldsymbol{d}_{\lambda_y} &= 2\boldsymbol{Q}_\Lambda \boldsymbol{d}_z \quad , & (39)\\
2\boldsymbol{R}_{v_\Lambda} \boldsymbol{d}_v - \boldsymbol{V}_\Lambda^T \boldsymbol{d}_{\lambda_y} &= 0 \quad , & (40)\\
2\boldsymbol{R}_{w_\Lambda} \boldsymbol{d}_w - \boldsymbol{W}_\Lambda^T \boldsymbol{d}_{\lambda_x} &= 0 \quad , & (41)\\
\boldsymbol{X}_\Lambda \boldsymbol{d}_x - \boldsymbol{B}_\Lambda \boldsymbol{d}_u - \boldsymbol{W}_\Lambda \boldsymbol{d}_w &= 0 \quad , & (42)\\
\boldsymbol{I}_\Lambda \boldsymbol{d}_y - \boldsymbol{C}_\Lambda \boldsymbol{d}_x - \boldsymbol{D}_\Lambda \boldsymbol{d}_u - \boldsymbol{V}_\Lambda \boldsymbol{d}_v &= 0 \quad . & (43)
\end{aligned}$$

Similar equations can be derived if we first formulate the KKT conditions for the continuous problem followed by the discretization of the arising boundary value problem.

We are interested in the computation of the gradients of the Lagrange functional with respect to the inputs. Suppose that we have fixed approximations

$$\begin{aligned}
\boldsymbol{w}'_\Lambda &= \boldsymbol{d}'^T_w \boldsymbol{\Psi}_\Lambda \quad , & (44)\\
\boldsymbol{v}'_\Lambda &= \boldsymbol{d}'^T_v \boldsymbol{\Psi}_\Lambda \quad . & (45)
\end{aligned}$$

Further let Π be an index set containing the indices (j', k') of basis functions $\psi_{j',k'}$ chosen in the sense of potential candidates which are not yet contained in Λ, i.e. $\Pi \cap \Lambda = 0$. We keep the associated expansion coefficients $\boldsymbol{d}_{w_\Pi}$ and $\boldsymbol{d}_{v_\Pi}$ at zero and *formally* augment the representation of the input functions such that

$$\begin{aligned}
\boldsymbol{w}'_\Lambda &= \boldsymbol{w}'_\Lambda + \boldsymbol{d}_{w_\Pi}^T \boldsymbol{\Psi}_\Pi \quad \text{with} \quad \boldsymbol{d}_{w_\Pi}^T = 0 \quad , & (46)\\
\boldsymbol{v}'_\Lambda &= \boldsymbol{v}'_\Lambda + \boldsymbol{d}_{v_\Pi}^T \boldsymbol{\Psi}_\Pi \quad \text{with} \quad \boldsymbol{d}_{v_\Pi}^T = 0 \quad . & (47)
\end{aligned}$$

We wish to determine the variation of the Lagrange formula over the candidate directions $\boldsymbol{\Psi}_\Pi, \boldsymbol{\Psi}_\Pi$ at $\boldsymbol{w}'_\Lambda, \boldsymbol{v}'_\Lambda$. Before we derive the variational equations we reformulate the discretized optimization problem in its augmented form:

$$\begin{aligned}
\min \quad & ((\boldsymbol{d}_y - \boldsymbol{d}_z)^T \boldsymbol{Q}_\Lambda (\boldsymbol{d}_y - \boldsymbol{d}_z) + \boldsymbol{d}'^T_w \boldsymbol{R}_{w_\Lambda} \boldsymbol{d}'_w + \boldsymbol{d}'^T_v \boldsymbol{R}_{v_\Lambda} \boldsymbol{d}'_v + \\
& + \boldsymbol{d}_{w_\Pi}^T \boldsymbol{R}_{w_\Pi} \boldsymbol{d}_{w_\Pi} + \boldsymbol{d}_{v_\Pi}^T \boldsymbol{R}_{v_\Pi} \boldsymbol{d}_{v_\Pi} + \\
& \boldsymbol{d}_{w_\Pi}^T \boldsymbol{R}_{w_{\Pi,\Lambda}} \boldsymbol{d}'_w + \boldsymbol{d}'^T_w \boldsymbol{R}_{w_{\Lambda,\Pi}} \boldsymbol{d}_{w_\Pi} + \boldsymbol{d}_{v_\Pi}^T \boldsymbol{R}_{v_{\Pi,\Lambda}} \boldsymbol{d}'_v + \boldsymbol{d}'^T_v \boldsymbol{R}_{v_{\Lambda,\Pi}} \boldsymbol{d}_{v_\Pi})
\end{aligned} \tag{48}$$

subject to the constraints

$$\begin{aligned}
\boldsymbol{X}_\Lambda \boldsymbol{d}_x &= \boldsymbol{B}_\Lambda \boldsymbol{d}_u + \boldsymbol{W}_\Lambda \boldsymbol{d}'_w + \boldsymbol{W}_\Pi \boldsymbol{d}_{w_\Pi} \quad , & (49)\\
\boldsymbol{I}_\Lambda \boldsymbol{d}_y &= \boldsymbol{C}_\Lambda \boldsymbol{d}_x + \boldsymbol{D}_\Lambda \boldsymbol{d}_u + \boldsymbol{V}_\Lambda \boldsymbol{d}'_v + \boldsymbol{V}_\Pi \boldsymbol{d}_{v_\Pi} \quad . & (50)
\end{aligned}$$

The discretization matrices associated with the index set Π are defined by

$$
\begin{aligned}
(\boldsymbol{R}_{v\Pi})_{i,j} &= \boldsymbol{R}_{v_{i,j}}\langle\langle \boldsymbol{\Psi}_{\Pi_{v_i}}, \boldsymbol{\Psi}_{\Pi_{v_j}}\rangle\rangle \quad , & (51)\\
(\boldsymbol{R}_{v\Pi,\Lambda})_{i,j} &= \boldsymbol{R}_{v_{i,j}}\langle\langle \boldsymbol{\Psi}_{\Pi_{v_i}}, \boldsymbol{\Psi}_{\Lambda_{v_j}}\rangle\rangle \quad , & (52)\\
(\boldsymbol{R}_{v\Lambda,\Pi})_{i,j} &= \boldsymbol{R}_{v_{i,j}}\langle\langle \boldsymbol{\Psi}_{\Lambda_{v_i}}, \boldsymbol{\Psi}_{\Pi_{v_j}}\rangle\rangle \quad , & (53)\\
(\boldsymbol{R}_{w\Pi})_{i,j} &= \boldsymbol{R}_{w_{i,j}}\langle\langle \boldsymbol{\Psi}^H_{\Pi_{w_i}}, \boldsymbol{\Psi}^H_{\Pi_{w_j}}\rangle\rangle \quad , & (54)\\
(\boldsymbol{R}_{w\Pi,\Lambda})_{i,j} &= \boldsymbol{R}_{w_{i,j}}\langle\langle \boldsymbol{\Psi}^H_{\Pi_{w_i}}, \boldsymbol{\Psi}^H_{\Lambda_{w_j}}\rangle\rangle \quad , & (55)\\
(\boldsymbol{R}_{w\Lambda,\Pi})_{i,j} &= \boldsymbol{R}_{w_{i,j}}\langle\langle \boldsymbol{\Psi}^H_{\Lambda_{w_i}}, \boldsymbol{\Psi}^H_{\Pi_{w_j}}\rangle\rangle \quad , & (56)\\
(\boldsymbol{W}_\Pi)_{i,j} &= \boldsymbol{W}_{i,j}\langle\langle \boldsymbol{\Psi}^H_{\Lambda'_{x_i}}, \boldsymbol{\Psi}^H_{\Pi_{w_j}}\rangle\rangle \quad , & (57)\\
(\boldsymbol{V}_\Pi)_{i,j} &= \boldsymbol{V}_{i,j}\langle\langle \tilde{\boldsymbol{\Psi}}_{\Lambda_{y_i}}, \boldsymbol{\Psi}_{\Pi_{v_j}}\rangle\rangle \quad . & (58)
\end{aligned}
$$

It is clear that the solution of problem (48) is equal to the solution determined by equation system (51)- (58) if the assumption $\boldsymbol{d}_{w\Pi}, \boldsymbol{d}_{vPi} = \boldsymbol{0}$ holds. Suppose, we already have valid solutions $\boldsymbol{d}_x, \boldsymbol{d}_y, \boldsymbol{d}_{\lambda_x}, \boldsymbol{d}_{\lambda_y}$ for $\boldsymbol{d}'_w, \boldsymbol{d}'_v$ which satisfy part of the KKT conditions, i.e. (38), (39), (42), (43) for an arbitrary but fixed choice $\boldsymbol{w}'_\Lambda, \boldsymbol{v}'_\Lambda$. In that special case $\mathcal{L}_{\Lambda\cup\Pi} = \Theta_{\Lambda\cup\Pi}$ where $\mathcal{L}_{\Lambda\cup\Pi}$ and $\Theta_{\Lambda\cup\Pi}$ denote the augmented Lagrangian and augmented costfunctiona. Furthermore the partial derivatives fulfill

$$
\frac{\partial \mathcal{L}_{\Lambda\cup\Pi}}{\partial \boldsymbol{d}_x}|_{w',v'} = 0, \quad \frac{\partial \mathcal{L}_{\Lambda\cup\Pi}}{\partial \boldsymbol{d}_y}|_{w',v'} = 0, \quad \frac{\partial \mathcal{L}_{\Lambda\cup\Pi}}{\partial \boldsymbol{d}_{\lambda_x}}|_{w',v'} = 0, \quad \frac{\partial \mathcal{L}_{\Lambda\cup\Pi}}{\partial \boldsymbol{d}_{\lambda_y}}|_{w',v'} = 0, \tag{59}
$$

The gradients of the Lagrange functional with respect to $\boldsymbol{d}_{w\Pi}, \boldsymbol{d}_{v\Pi}$ are given by

$$
\begin{aligned}
\frac{\partial \mathcal{L}_{\Lambda\cup\Pi}}{\partial \boldsymbol{d}_{w\Pi}}|_{w',v'} &= \frac{\partial \Theta_{\Lambda\cup\Pi}}{\partial \boldsymbol{d}_{w\Pi}}|_{w',v'} + \boldsymbol{d}^T_{\lambda_x}\frac{\partial \boldsymbol{f}_{\Lambda\cup\Pi}}{\partial \boldsymbol{d}_{w\Pi}}|_{w',v'} \quad , & (60)\\
\frac{\partial \mathcal{L}_{\Lambda\cup\Pi}}{\partial \boldsymbol{d}_{v\Pi}}|_{w',v'} &= \frac{\partial \Theta_{\Lambda\cup\Pi}}{\partial \boldsymbol{d}_{v\Pi}}|_{w',v'} + \boldsymbol{d}^T_{\lambda_y}\frac{\partial \boldsymbol{g}_{\Lambda\cup\Pi}}{\partial \boldsymbol{d}_{v\Pi}}|_{w',v'} \quad . & (61)
\end{aligned}
$$

Note, that this result remains valid also for optimal $\boldsymbol{w}', \boldsymbol{v}'$ satisfying (40), (41). The sensitivity equations are well known and commonly applied to compute a search direction needed in NLP algorithms (see e.g. [6],[19]). They are quite often referred to as the sensitivities of the cost functional with regard to $\boldsymbol{d}_{w\Pi}, \boldsymbol{d}_{v\Pi}$ since $\mathcal{L}_{\Lambda\cup\Pi} = \Theta_{\Lambda\cup\Pi}$. In the sequel we refer also to the sensitivity of the Lagrange functional by the *sensitivity or gradients* of the cost functional.

