# Iterative Algorithms for Multiscale State Estimation, Part 2: Numerical Investigation*

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

[1] Lehrstuhl für Prozesstechnik,
[2] Institut für Geometrie und Praktische Mathematik,
RWTH Aachen, Germany.

April, 2001

to appear in

Journal of Optimization Theory and Applications

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

**Abstract**

Usually direct methods are employed for the solution of linear systems arising in the context of optimization. However, motivated by the potential of multiscale refinement schemes for large problems of dynamic state estimation, we investigate in this paper the application of iterative solvers based on concepts developed in Ref. 1. Specifically, we explore the effect of different system reductions for various Krylov–space iteration methods as well as three concepts of preconditioning. The first one is the normalization of states and outputs, which also favors error analysis. Secondly, diagonal, scale dependent preconditioners are compared. They all bound the condition numbers independently of the refinement scale, but exhibit significant quantitative differences. Finally, the effect of the regularization parameter on the condition and iteration numbers is analyzed. It turns out that a so called simplified Uzawa scheme with Jacobi preconditioning and suitable regularization parameter is most efficient. However, the experiments also reveal that further improvements are necessary.

**Key Words**: Dynamic optimization, optimal control, iterative linear algebra, wavelets, preconditioning.

# 1 Introduction

Due to the on–line requirements of process monitoring, multiscale state estimation aims at providing an approximation of a dynamic system at any instant of time. This can be achieved by providing a coarse approximation after a very short period and, in the framework of nested iteration, by upgrading this approximation through a hierarchy of successively refined optimization problems (Refs. 1–2). In order to fully exploit prior calculations in the next solution step and observing the fact that it is sufficient to solve each optimization problem only up to the discretization error accuracy, iterative solvers suggest themselves. However, since iterative techniques have to date apparently not received much attention in the context of state estimation and/or control on receding horizons (see Ref. 3 for an introduction), the objective of this paper is to investigate several iterative schemes in the context of multiscale state estimation.

The outline of the paper is as follows: In Section 2 we introduce the specific problem formulation considered here. Then, to explain typical obstructions arising in the above context, we briefly recall several measures of observability and apply them to the two model problems described at the end of the section. Section 3 is devoted to the description of a wavelet based multiscale discretization for problems of this type. In Section 4 we present extensive numerical experiments. Specifically we explore the influence of several system reductions on the performance of different iterative schemes, discuss the effect of certain state normalizations as well as the interplay of the choice of regularization parameters with a tailored preconditioned Uzawa scheme introduced in Part 1 (Ref. 1). We conclude with a brief summary of our findings in Section 5.

# 2 Problem Formulation

The dynamic state estimation can be formulated as a constrained optimization problem, where the difference between measured data $\boldsymbol{z}$ and modeled outputs $\boldsymbol{y}$ is to be minimized (Refs. 2–3). The model equations including in addition to $\boldsymbol{y}$ the states $\boldsymbol{x}$ and the known controls $\boldsymbol{u}$ form the constraints. Moreover, model error functions $\boldsymbol{v}$ and $\boldsymbol{w}$ are included in the model equations to represent unstructured model uncertainty. They have to be estimated together with states and outputs. We stress that the problem in its original form is an ill–posed inverse problem. Therefore some regularization is necessary (Ref. 4).

Since we concentrate on the solution of the final linear systems, we consider here only linear model problems without inequality constraints. Moreover, while our general approach of state estimation is based on a moving horizon approach, we fix the horizon in this paper and scale it to $[0, 1]$. Furthermore, in the chosen examples we allow model uncertainties in all model equations which possibly leads to overparameterization and, therefore, to higher problem complexity. Motivated by the Schur complement considerations in Part 1 (Ref. 1), we do not distinguish between the regularizations of $\boldsymbol{y} - \boldsymbol{z}$ and $\boldsymbol{v}$, but between their regularization and the one of $\boldsymbol{w}$. Hence, we focus here on the following particular setting, where we suppress for convenience explicit references to the domain $[0, 1]$.