In particular, the gradients in our specific example are

$$
\begin{aligned}
\frac{\partial \mathcal{L}_{\Lambda\cup\Pi}}{\partial \boldsymbol{d}_{w\Pi}}|_{w',v'} &= \boldsymbol{R}_{w\Pi,\Lambda}\boldsymbol{d}'_w + \boldsymbol{d}'^T_w\boldsymbol{R}_{w\Lambda,\Pi} - \boldsymbol{d}^T_{\lambda_x}\boldsymbol{W}_\Pi \quad , & (62)\\
\frac{\partial \mathcal{L}_{\Lambda\cup\Pi}}{\partial \boldsymbol{d}_{v\Pi}}|_{w',v'} &= \boldsymbol{R}_{v\Pi,\Lambda}\boldsymbol{d}'_v + \boldsymbol{d}'^T_v\boldsymbol{R}_{v\Lambda,\Pi} - \boldsymbol{d}^T_{\lambda_y}\boldsymbol{V}_\Pi \quad . & (63)
\end{aligned}
$$

Once the discretization matrices are available, the evaluation of the gradients requires only a simple matrix vector multiplication of the discretization matrices with the already available Lagrange multipliers $\boldsymbol{d}_{\lambda_x}, \boldsymbol{d}_{\lambda_y}$ and solutions $\boldsymbol{d}'_w, \boldsymbol{d}'_v$ and is therefore very inexpensive. For linear systems the construction of the discretization matrices is not a problem since the necessary inner products can be precomputed and stored appropiately. The specifically needed matrices can be assembled from such stored inner products in a straight forward manner. The index sets Π_w, Π_v should posses full row rank otherwise the sensitivites in the associated rows will be zero. This can be guaranteed if $\Pi_w \subseteq \Lambda_x, \Pi_v \subseteq \Lambda_y$, since then each row of the inner products in the matrices $\boldsymbol{W}_\Pi, \boldsymbol{V}_\Pi$ have at least one entrie which is different from zero.

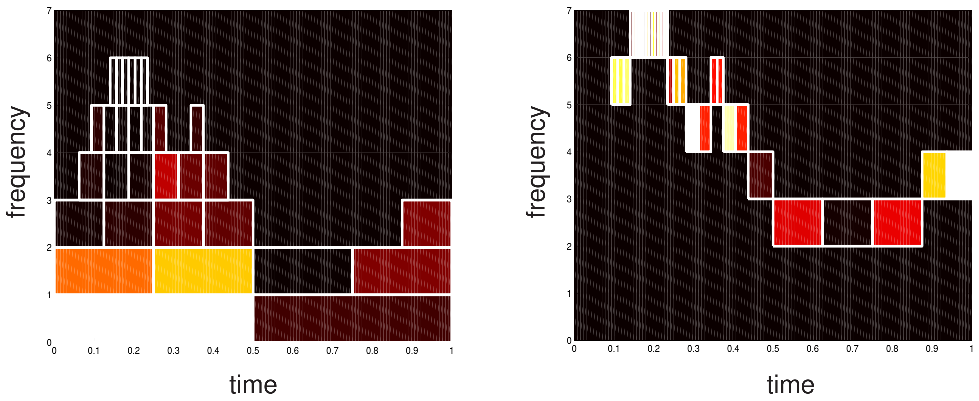

Figure 4: Time-frequency plot of wavelet coefficients $\boldsymbol{d}_w$ (left) and the associated surrounding layer (right) at some refinement stage ℓ for w_{Λ_ℓ} depicted in figure 12

5.3.2 Choice of refinement sets Γ'_w, Γ'_v

So far, we have derived the equations for the evaluations of the sensitivities. In a next step we have to choose the candidate sets $\Pi_{w_\ell}, \Pi_{v_\ell}$. Recall that we do have valid solutions to the optimization problem in $\boldsymbol{\Upsilon}_{\Lambda_\ell}$. The hierachical structure of the wavelets together with smoothness assumptions on the solution suggests that the local neighbors of the trial functions in $\Lambda_{w_\ell}, \Lambda_{v_\ell}$ are excellent candidates for potential refinement. As local neighbors of a trial function $\psi_{j,k}$ we refer to $\psi_{j,k-1}, \psi_{j,k+1}$ as well as trial functions of the next higher scale $\psi_{j+1,2k}, \psi_{j+1,2k+1}$. All local neighbors of an index set $\Lambda_{w_\ell}, \Lambda_{v_\ell}$ which are not contained in $\Lambda_{w_\ell}, \Lambda_{v_\ell}$ are collected in $\Pi_{w_\ell}, \Pi_{v_\ell}$, referred to as the so called *surrounding layer* of $\Psi_{\Lambda_w}, \Psi_{\Lambda_v}$. A typical example of the surrounding layer is depictured in figure 4*. Note that the definition of neighbors presented is somehow arbitrary. It can be easily modified such that the surrounding layer comprises more than one scale.

Now that we have chosen the potential index sets $\Pi_{w_\ell}, \Pi_{v_\ell}$ we compute the gradients

$$\frac{\partial \mathcal{L}_\Lambda}{\partial \boldsymbol{d}_{w_\Pi}}\,|_{w',v'} \quad , \frac{\partial \mathcal{L}_\Lambda}{\partial \boldsymbol{d}_{v_\Pi}}\,|_{w',v'} \tag{64}$$

according to (62),(63). Recall that this evaluation involves only simple matrix vector multiplications of the discretization matrices associated to the potential trial functions $\Psi_{\Pi_w}, \Psi_{\Pi_v}$ with the solution vectors $\boldsymbol{d}'_w, \boldsymbol{d}'_v$ and the the Lagrange multipliers $\boldsymbol{d}_{\lambda_x}, \boldsymbol{d}_{\lambda_y}$.

In analogy to the previous reasoning we determine the largest absolute entries of the vectors $\frac{\partial \mathcal{L}_\Lambda}{\partial \boldsymbol{d}_{w_\Pi}}\,|_{w',v'}, \frac{\partial \mathcal{L}_\Lambda}{\partial \boldsymbol{d}_{v_\Pi}}\,|_{w',v'}$. Since the associated trial functions have a high impact on the cost functional they are collected in $\Gamma'_{w_\ell}, \Gamma'_{v_\ell}$. As before the number of collected trial functions is either fixed or depends on some threshold and therefore controls either numerical complexity or approximation quality.

The gradient approach has been presented so far for equality constraints only. If we compute the gradients based on the equality restrictions while inequalities are also present, it might very well be the case that we have large gradients in areas where inequality restrictions are active. The refinement will most probably not increase the approximation quality since the active inequalities already bound the solution. Since we do have information from the current solutions $\boldsymbol{d}_{w_\ell}, \boldsymbol{d}_{v_\ell}$ which inequalities are active we can use a simple heuristic that excludes refinement at active boundaries. Mesh points which are embedded by a specified number of active inequalities from both sides are excluded from potential refinement.

*Here 2 trial functions are not shown. The reason will be explained at the end of this section.

5.4 Interaction

So far we have determined the refinement sets $\Gamma'_{z_\ell}, \Gamma'_{u_\ell}, \Gamma'_{x_\ell}, \Gamma'_{y_\ell}, \Gamma'_{w_\ell}, \Gamma'_{v_\ell}$. As already indicated there are dependencies which have to be considered as well. First the gradient evaluation of $\boldsymbol{w}, \boldsymbol{v}$ requires that the tested trial basis functions are also contained in the discretization set of $\boldsymbol{x}, \boldsymbol{y}$. Otherwise the gradients are zero due to singular rows of the matrices $\boldsymbol{W}_\Pi, \boldsymbol{V}_\Pi$. As a consequence, the set Γ_{x_ℓ} has to contain the surrounding layer $\Pi_{w_{\ell+1}}$ of Γ_{w_ℓ} such that the sensitivities can be evaluated in the step $\ell + 1$. The same relation holds for Γ_{y_ℓ} and $\Pi_{v_{\ell+1}}$.

Second, since the costfunctional envolves numerical evaluation of $\boldsymbol{d}_z - \boldsymbol{d}_y$, the index set of the measurement function z_i and the model output y_i are assumed to be the same. Summarizing the interactions we obtain

$$
\begin{aligned}
\Gamma_{w_\ell} &:= \Gamma'_{w_\ell} & \Gamma_{v_\ell} &:= \Gamma'_{v_\ell} \\
\Gamma_{u_\ell} &:= \Gamma'_{u_\ell} & \Gamma_{x_\ell} &:= \Gamma'_{x_\ell} \cup \Pi_{w_{\ell+1}} \cup \Pi_{v_{\ell+1}} \cup \Gamma'_{u_\ell} \cup \Gamma'_{z_\ell} \quad , \\
\Gamma_{y_\ell} &:= \Gamma_{x_\ell} & \Gamma_{z_\ell} &:= \Gamma_{y_\ell} \quad .
\end{aligned}
$$

Finally the index set used in the refined step are

$$
\begin{aligned}
\Lambda_{x_{\ell+1}} &:= \Lambda_{x_\ell} \cup \Gamma_{x_\ell} \quad , & \Lambda_{y_{\ell+1}} &:= \Lambda_{y_\ell} \cup \Gamma_{y_\ell} \quad , \\
\Lambda_{w_{\ell+1}} &:= \Lambda_{w_\ell} \cup \Gamma_{w_\ell} \quad , & \Lambda_{v_{\ell+1}} &:= \Lambda_{v_\ell} \cup \Gamma_{v_\ell} \quad , \\
\Lambda_{z_{\ell+1}} &:= \Lambda_{z_\ell} \cup \Gamma_{z_\ell} \quad , & \Lambda_{u_{\ell+1}} &:= \Lambda_{u_\ell} \cup \Gamma_{u_\ell} \quad .
\end{aligned}
$$

In a last step completeness of their tree structures is checked. In case the tree structure is violated the missing trial functions are filled in. At first glance the refinement set Γ_x appears to be a large set. But one has to realize that the sets typically have a fair amount of intersection among each other such that the update is rather moderate. Relaxation of the assumption that all states and outputs are discretized on the same mesh might lower the computational cost. This option will be investigated further in future.

6 Algorithmic outline

The specific details for the generation of the refinement sets have been presented in the last section. Now we summarize the refinement algorithm. The general outline for a fixed horizon is shown in figure 5.

Given the measurement functions $\boldsymbol{z}, \boldsymbol{u}$ as well as initial index sets $\Lambda_{\xi,0}$, $\xi \in \{x, y, w, u, z\}$, which for convenience are chosen to be equidistant, we assemble the discretization matrices needed to formulate the QP. Next, the QP is solved either by available general purpose solvers off the shelf or alternatively by problem adapted methods which account for the special problem structure. In general, we prefer the latter since our solution methodology requires the exploitation of warm start potential to reduce the numerical cost. In particular, we are currently investigating tailored solvers which are based on iterative methods for the treatment of the linear algebra. It is beyond the scope of this paper to discuss the adapted solver, instead we refer to [5]. In this paper we employ the QP solver SQOPT [17], a black-box method to solve large sparse problems, which is based on an active set strategy.

SQOPT generates the primal solution $\boldsymbol{d}_{x_\ell}, \boldsymbol{d}_{y_\ell}, \boldsymbol{d}_{w_\ell}, \boldsymbol{d}_{v_\ell}$ as well as the Lagrange multipliers $\boldsymbol{d}_{x_{\lambda_\ell}}, \boldsymbol{d}_{y_{\lambda_\ell}}$. These solutions together with the current approximations $\boldsymbol{d}_{z_\ell}, \boldsymbol{d}_{u_\ell}$ are used to compute the refinement sets $\Gamma_{z_\ell}, \Gamma_{u_\ell}, \Gamma_{x_\ell}, \Gamma_{y_\ell}, \Gamma_{w_\ell}, \Gamma_{v_\ell}$ as outlined in Section 5.3.1. Further, the error estimates and the sensitivities of the cost functional on which the assembling of the refinement sets is based, serve as stopping criteria for the refinement sequence. If the approximation accuracy does not fulfill a pre-specified tolerance the refinement process is continued.

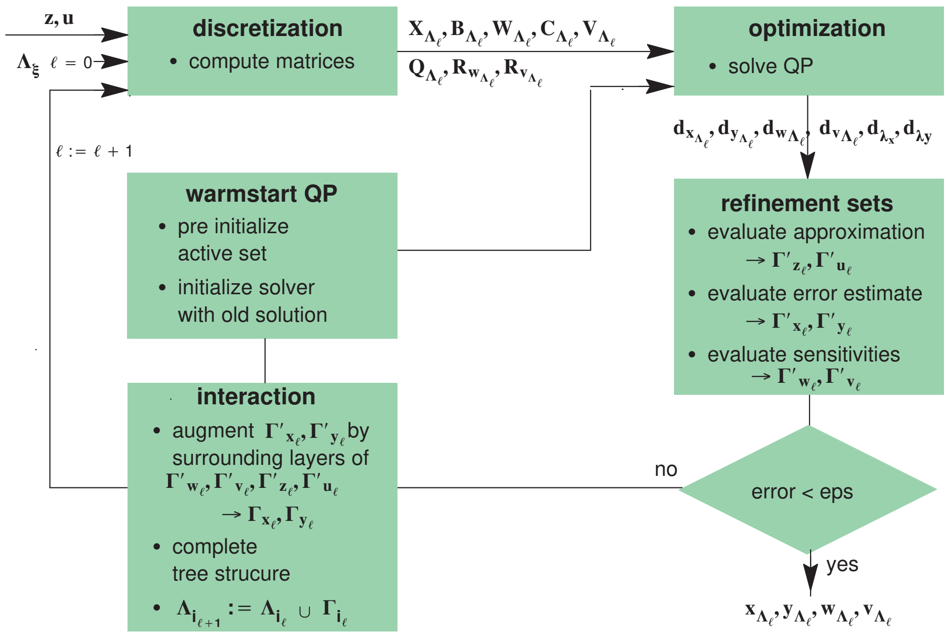

Figure 5: Outline of the adaptive refinement strategy

Interactions among the sets computed so far are taken into account. The structure of the sets is analyzed and if necessary the sets are filled with additional basis functions such that a complete tree structure is formed. The refinement set together with the current discretization set form the set used for the next discretization $\Lambda_{\ell+1} = \Lambda_\ell \cup \Gamma_\ell$. Finally the refinement loop counter is raised by one $\ell := \ell + 1$ and the discretization matrices are again assembled.

As already indicated in the introduction there are several layers of information which might be reused in the current refinement cycle to reduce the numerical cost of the algorithm. Among others, the assembling of the discretization matrices can be simplified by reusing the discretization matrices, e.g. the Hessian as well as the Jacobian, obtained at stage $\ell - 1$. Furthermore the QP solver can be pre-initialized with an active set as well as the old solution approximation. Details on the reuse of information for warm starts are given in [4, 5].

7 Case Study

The adaptive refinement strategy is applied to two small examples. In the first example we are interested to estimate unknown input function besides estimating the states whereas the task of the second example is pure state estimation.

7.1 Example 1

First we consider an illustrative toy example

$$\min \int_0^1 (y - z)^2 + \varepsilon w^2 d\tau \tag{65}$$

subject to the constraints

$$\begin{aligned} \dot{\boldsymbol{x}} &= \boldsymbol{A}\boldsymbol{x} + \boldsymbol{W}w && (66)\\ y &= \boldsymbol{C}\boldsymbol{x} \quad , && (67) \end{aligned}$$

$$x_1 \leq 1 \quad (68)$$
$$-10 \leq w \leq 10 \quad (69)$$

where

$$\boldsymbol{A} = \begin{bmatrix} -2.7860 & -3.9070 \\ 3.9070 & 0 \end{bmatrix}, \quad W = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \quad C^T = \begin{bmatrix} 0 \\ 1 \end{bmatrix}. \quad (70)$$

The nominal solutions $\tilde{\boldsymbol{x}}, \tilde{y}$, on which the estimation example is based upon are computed by integrating the model equations (66),(67) with a standard Runge–Kutta method starting from the initial condition $\boldsymbol{x}(0)^T = [0,0]^T$. As nominal input function $\tilde{w}$ we took a deterministic function which is sampled on a mesh with $h = 2^{-8}$. The last curves of figures 7 - 8 depict the time evolution of $\tilde{\boldsymbol{x}}, \tilde{y}, \tilde{w}$. Further we take the nominal solution as measurement samples $z(t_i) = \tilde{y}(t_i)$ and interpolate the discrete values to obtain a continuous measurement function. No measurement noise is present.

Given the measurement function z, we have to reconstruct the initial condition $\boldsymbol{x}(0)$ as well as the time evolution of $\boldsymbol{x}, y, w$ which should turn out to converge to the nominal solutions. As we will see later on, the nominal solutions cannot be reconstructed since we have chosen a cost functional (65) where w is regularized with $\varepsilon = 10^{-5}$. If ε is chosen to be zero, w appears only linearly in the Lagrangian and yields a bang bang solution type [7]. Especially if measurement noise were present (which again is not the case in this example) regularization is necessary for a reduction of the variance of the solution. On the other side, regularization introduces also a bias of the estimated solutions compared to the nominal one. Since $\tilde{w} \neq 0$ the estimated w_Λ will not converge to the nominal solution. To establish such convergence we would have to regularize the optimization problem with $\varepsilon(\tilde{w} - w_\Lambda)$. However, this is impractical since $\tilde{w}$ usually is not known.

The estimation problem is solved starting from a coarse equidistant mesh with $h = 2^{-4}$ for $\boldsymbol{x}, y$ and $h = 2^{-3}$ for w. The approximation quality of the discretized optimization problem is increased on the basis of scalewise and adaptive refinement. In the particular example we employ two adaptive refinement strategies based on the sensitivity evaluation and a neighbor detection method.

The sensitivity method evaluates potential additional trial functions of the input w on the basis of their gradients with respect to the cost functional. The representation of w is refined taking n trial functions into account which are associated with the largest gradients. In the example we chose $n = 1$ which yields the smallest possible refinement set Γ_w. In case the complete tree assumption is violated for a specific $\Lambda_w \cup \Gamma_w$ we add additional trial functions to complete the tree structure. The neighbor based method searches the current solution vector $| \boldsymbol{d}_{w_\Lambda} |$ for the n largest coefficients and takes their local neighbors without additional tests into account. Like before we chose $n = 1$. Since one trial function can have up to 4 local neighbors the cardinality of the refinement set $1 \leq \sharp\Gamma_w \leq 4$. Again, deficiencies in the tree structure will be corrected afterwards. Finally the scalewise method takes all trial functions of the next higher scale into account. It corresponds to a dyadic subdivision of the mesh.

Refinement of w_Λ will automatically refine $\boldsymbol{x}_\Lambda, y_\Lambda$ through the established interaction (see Section 5.4). In the specific example refinement of $\boldsymbol{x}_\Lambda, y_\Lambda$ on basis of local error estimation leads to similar sets since there are no driving functions $\boldsymbol{u}$ present and all of the states reflect changes in the input w. Refinement based on local error estimation of $\boldsymbol{x}$ is therefore not needed.

Figures 6 - 8 show $\boldsymbol{x}_{\Lambda_\ell}, y_{\Lambda_\ell}, w_{\lambda_\ell}, \ell = \{0, 4, 8, 12, 16, 20, 24\}$ which are computed with the aid of the gradient method. Furthermore the figures reveal the nominal solutions (last curve). The curves of scalewise and neighbor based refinement are not shown here but look quite similar. To enable a comparison among the specific values the cost functional Θ_{Λ_ℓ} as well as $\| \tilde{y} - y_{\Lambda_\ell} \|$, $\| \tilde{x}_1 - x_{\Lambda_\ell} \|$, $\| \tilde{w} - w_{\Lambda_\ell} \|$ are plotted in figure 9 as a function of the total number of trial functions $\sharp\Lambda_{x_\ell} + \sharp\Lambda_{y_\ell} + \sharp\Lambda_{w_\ell}$. The same accuracy in Θ_ℓ and $\| \tilde{y} - y_{\Lambda_\ell} \|$ is achieved with less trial functions in the adaptive case. The savings are nice especially for coarse solutions. The gradient method performs slightly better than the neighbor based method. As it can be clearly seen in all cases the computed estimates are biased which is caused by the regularization. An interesting phenomenon can be observed by inspection of $\| \tilde{w} - w_{\Lambda_\ell} \|$. First there is a substantial decrease in the error followed be an increase. With increasing refinement the step size h tends to zero and, as a consequence, the initial values of the Lagrange multipliers converge to zero $\lambda_x(0) \to 0$ since we have not specified initial conditions for the states (see the Appendix for a more detailed discussion with respect to the boundary values for dynamic optimization problems with unspecified initial conditions). This implies $w(0)_\Lambda \to 0$. For large step sizes the estimates remain closer to their nominal values until the solution structure changes (see figure 8).

For comparison we solve a slightly modified estimation problem. All the ingredients of the so far considered problem remain the same, except now we fix the initial conditions of the states by an additional equality constraint. Hence, we switch from an estimation to a control problem with z as the reference input.

The estimated quantities are depicted in figures 10 - 13. The approximation errors verify a reduction in the errors of $w_\Lambda, x1_\Lambda$, although the values of the cost functional Θ_Λ has been increased. The increase of the cost functional is not surprising since we fixed 2 degrees of freedom in the optimization problem. However, one might have expected that the estimation quality of all quantities gets worse if degrees of freedom are fixed. This is due to regularization with the wrong regularization function, in our example the 0 function. Here we only want to illuminate the effect of an incorrect choice of the regularization. However both examples indicate that adaptive refinement should be preferred in any case.

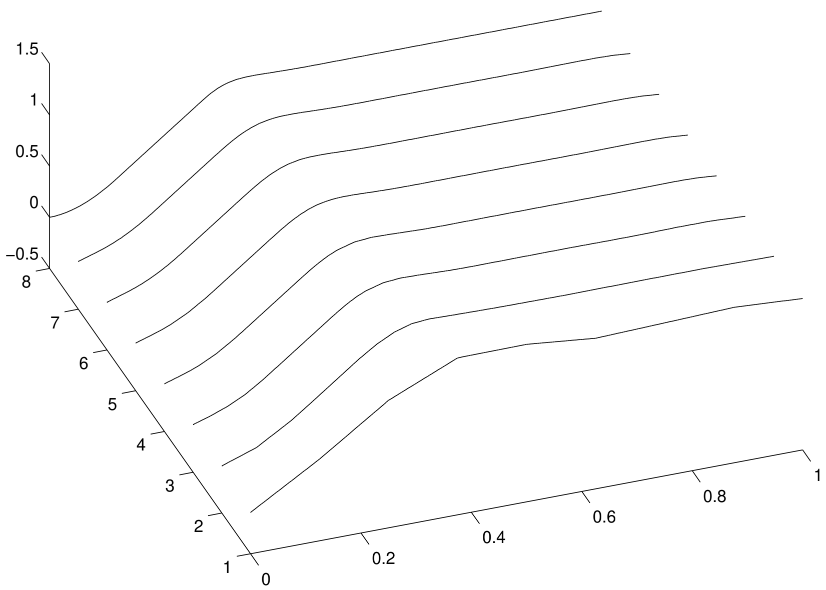

Figure 6: Sequence of refinement for the estimated model output y_Λ. The initial conditions of the states are free. The last curve denotes the reference function y.

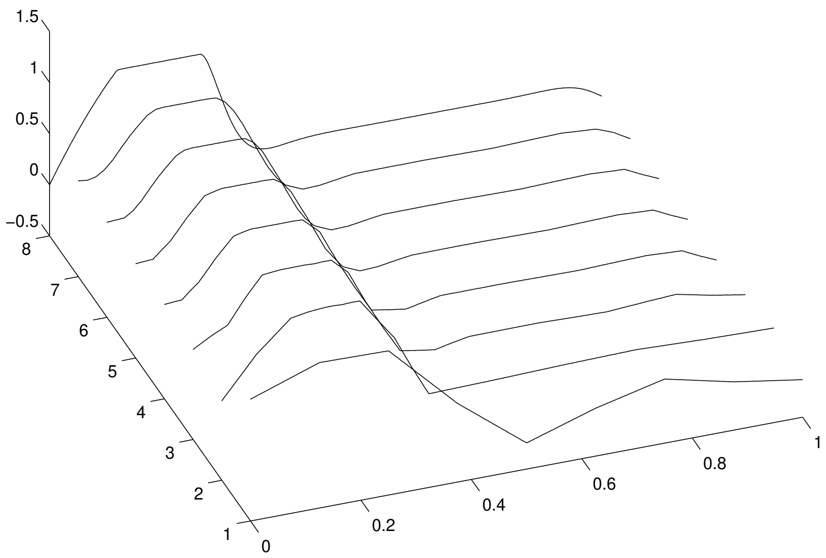

Figure 7: Sequence of refinement for the estimated model state x_{1_Λ}. The initial conditions of the states are free. The last curve denotes the reference function x_1.