For given $\boldsymbol{z} \in (L_2)^{n_y}$ and $\boldsymbol{u} \in (L_2)^{n_u}$ and unknown functions $\boldsymbol{x} \in (H^1)^{n_x}$, $\boldsymbol{w} \in (L_2)^{n_x}$ and $\boldsymbol{y}, \boldsymbol{v} \in (L_2)^{n_y}$ the resulting optimization problem has the form

$$\min_{\boldsymbol{x}_0, \boldsymbol{v}, \boldsymbol{w}} \left\{ ||\boldsymbol{y} - \boldsymbol{z}||_{L_2^{n_y}} + ||\boldsymbol{v}||_{L_2^{n_y}} + \varepsilon \, ||\boldsymbol{w}||_{L_2^{n_x}} \right\} \tag{1}$$

subject to the constraints

$$\dot{\boldsymbol{x}} - \boldsymbol{A}\boldsymbol{x} - \boldsymbol{w} = \boldsymbol{B}\boldsymbol{u} \, , \tag{2}$$

$$\boldsymbol{y} - \boldsymbol{C}\boldsymbol{x} - \boldsymbol{v} = 0 \, . \tag{3}$$

The size of the problem is dominated by the number of states $n_x$. Note that in the above regularized problem the initial data $\boldsymbol{x}(0)$ are not given but have to be estimated. In addition, the observability measure of the model represented by the constraints, the stiffness of the differential equations, the incorporated model error functions and the choice of the regularization parameter $\varepsilon$ influence the condition of the formulated optimization problem and hence the numerical treatment of the resulting linear systems of equations. We briefly recall now the notion of observability.

## 2.1 Observability and Observability Measures

The linear system (2)–(3) is called observable if and only if for fixed $\boldsymbol{u}, \boldsymbol{w}, \boldsymbol{v}$ we can uniquely reconstruct the state $\boldsymbol{x}$ and their initial values from given outputs $\boldsymbol{y}$. The *observability measure* quantifies the sensitivity of $\boldsymbol{y}$ with respect to variations in $\boldsymbol{x}(0)$. Obviously, the size of the observability measure influences the condition of the optimization problem. Different measures have been proposed in the literature. Since it is not clear yet which observability measure is most adequate for our setting we apply the most commonly used measures, described below, to the test examples. For a more thorough treatment we refer to Refs. 5–6.

In principle, a reconstruction of $\boldsymbol{x}$ is obtained by subsequent differentiation of $\boldsymbol{y}$ and repeated substitution of (2) into the differentiated equation (3) up to degree $n_x - 1$. This leads one to define

$$O_1^\rho \;:=\; [\boldsymbol{C}^T, \boldsymbol{A}^T\boldsymbol{C}^T, (\boldsymbol{A}^T)^2\boldsymbol{C}^T, \ldots, (\boldsymbol{A}^T)^{\rho-1}\boldsymbol{C}^T]^T \quad \rho = 1, \ldots, n_x, \tag{4}$$

and to call the linear system (2)–(3) observable if and only if the matrix $O_1^{n_x}$ is of rank $n_x$. The Kalman observability index $\rho$ refers to the smallest integer needed to ensure full rank. The *Kalman observability matrix measure* is defined by the smallest singular value $\sigma_{min}(O_1^\rho)$ of the observability matrix $O_1^\rho$.

Another way to measure observability is given by the Popov–Belevitch–Hautus rank test. Let $\lambda_i$ denote the eigenvalues of $\boldsymbol{A}$ and

$$O_2(\lambda_i) \ := \ [\lambda_i \boldsymbol{I} - \boldsymbol{A}^T, \boldsymbol{C}^T]^T. \tag{5}$$

Then, the pair $(\boldsymbol{A}, \boldsymbol{C})$ is observable if and only if $rank[O_2(\lambda_i)] = n_x$ for all $\lambda_i,\ i = 1, \ldots, n_x$. As before, the observability measure is defined by the smallest singular value $\bar{\sigma}_{min}(O_2) := \min_{\lambda_i, i=1,\ldots,n_x}\{\sigma_{min}(O_2(\lambda_i))\}$.