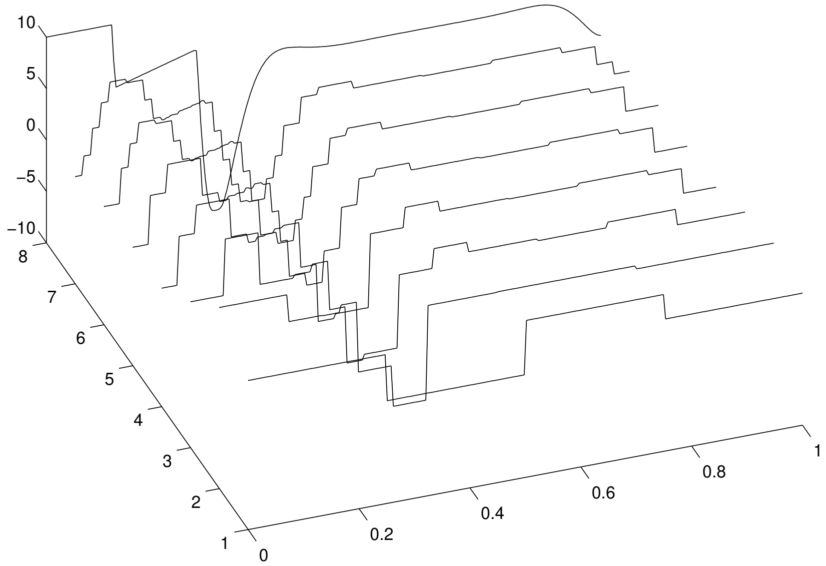

Figure 8: Sequence of refinement for the estimated input w_Λ. The initial conditions of the states are free. The last curve denotes the reference function w.

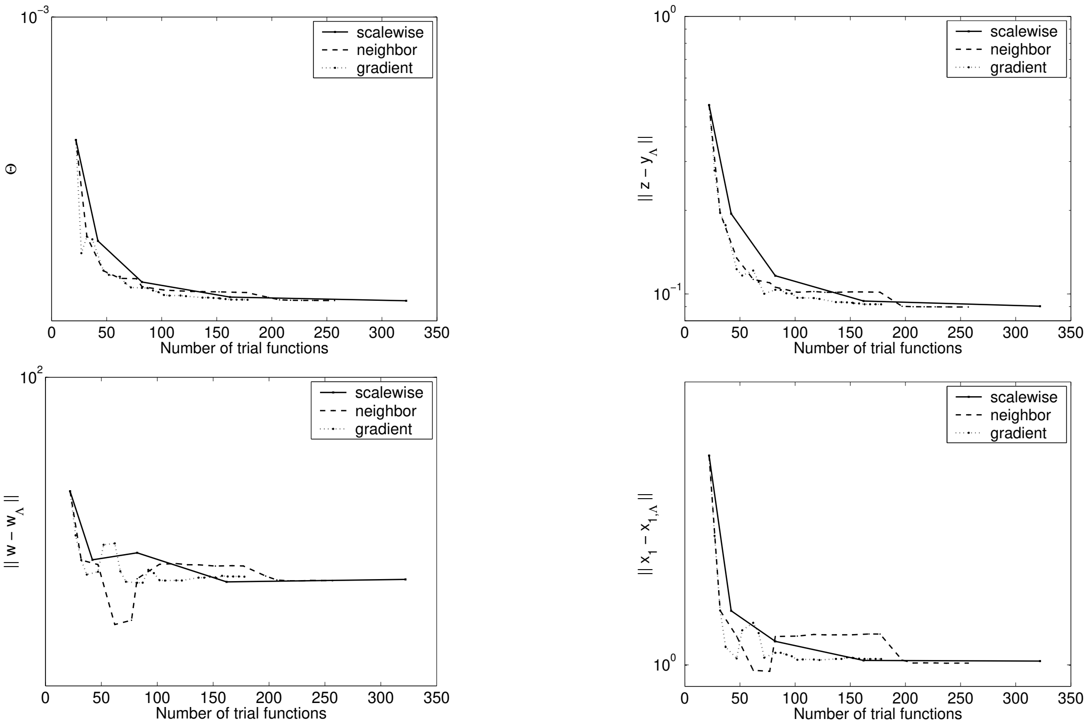

Figure 9: Comparison of scalewise vs. adaptive refinement strategy. The adaptive strategy comprises the gradient based evaluation and the neighbor heuristic. The regularization parameter is $\varepsilon = 10^{-5}$. The initial condition are free

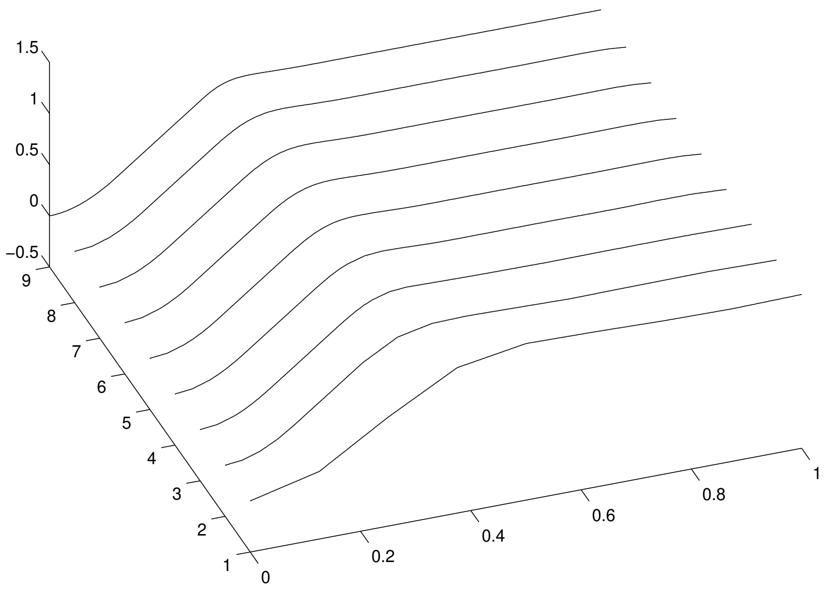

Figure 10: Sequence of refinement for the estimated model output y_Λ. The initial conditions of the states are fixed. The last curve denotes the non estimated reference function y

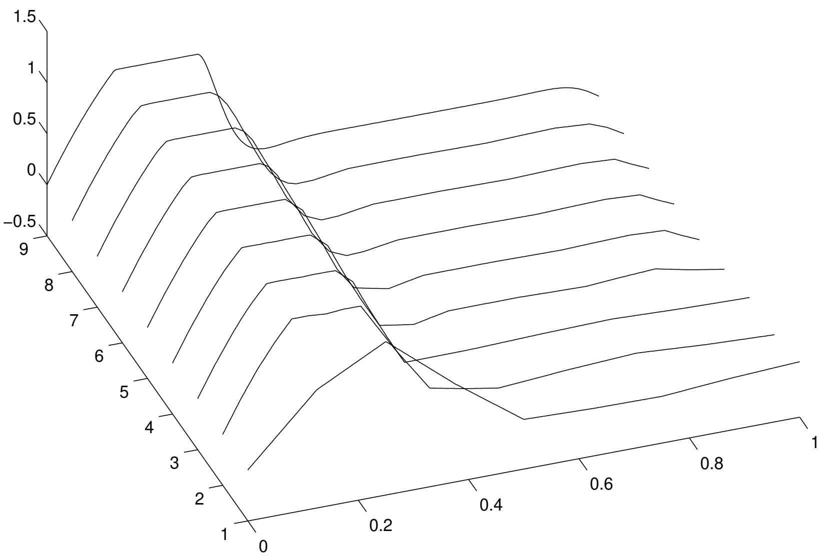

Figure 11: Sequence of refinement for the estimated model state x_{1_Λ}. The initial conditions of the states are fixed. The last curve denotes the non estimated reference function x_1

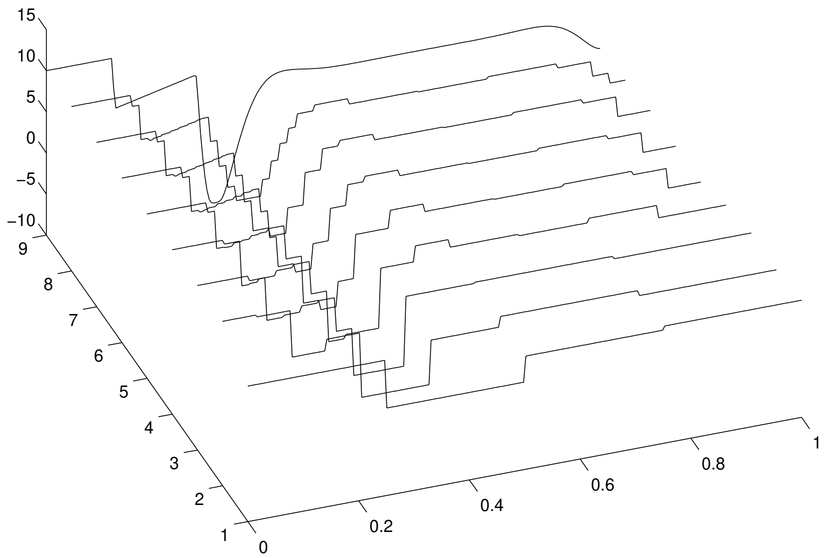

Figure 12: Sequence of refinement for the estimated input w_Λ. The initial conditions of the states are fixed. The last curve denotes the non estimated reference function w

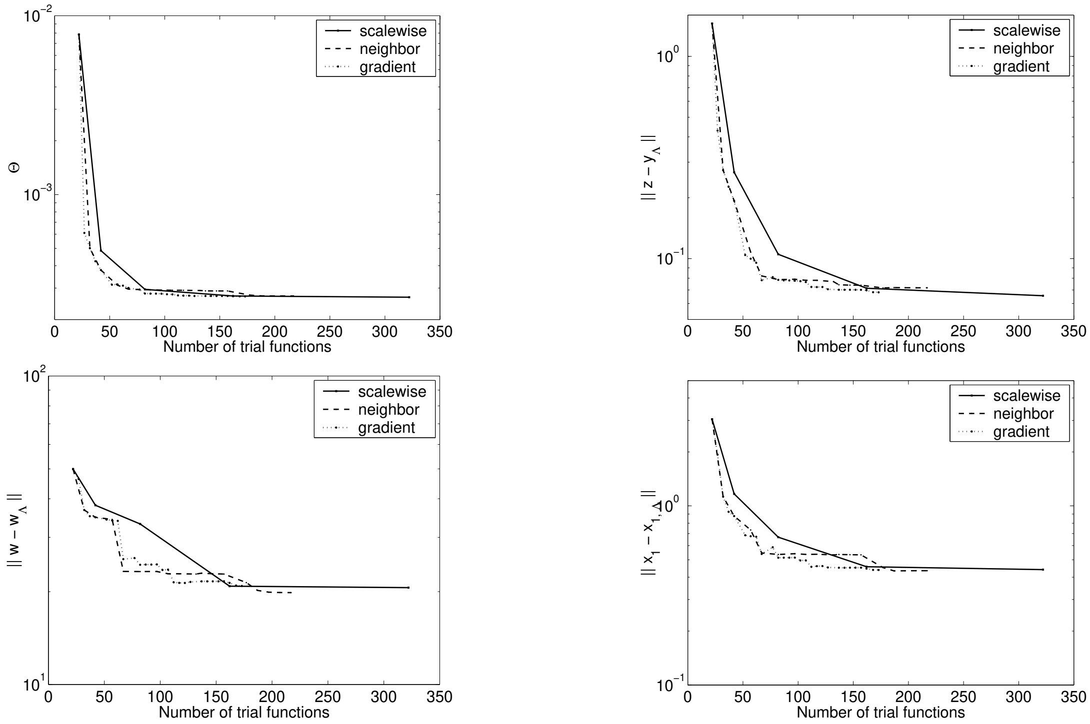

Figure 13: Comparison of scalewise vs. adaptive refinement strategy. The adaptive strategy comprises the gradient based evaluation and the neighbor heuristic. The regularization parameter is $\varepsilon - 10^{-5}$. The initial condition are fixed.

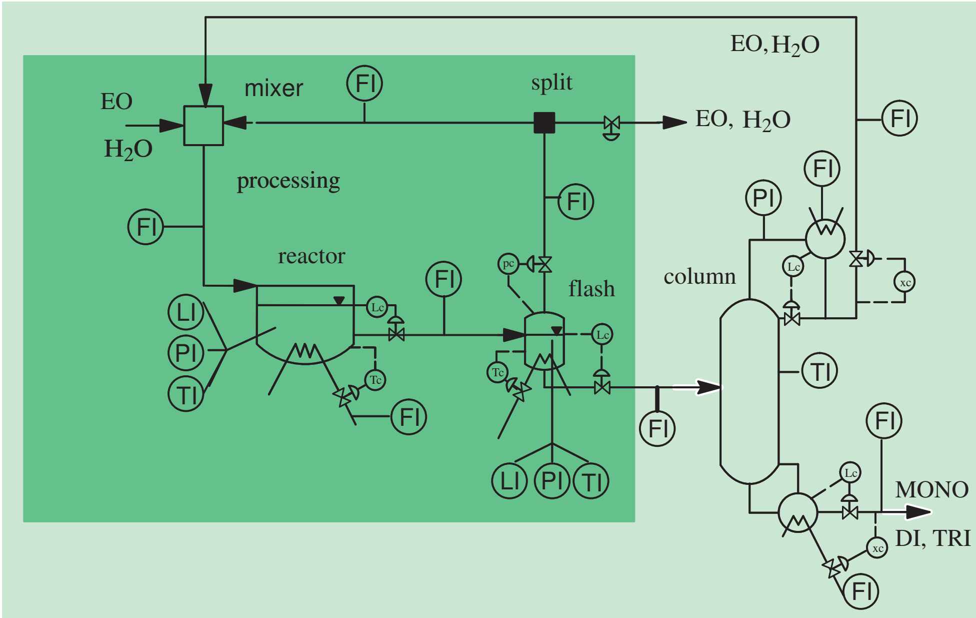

Figure 14: Schematical outline of the ethylene–glycol process

7.2 Example 2

In the next example we consider a linear state estimation problem which is based on a subprocess within the ethylene–glycol process. In particular we focus on process units where the two educts ethylene–oxide and water are converted to the product mono–ethylene–glycol. In addition, the process also reveals unwanted byproducts which in our setting are assumed to be di–ethylene–glycol and tri–ethylene–glycol.

The process units considered in the estimation example are a mixer, a reactor, and a separator (flash). The splitter as well as the distillation column including the outer recycle are neglected. A schematic outline of the process is given in figure 14. Both educts enter the system through a liquid feed stream where they are blended with the recycle stream in the mixing unit. The output of the mixer serves as input stream to the reactor where an exothermic conversion takes place based on the reaction scheme

$$
\begin{aligned}
\textit{ethylene-oxide}(EO) + \textit{water}(H2O) &\rightarrow \textit{mono-ethylene-glycol}(MONO) \quad , \\
\textit{mono-ethylene-glycol}(MONO) + \textit{ethylene-oxide}(EO) &\rightarrow \textit{die-ethylene-glycol}(DI) \quad , \\
\textit{die-ethylene-glycol}(DI) + \textit{ethylene-oxid}(EO) &\rightarrow \textit{tri-ethylene-glycol}(TRI) \quad .
\end{aligned}
$$

The liquid stream, essentially enriched with the product EO is drawn from the reactor and serves as input to the flash. In the flash, the unconverted EO as well as a fair amount of water is separated from the product and leaves the unit as vapor on top and serves as a recycle. The liquid fraction is the desired product stream and is drawn from the bottom. The product stream serves as input for additional processing units. Both streams leaving the flash, the light fraction on top as well as the heavy fraction on bottom, contain all five components. They only differ in composition.

A process model can be derived from first principles, e. g. the formulation of energy and mass conservation laws. The balance equations will yield nonlinear differential algebraic equations (DAEs). In our particular example we assume that the pressure in the reactor is fixed and that we can neglect the gas holdup in the flash. Among others, these assumptions lead to

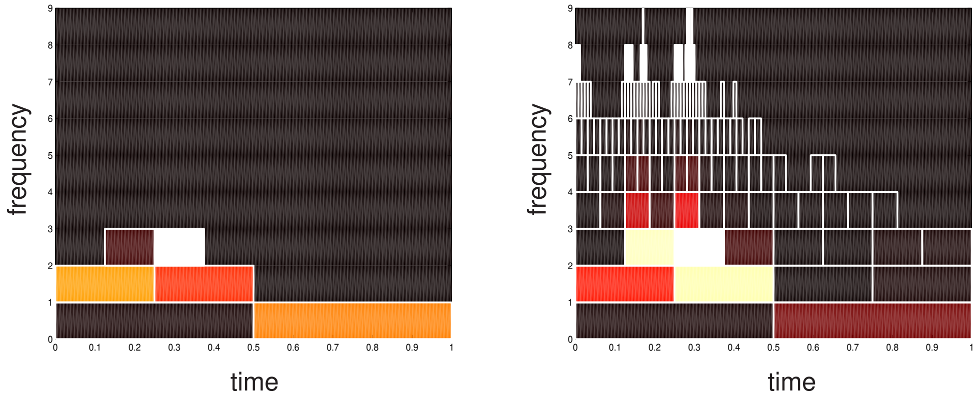

Figure 15: Time-frequency plot of wavelet coefficients $\boldsymbol{d}_{y_4}$ for the starting $\ell = 0$ and final $\ell = 6$ discretization mesh.

a DAEs of index one. The model is based on the formulation given in [24] and is implemented in SPEEDUP, a dynamic simulator package widely use in the chemical engineering community.

For the particular example we are interested in a linear model of the process. SPEEDUP offers the matrices of a linear state space model valid around a stationary point. The algebraic equations of the nonlinear DAEs are already substituted such that a linear model of the form

$$\begin{aligned} \dot{\boldsymbol{x}} &= \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{u} \quad , \\ \boldsymbol{y} &= \boldsymbol{C}\boldsymbol{x} \quad , \end{aligned}$$

is derived. The model considered comprises $n_x = 12$ states, $n_u = 8$ forcing functions and $n_y = 5$ model outputs. The physical interpretation of the states, forcing functions and outputs including the associated units are given in tables (1) -(3). The values of the model matrices $\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}$ can be found in the appendix. The forcing functions are assumed to be known for the estimation of the states. They are generated as follows: $u_1 - u_3$ are considered as disturbances to the process. In the present case study, while u_1, u_2 are kept at zero we assume a disturbance in the feed concentration u_3. As a consequence the process moves away from the steady state. Proportional-integral controllers are assigned to the remaining inputs $u_4 - u_8$ such that the measured quantities will be forced back to their steady state values. The resulting time evolutions of $\boldsymbol{u}$ are depictured as deviation variables in the appendix.

Given those input functions $\boldsymbol{u}$ and the initial condition $\boldsymbol{x}(0) = \boldsymbol{0}$ we compute the nominal solutions $\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{y}}$ on which the estimation example is based upon with a Runge-Kutta-Method. Like in example 1, discrete values of the nominal solution are taken as measurement samples, $\boldsymbol{z}(t_i) = \tilde{\boldsymbol{y}}(t_i)$ and are interpolated to obtain a continuous representation $\boldsymbol{z}$. No measurement noise is present. Thus the task of the estimation is to reconstruct the initial conditions and the associated transients $\boldsymbol{x}, \boldsymbol{y}$.

Since we assume perfect measurements as well as a perfect process model we do not include the input functions $\boldsymbol{w}, \boldsymbol{v}$ into the problem formulation. All quantities in the estimation problem are scaled with the L_2 norm of the nominal values. Due to this scaling the weighting matrix in the cost functional is chosen to be $\boldsymbol{Q} = \boldsymbol{I}$. This ensures that all quantities are weighted equally. Estimates of model states and model outputs are computed using the adaptive and scalewise approach and are compared to their nominal values. Scalewise refinement is started from a coarse equidistant mesh based on 9 trial functions.

In the adaptive refinement we also start with 9 trial functions, but these belong to a non equidistant mesh. The mesh is computed by thresholding the measurement functions as outlined in Section 5.1. The time–frequency plot of the trial functions used in the initial mesh are shown in figure 15. During the refinement cycles we insure sufficient increase of measurement information by requiring that the threshold on the measurements decreases in an appropriate way. In particular, the refinement of the measurement function approximation is based on equation (35) where $\varepsilon'_\ell = 0.25/8^\ell$. Thus in each refinement step the approximation error is lowered and the approximation quality is increased sufficiently.

Furthermore, we control the local error of the differential equation based on the one step methods presented in Section 5.2. We require that all mesh midpoints which yield estimated errors above a certain threshold are taken into account in the next step. The threshold is chosen as half of the maximal estimated local error at the current refinement stage.

Six refinement cycles are performed. Figure 15 displays the time–frequency representation of the final mesh. Estimates of the model outputs and states with their associated meshes as well as their nominal values can be seen in figures 17 – 26. Note, that all variables are plotted as deviation values. Differences of estimates and nominal values are hardly noticeable. The local and global behaviour of all quantities are clearly captured by the trial functions.

A comparison of scalewise and adaptive refinement is shown in figure 16 where the sum of the relative L_2 norms $\sum \|y_{\Lambda_{i,\ell}} - \tilde{y}_i\|/\|\tilde{y}_i\|$ against the number of used trial functions are displayed. Adaptive refinement uses less trial functions to obtain similar approximation quality compared to the scalewise approach. On the right side of the figure the computation times of the QP solver are displayed. Since the computation time is strongly coupled to the number of trial functions the result is not surprising. Adaptive discretization is more economical compared to equidistant discretization. However, one question is immediately raised. Is adaptive refinement more economical compared to one single computation carried out on a fixed mesh? More precisely, besides the advantage of having a valid solution to the estimation problem in smaller time increments, does the accumulated effort pay, needed to compute *each* of the so far performed refinement steps? As it can be seen in the figure, the accumulated computation time to obtain a certain approximation level is almost everywhere somewhat below the time needed by a solution based on a single equidistant approximation.

It has to be mentioned that the computation does not employ a warm start with the old solution vector since the general QP solver used, although designed to solve large sparse systems, is based on an active set method employing direct linear algebra methods. Since in this example we have only equality restrictions, reductions through warms starts can not be achieved with this QP solver. However the situation essentially changes dramatically if inequality restrictions are present or a iterative solver for the linear algebra is chosen.

As already mentioned, the multiscale estimates converge to their nominal values. After inspection of the estimated quantities one may ask the question why the discretization mesh is quite dense at the beginning of the time interval even if the nominal values are zero? As it turns out during refinement, the local errors are large at those locations. The underlying reason is that the mesh has to be chosen sufficiently fine to resolve the dynamics of the process. Especially for the discretization scheme used, which is equivalent to the trapezoidal integration formula, it is well known that the step size has to capture the smallest eigenvalue otherwise, oscillatory behaviour is introduced. For the employed model the largest and smallest absolute eigenvalues are $\mid eig_{\max}(\boldsymbol{A}) \mid \approx 190$, $\mid eig_{\min}(\boldsymbol{A}) \mid = 0$. such that the appropriate step sizes are reached only for large scales. Figure 26 shows the estimated flash concentration EO at refinement stage $\ell = 5$. The oscillations at the beginning of the interval are clearly noticeable. Fortunately the flash concentration seems to be the only state of the system which reveals such behaviour.

state	description	unit
x_1	flash temperature	Celsius
x_2	flash holdup liquid phase	kmol
x_3	flash concentration EO liquid phase	kmol/kmol
x_4	flash concentration H2O liquid phase	kmol/kmol
x_5	flash concentration MONO liquid phase	kmol/kmol
x_6	flash concentration DI liquid phase	kmol/kmol
x_7	reactor holdup	kmol
x_8	reactor temperature	Celsius
x_9	reactor concentration EO	kmol/kmol
x_{10}	reactor concentration H2O	kmol/kmol
x_{11}	reactor concentration MONO	kmol/kmol
x_{12}	reactor concentration DI	kmol/kmol

Table 1: State values of the linear ethylene–glycol model

output	description	unit
y_1	flash temperature	Celsius
y_2	reactor temperature	Celsius
y_3	reactor liquid level	m
y_4	flash liquid level	m
y_5	flash pressure	bar

Table 2: Output values of the linear ethylene–glycol model

output	description	unit
u_1	feed flow mixer	kmol/h
u_2	feed temperature mixer	Celsius
u_3	feed concentration EO	kmol/kmol
u_4	reactor cooling duty	GJ
u_5	reactor flow out	kmol/h
u_6	flash heating duty	GJ
u_7	flash vapour stream out	kmol/h
u_8	flash liquid stream out	kmol/h

Table 3: Input values of the linear ethylene–glycol model

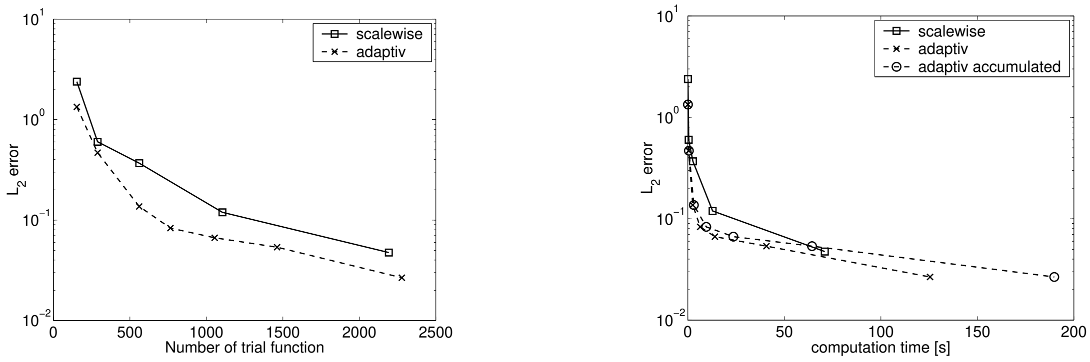

Figure 16: Left: Comparison of scalewise vs. adaptive refinement. The state estimation problem of the glycol process is solved on various resolution levels. The adaptive refinement strategy is based on local error estimates and thresholding of the measurements. Right: Computation times needed by the QP solver. For the adaptive case, the accumulated time, $T_i = \sum_{\ell=0}^{i} T_\ell$ is also shown. Adaptive refinement needs less computation time to achieve comparable accuracy on coarser meshes even if the accumulated computation times are considered. However for very accurate solutions the advantage seems to level off.

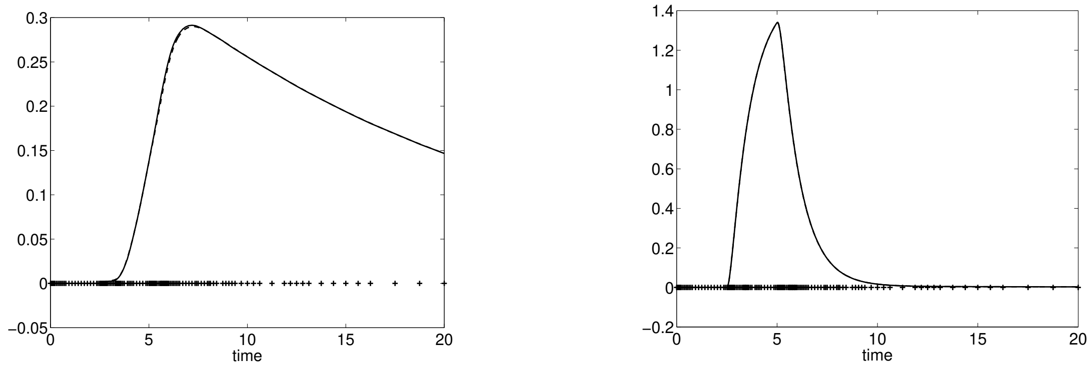

Figure 17: flash temperature, reactor temperature

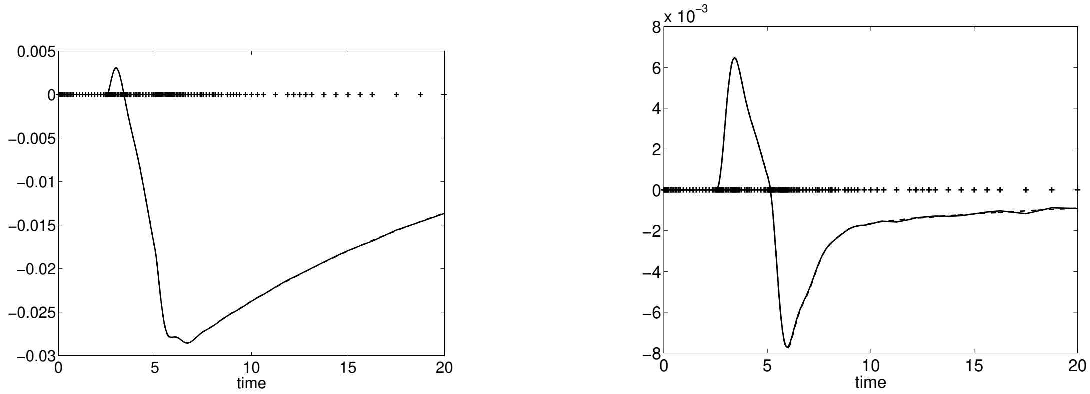

Figure 18: reactor liquid level, flash liquid level

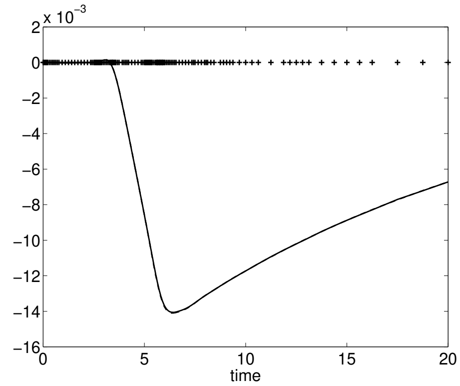

Figure 19: flash pressure

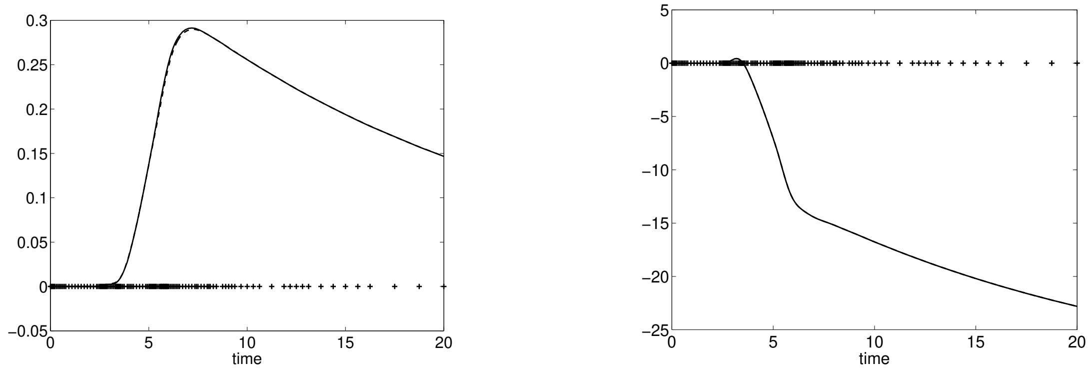

Figure 20: flash temperature, flash holdup liquid phase

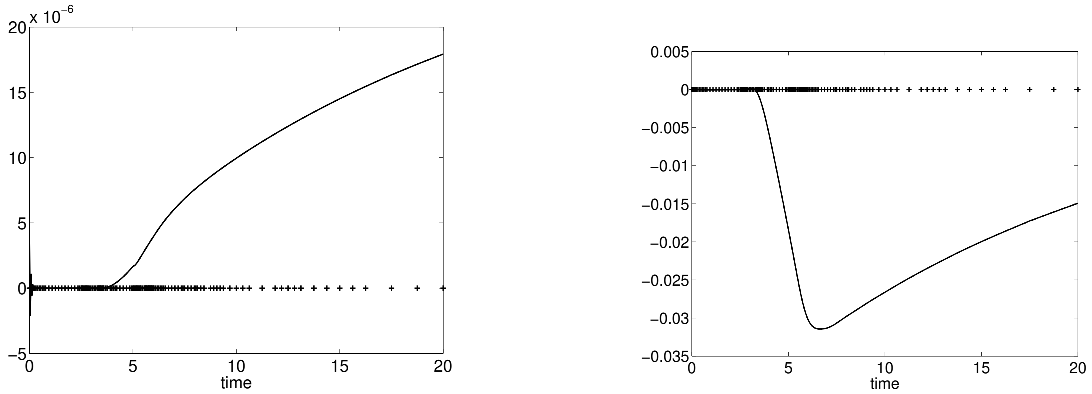

Figure 21: flash concentration EO, flash concentration H2O

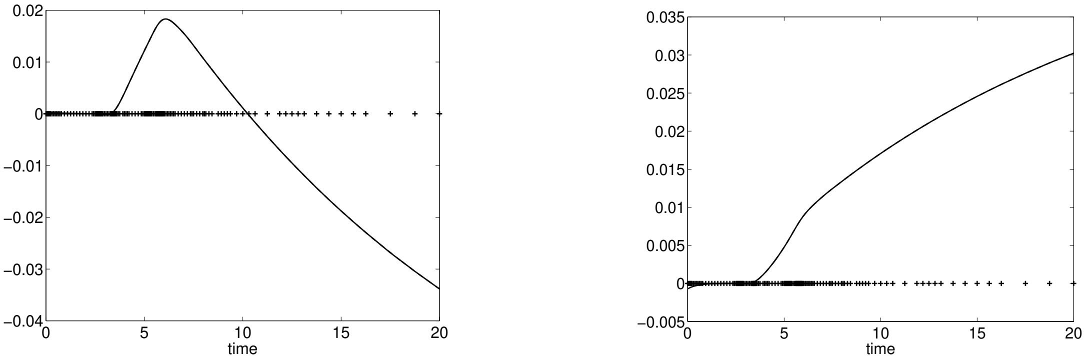

Figure 22: flash concentration MONO, flash concentration DI

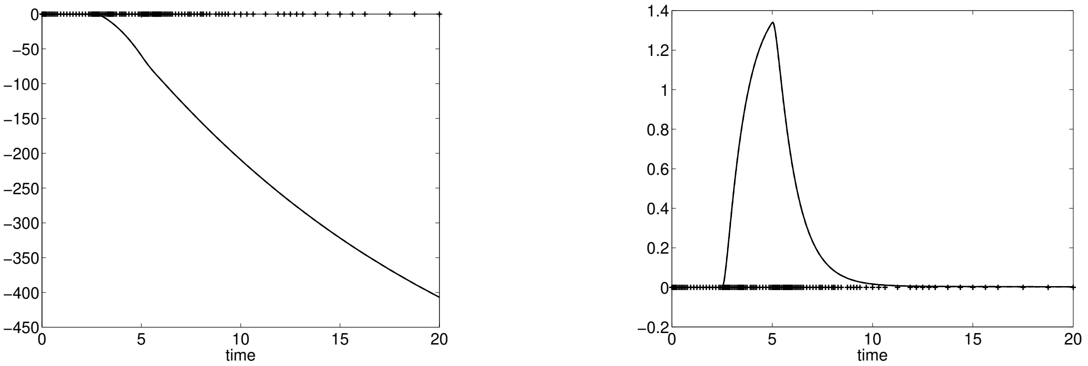

Figure 23: reactor holdup, reactor temperature

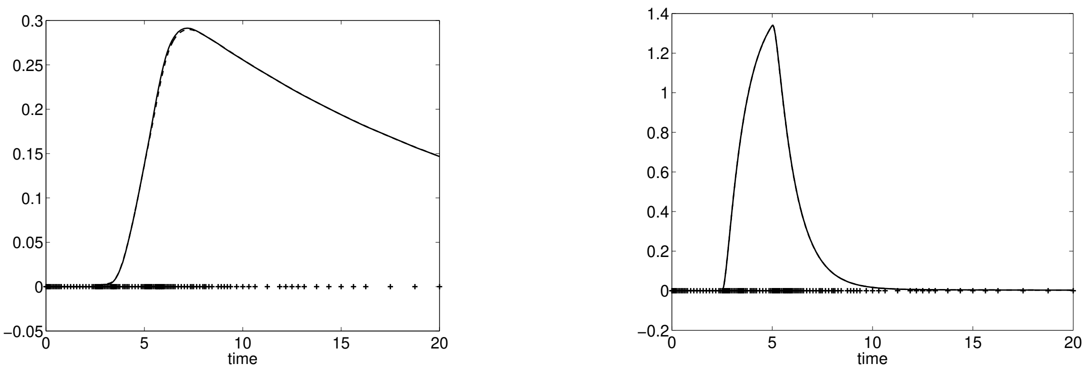

Figure 24: reactor concentration EO, reactor concentration H2O

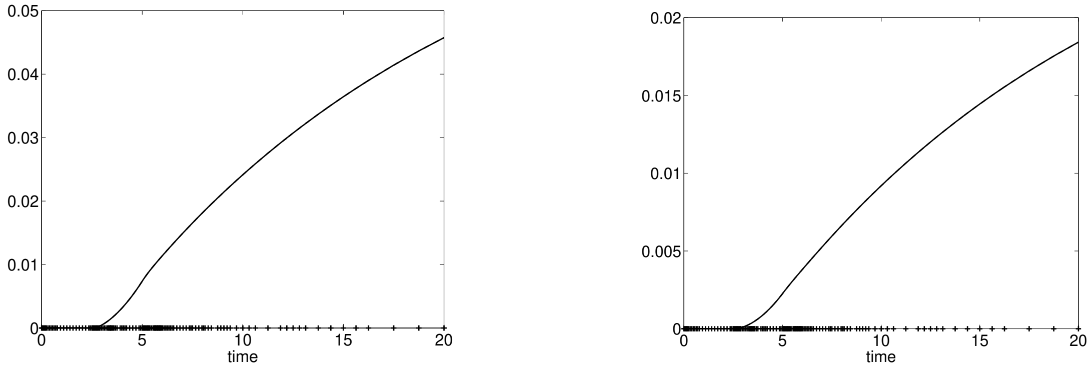

Figure 25: reactor concentration MONO, reactor concentration DI

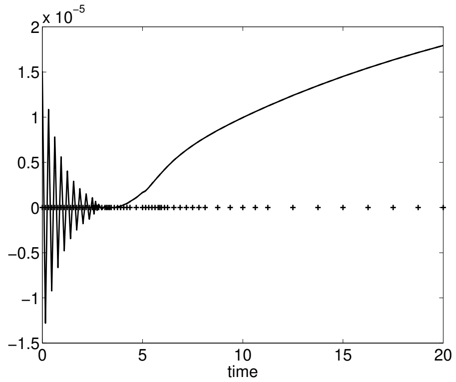

Figure 26: Intermediate estimate of the state flash concentration EO for the adaptive discretization at refinement stage $\ell = 5$. The smallest eigenvalue of the system matrix $eig_m in(A) \approx -190$ introduces oscillatory behaviour. The step size has to be small enough to capture the dynamics.

8 Summary

In the present paper we explored within the refinement concept the reuse of current solutions to generate improved adaptive discretization meshes. The developed strategy is particularly tailored to linear dynamic data reconciliation problems which are formulated in a multiscale setting based upon wavelets. The generation of improved non equidistant discretization meshes in each refinement step was established with a time–frequency analysis of the known input functions combined with local error estimation of the differential model equations as well as with the sensitivity analysis of potential trial functions with respect to the cost functional. It has been demonstrated on two examples that the proposed strategy is capable of identifying adaptive meshes which are more economical than equidistant meshes with respect to the ratio of approximation quality versus number of used trial functions.

The refinement approach solves optimization problems of increasing complexity and prompts in each approximation cycle a valid solution to the current problem. There is evidence, that for certain problems the accumulated numerical expense needed to solve all approximations of past refinements in order to reach a prespecified accuracy is even lower compared to the cost necessary for obtaining equal approximation quality with *one single* equidistant mesh.

The accumulated numerical expense of the refinement approach is influenced among others mainly by two key factors. First, the progress towards the generation of adaptive meshes to reach a highly accurate solution should be sufficiently large, such that only a few refinement steps are needed. Second, to lower the numerical cost further warm start potential such as reuse of Hessian and Jacobian matrices, exploitation of the previous set of inequalities as well as utilizing the previous solution as initial guess for the numerical solver, should be also taken into account to the greatest possible extent. However, in this paper none of these latter warm start options have been explored. These options will be presented elsewhere.

Based on the results obtained so far, we see enough evidence to conclude, that the concept of successively refining a problem approximation offers promising future perspectives for solving not only dynamic data reconciliation problems in a robust and efficient way, but also dynamic optimization problems in general.

References

[1] J.S. Albuquerque and L.T. Biegler. Decomposition algorithms for on-line estimation with nonlinear DAE models. *Comput. Chem. Eng.*, 21(3):283–299, 1997.

[2] J. Bard. *Nonlinear Parameter Estimation*. Academic Press, 1974.

[3] T. Binder, L. Blank, W. Dahmen, and W. Marquardt. Towards multiscale dynamic data reconciliation. In R. Berber and C. Kravaris, editors, *Nonlinear Model Based Process Control*, NATO–ASI Series, pages 623–665, Dordrecht, The Netherlands, 1998. Kluwer Academic Publishers.

[4] T. Binder, L. Blank, W. Dahmen, and W. Marquardt. Comparison of active set and interior point methods for the solution of multiscale discretizations of optimal control problems. LPT Report, in Vorbereitung, RWTH Aachen, 1999.

[5] T. Binder, L. Blank, W. Dahmen, and W. Marquardt. Iterative algorithms for multiscale dynamic data reconciliation. IGPM–Report, in Vorbereitung, RWTH Aachen, 1999.

[6] K.F. Bloss, L.T. Biegler, and W.E. Schiesser. Dynamic process optimization through adjoint formulations and constraint aggregation. *Ind. Eng. Chem. Res.*, 38:421–432, 1999.

[7] A.E. Bryson and Y.-C. Ho. *Applied Optimal Control*. Taylor & Francis, 1975.

[8] C.K. Chui. *An Introduction to Wavelets*. Academic Press, Boston, 1992.

[9] A. Cohen, I. Daubechies, and J.C. Feauveau. Biorthogonal Bases of Compactly Supported Wavelets. *Comm. Pure and Appl. Math.*, 45:485–560, 1992.

[10] J.E. Cuthrell and L.T. Biegler. On the optimization of differential–algebraic process systems. *AIChE J.*, 33(8):1257–1270, 1987.

[11] W. Dahmen. Wavelet and Multiscale Methods for Operator Equations. *Acta Numerica*, pages 55–228, 1997.

[12] W. Dahmen, A. Kunoth, and K. Urban. Biorthogonal spline-wavelets on the interval–stability and moment conditions. *Appl. Comp. Harmon. Anal.*, 6:132–196, 1999.

[13] I. Daubechies. *Ten Lectures on Wavelets.* Society for Industrial and Applied Math., Philadelphia, 1992.

[14] C. de Boor. *A Practical Guide to Splines.* Springer-Verlag, New York, 1978.

[15] P. Deuflhard. *Integration gewöhnlicher Differentialgleichungen.* Springer, Heidelberg, 1994.

[16] H.W. Engl. Regularization methods for the stable solution of inverse problems. *Surv. Math. Ind.*, 3:71–143, 1993.

[17] P.E. Gill, M. Murray, and M.A. Saunders. User's guide for SQOPT 5.3: a Fortran package for large–scale linear and quadratic programming. Internal report, Department of Operations Research, Stanford University, 1997.

[18] I.-W. Kim, M.J. Liebman, and T.F.Edgar. Robust error–in–variables estimation using nonlinear programming techniques. *AIChE J.*, 36(7):985–993, 1990.

[19] V. Law and Y. Sharma. Computation of the gradient and sensitivity coefficients in sum of squares minimization problems with differential equation models. *Comput. Chem. Eng.*, 21(12):1471–1479, 1997.

[20] C.V. Rao and J.B. Rawlings. Nonlinear moving horizon estimation. In *Proc. of Nonlinear Model Predictive Control Workshop – Assessment and Future Directions*, Ascona, Switzerland, in press, 3 –5 June 1999. Birkhäuser Verlag.

[21] C.V. Rao, J.B. Rawlings, and J.H. Lee. Stability of constrained linear moving horizon estimation. In *Proceedings of American Control Conference*, San Diego, June 1999 1999.

[22] D. Robertson, J.H. Lee, and J.B. Rawlings. A moving horizon–based approach for least–squares estimation. *AIChE J.*, 42(8):2209–2223, 1996.

[23] G. Strang. *Wavelets and Filter Banks.* Wellesley–Cambridge, 1996.

[24] S. Walter. *Optimierung und dynamische Simulation des Ethylenglykol–Prozesses.* Studienarbeit, Lehrstuhl für Prozeßtechnik, RWTH Aachen, 1999.

A Optimality conditions of the continous problem

The formulation of the estimation problem differs from classical optimal control problem formulation in a sense that the initial conditions are not given. Therfore we will state the necessary optimality conditions explicitely. We neglect the inequality restriction and consider the problem (11) with equality constraints (4),(5).

The necessary conditions for the optimum can be derived using standard variational calculus. A common approach for such analysis is based on the Hamiltonian function as defined in [7]:

$$\boldsymbol{H}(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z},\boldsymbol{u},\boldsymbol{w},\boldsymbol{v},\boldsymbol{\lambda}_x,\boldsymbol{\lambda}_y) \quad = \quad -\Theta_0 + {\boldsymbol{\lambda}_x}^T\boldsymbol{f} + {\boldsymbol{\lambda}_y}^T(\boldsymbol{y}-\boldsymbol{g}) \quad , \tag{71}$$

where $\boldsymbol{\lambda}_x$ and $\boldsymbol{\lambda}_y$ denote the Lagrange multipliers associated with the equality restrictions. Given this definition the Lagrange function can be written as

$$\mathcal{L}(\boldsymbol{x},\boldsymbol{y},\boldsymbol{w},\boldsymbol{v},\boldsymbol{\lambda}_x,\boldsymbol{\lambda}_y,\boldsymbol{u},\boldsymbol{z}) \quad = \quad \int_0^1 ({\boldsymbol{\lambda}_x}^T\dot{\boldsymbol{x}} - \boldsymbol{H})d\tau \quad . \tag{72}$$

Partial integration together with variational calculus yields a necessary condition. A potential candidate for the optimal solution has to satisfy [7]

$$
\begin{aligned}
0 \overset{!}{=} \; & \boldsymbol{\lambda}_x{}^T(1)\bar{\boldsymbol{x}}(1) - \boldsymbol{\lambda}_x{}^T(0)\bar{\boldsymbol{x}}(0) - \\
& \int_0^1 \left(\dot{\boldsymbol{\lambda}}_x + \frac{\partial \boldsymbol{H}}{\partial \boldsymbol{x}}\right)^T \bar{\boldsymbol{x}} + \left(\frac{\partial \boldsymbol{H}}{\partial \boldsymbol{w}}\right)^T \bar{\boldsymbol{w}} + \left(\frac{\partial \boldsymbol{H}}{\partial \boldsymbol{y}}\right)^T \bar{\boldsymbol{y}} + \left(\frac{\partial \boldsymbol{H}}{\partial \boldsymbol{v}}\right)^T \bar{\boldsymbol{v}} \, d\tau \quad , \qquad (73)
\end{aligned}
$$

in addition to the model equations (4),(5) for arbitrary choices of $\bar{\boldsymbol{x}}, \bar{\boldsymbol{y}}, \bar{\boldsymbol{w}}, \bar{\boldsymbol{v}}$. (73) holds if each term by itself is 0. Note, that for arbitrary choice of $\bar{x}$ the Lagrange functions have to vanish on their boundaries, e.g. $\boldsymbol{\lambda}_x(0) = 0, \boldsymbol{\lambda}_x(1) = 0$. In the case that the initial condition $\boldsymbol{x}(0) = x_0$ is given, $\bar{\boldsymbol{x}}(0)$ could not be arbitrary. Instead we would have $\bar{\boldsymbol{x}}(0) = 0$ and as a consequence $\boldsymbol{\lambda}_x(0)$ would be free since $\boldsymbol{\lambda}_x(0)\bar{\boldsymbol{x}}(0) = 0$ is always fulfilled.

Back substitution of the Hamiltonian yield the adjoint equations

$$
\begin{aligned}
\dot{\boldsymbol{\lambda}}_x &= \frac{\partial \Theta_0}{\partial \boldsymbol{x}} - (\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}})^T \boldsymbol{\lambda}_x + (\frac{\partial \boldsymbol{g}}{\partial \boldsymbol{x}})^T \boldsymbol{\lambda}_y \quad , && (74)\\
\boldsymbol{\lambda}_y &= \frac{\partial \Theta_0}{\partial \boldsymbol{y}} \quad , && (75)\\
0 &= -\frac{\partial \Theta_0}{\partial \boldsymbol{w}} + (\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{w}})^T \boldsymbol{\lambda}_x \quad , && (76)\\
0 &= -\frac{\partial \Theta_0}{\partial \boldsymbol{v}} - (\frac{\partial \boldsymbol{g}}{\partial \boldsymbol{v}})^T \boldsymbol{\lambda}_y \quad , && (77)\\
\boldsymbol{\lambda}_x(0) &= 0 \quad , \; \boldsymbol{\lambda}_x(1) = 0 \quad . && (78)
\end{aligned}
$$

The optimal solution has to satisfy the adjoint equations (74)–(78) as well as the model restrictions (4),(5). Both equation systems are commonly referred to as the boundary value problem.

Similar equations can be derived if we first formulate the KKT conditions for the continuous problem followed by the discretization of the boundary value problem using the same trial and test spaces introduced in Section 4.2. However there are differences which have to be noted. First, the boundary conditions (78) are explicitly enforced in the continuous formulations, while this is not the case in the direct discretization approach. Numerical simulations indicate that the values on the boundary $\lambda_x(0), \lambda_x(1)$ approach zero for step sizes $h \to 0$. Second, if we discretize the boundary value problem, we have to require that $\lambda_x \in H^1(I)$. In our current discretization frame, which is primarily designed for the direct approach, we require only $\lambda_x \in L_2(I)$. However, despite those differences both formulations converge eventually to the correct optimal solution.

B Linear model matrices for Example 2

The model matrices of the linear state space model are given for model 2 in this appendix. They have been derived by means of finite differences from the nonlinear model equations. The algebraic equations have been eliminated by substitution. The linearization has been carried out on the stationary point given in tables (1) - (3) Note, that the matrix elements have physical units. However, for the sake of brevity the matrices given contain only the numerical values. Since the units of the states, inputs and outputs have been given, it is no problem to derive the units of each matrix element.

$$
\boldsymbol{A} \quad = \quad \left[\begin{array}{ccccc}
-3.7601e+00 & -8.3595e-12 & 2.7459e+04 & 1.5457e+01 & -2.6061e+01 \\
0 & 0 & 0 & 0 & 0 \\
6.9207e-05 & 0 & -1.6043e+02 & 7.2909e-03 & 1.1554e-04 \\
9.2144e-04 & -5.0692e-13 & 1.5708e+02 & -2.0333e+00 & 1.5903e-01 \\
-9.5337e-04 & -1.2467e-13 & 1.3736e+00 & 8.7750e-02 & -2.0951e+00 \\
-3.5143e-05 & 1.1622e-13 & 3.3802e-02 & 2.1594e-03 & 3.4220e-05 \\
0 & 0 & 0 & 0 & 0 \\
-5.6147e-10 & 0 & -8.4089e-04 & -4.1485e-06 & -4.4535e-06 \\
-7.8862e-06 & 0 & 1.8061e+01 & -8.3080e-04 & -1.3166e-05 \\
-1.0500e-04 & 0 & -1.7900e+01 & 1.1659e-02 & -1.8121e-02 \\
1.0864e-04 & 0 & -1.5652e-01 & -9.9991e-03 & 1.8708e-02 \\
4.0046e-06 & 0 & -3.8517e-03 & -2.4606e-04 & -3.8994e-06
\end{array}\right.
$$

$$
\begin{array}{ccccc}
-6.0247e+00 & 0 & 5.7958e+00 & 1.8361e+02 & 1.9116e+02 \\
0 & 0 & 0 & 0 & 0 \\
1.1342e-05 & 0 & 0 & 8.0770e+00 & 0 \\
1.5611e-02 & 0 & 0 & 0 & 8.0770e+00 \\
1.3650e-04 & 0 & 0 & 0 & 0 \\
-1.9517e+00 & 0 & 0 & 0 & 0 \\
0 & -1.0837e-01 & -2.1461e+01 & -4.4744e+05 & -1.4925e+03 \\
-5.3142e-06 & 5.8796e-02 & 1.0655e+01 & 2.4277e+05 & 8.0741e+02 \\
-1.2924e-06 & -4.7424e-05 & -9.3920e-03 & -1.9673e+02 & -6.5315e-01 \\
-1.7788e-03 & 4.0784e-06 & 8.0771e-04 & 1.6840e+01 & -8.8271e-01 \\
-1.5554e-05 & 3.4933e-05 & 6.9184e-03 & 1.4424e+02 & 5.1515e-01 \\
2.3650e-03 & 6.8028e-06 & 1.3473e-03 & 2.8089e+01 & 8.1138e-02
\end{array}
$$

$$
\left.\begin{array}{cc}
1.3285e+02 & 6.2955e+01 \\
0 & 0 \\
0 & 0 \\
0 & 0 \\
8.0770e+00 & 0 \\
0 & 8.0770e+00 \\
-1.3228e+03 & -9.1553e+02 \\
7.5825e+02 & 4.1413e+02 \\
-5.7889e-01 & -4.0066e-01 \\
2.1671e-01 & 2.0933e-01 \\
-8.3404e-01 & 1.5442e-01 \\
2.6244e-01 & -1.1061e+00
\end{array}\right]
$$

$$
\boldsymbol{B} = \begin{bmatrix}
0 & 0 & -5.4536e-13 & 0 & -2.7650e-02 & 3.1853e+01 & -1.3339e+00 & 0 \\
0 & 0 & 0 & 0 & 1.0000e+00 & 0 & -1.0000e+00 & -1.0000e+00 \\
0 & 0 & 0 & 0 & 2.0224e-06 & 0 & -2.6578e-06 & 0 \\
0 & 0 & 7.2648e-16 & 0 & 1.1402e-03 & 0 & -1.4984e-03 & 0 \\
0 & 0 & 0 & 0 & -9.1469e-04 & 0 & 1.2021e-03 & 0 \\
0 & 0 & 0 & 0 & -1.8388e-04 & 0 & 2.4165e-04 & 0 \\
1.0000e+00 & 0 & 0 & 0 & -1.0000e+00 & 0 & 1.0000e+00 & 0 \\
1.6173e-11 & 3.2412e-01 & -1.0007e-07 & 5.4960e+00 & 0 & 0 & 5.6718e-12 & 0 \\
1.4425e-04 & 0 & 3.2840e-01 & 0 & 0 & 0 & 7.2405e-08 & 0 \\
-9.9471e-05 & 0 & -3.2840e-01 & 0 & 0 & 0 & 4.0820e-05 & 0 \\
-3.6539e-05 & 0 & 0 & 0 & 0 & 0 & -3.2747e-05 & 0 \\
-6.6762e-06 & 0 & 0 & 0 & 0 & 0 & -6.5829e-06 & 0
\end{bmatrix}
$$

$$
\boldsymbol{C} = \begin{bmatrix}
1.0000e+00 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1.0000e+00 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 4.3787e-03 & 5.9474e-03 & -3.5393e+01 & -5.1264e+01 & -3.4462e+01 & -1.7101e+01 \\
2.4650e-03 & 1.4218e-02 & -8.0071e+00 & -1.1706e+01 & -7.8681e+00 & -3.9012e+00 & 0 & 0 & 0 & 0 & 0 & 0 \\
1.3739e-02 & 0 & 9.0322e+00 & 5.7702e-01 & 9.1440e-03 & 8.9760e-04 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}
$$

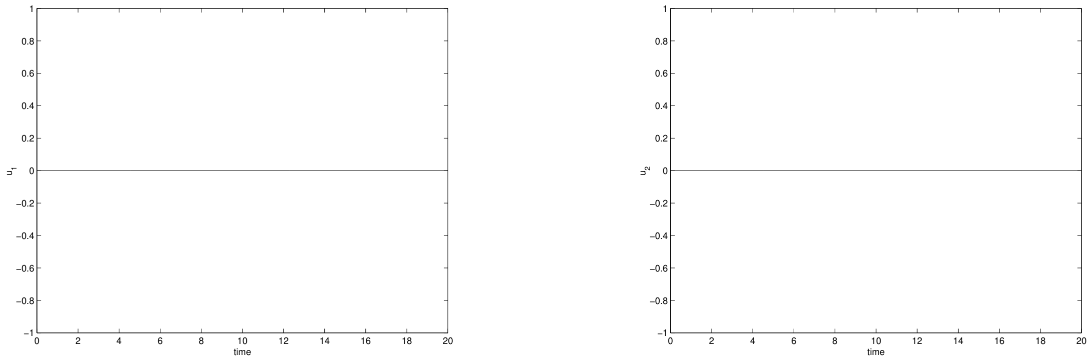

Figure 27: feed flow mixer, feed temperature mixer

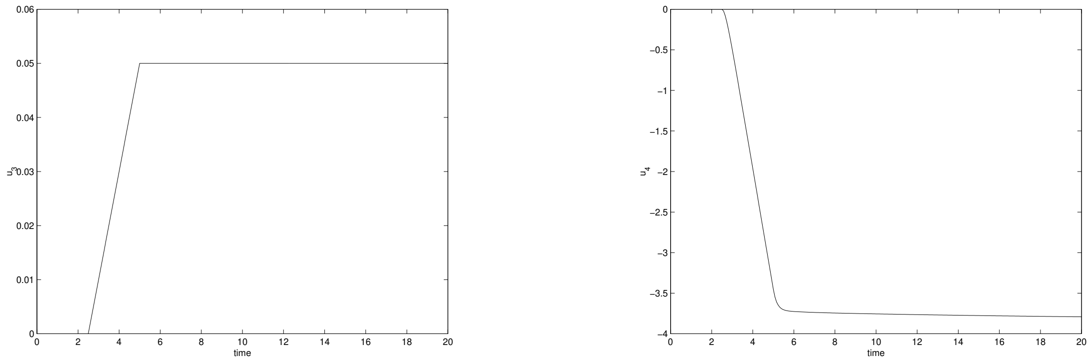

Figure 28: Input functions feed concentration EO, reactor cooling duty

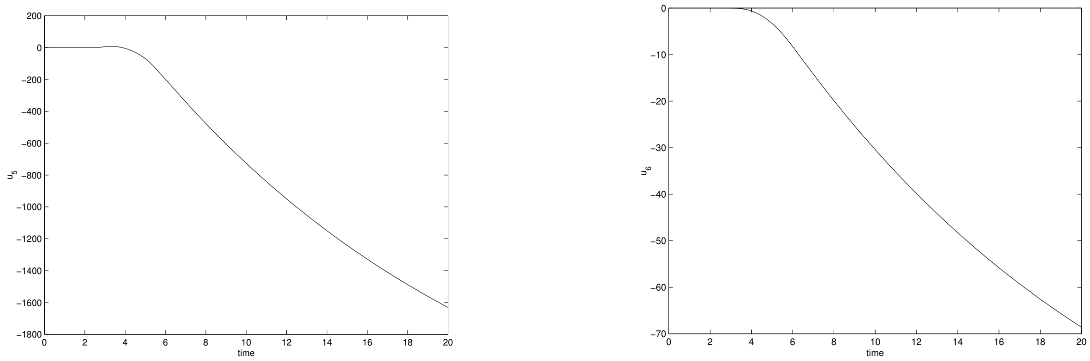

Figure 29: Reactor flow out, flash heating duty

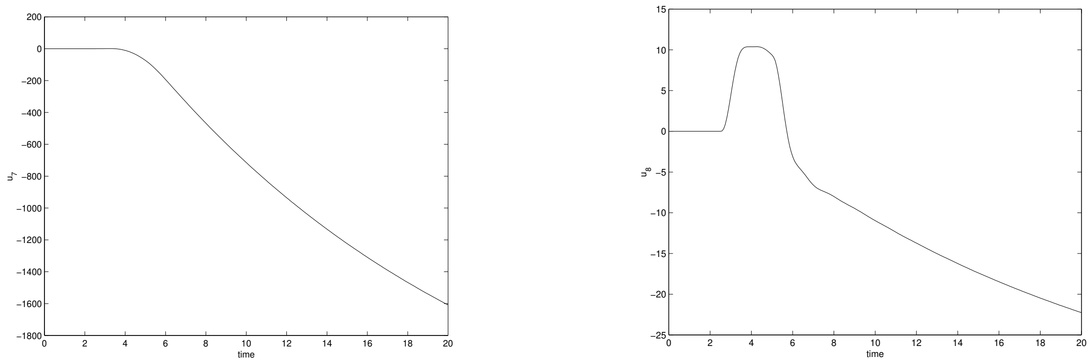

Figure 30: flash vapour stream out, flash liquid stream in