The observability Gramian is defined by

$$O_3 := \int_0^\infty e^{\boldsymbol{A}^T\tau} \boldsymbol{C}^T \boldsymbol{C} e^{\boldsymbol{A}\tau} d\tau \tag{6}$$

and involves the fundamental solution $e^{\boldsymbol{A}t}$ of the linear system (2)–(3). Observability is given if $\text{rank}(O_3) = n_x$. The associated observability measure is $\sigma_{min}(O_3)$. Note that the observability Gramian is only computable for strictly stable systems, i.e. all eigenvalues $\lambda_i$ of $\boldsymbol{A}$ have to have negative real part.

The modal transformation based measure is based on the observability test

$$\kappa_i \ := \ \left(\bar{\boldsymbol{m}}_i^T \boldsymbol{C}^T \boldsymbol{C} \boldsymbol{m}_i\right) / \left(\bar{\boldsymbol{m}}_i^T \boldsymbol{m}_i\right) > 0 \quad \text{for all eigenvectors } \boldsymbol{m}_i \text{ of } \boldsymbol{A}$$

where $\bar{\boldsymbol{m}}_i$ denotes the complex conjugate of $\boldsymbol{m}_i$. The associated observability measure is given by $\kappa_{min} = \min_i\{\kappa_i\}$.

## 2.2 Test Problems

We describe next two small test problems which, however, exhibit the essential obstructions one encounters in connection with large realistic problems.

*Example 2.1:* In the first example the input is a given control $u$ which is subsequently integrated four times. The result is the output of the system. Amplification factors between the stages and the feedbacks appear in the model equations as constants $k_i \in \mathbb{R}$ and $c_i \in \mathbb{R}$ respectively. Then the resulting system matrices are given by

$$\boldsymbol{A} = \begin{pmatrix} c_1 & 0 & 0 & 0 \\ k_1 & c_2 & 0 & 0 \\ 0 & k_2 & c_3 & 0 \\ 0 & 0 & k_3 & c_4 \end{pmatrix} \quad \boldsymbol{B} = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} \quad \boldsymbol{C}^T = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}. \tag{7}$$

Model correction terms are permitted in all equations. Hence, uniqueness of the estimation problem can only be expected for a suitable regularization. As a case study we choose the constants $\boldsymbol{k} = (5, 10, 5)^T$, $\boldsymbol{c} = -(20, 10, 0.5, 5)^T$. The control function $u$ is chosen as a piecewise linear function, oscillating between 10 and $-10$ with $u(0) = 10$ and step size 0.25. Moreover, we use initial data $\boldsymbol{x}(0) = (1, 0, 0.5, -0.5)^T$ and $\boldsymbol{w} = \boldsymbol{0}$, $\boldsymbol{v} = \boldsymbol{0}$ to generate the nominal solution. As will be seen, despite its very simple structure and low dimension, this example might exhibit typical system properties such as small observability measures, which is typical for industrial applications.

*Example 2.2:* As a second test problem we consider a simple model of a chemical process which produces ethylene glycol from ethylene oxide and glycol (Figure 1). The nonlinear model lumps a number of related states and outputs. Further simplification is accomplished by linearization at some stationary operating point and by subsequent elimination

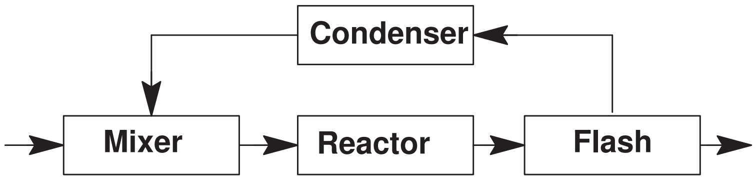


Figure 1: Glycol process.

of the algebraic state variables to result in a linear model with twelve states representing the concentrations of the five major chemical components, the temperature and the holdup in the reactor as well as in the flash. There are five measurements which comprise the temperatures and the levels in the reactor and in the flash as well as the pressure in the flash. The nominal solutions are generated from nominal zero initial conditions and given forcing functions. The forcing functions comprise flow rates within the process as well as heating and cooling duties in the reactor and the flash. For brevity we do not give the numerical values of the system matrices. Details on the model can be found in Ref. 7. However, the eigenvalues $\lambda_i$ provide insight into system behavior and the values of the unscaled model are given by $\{-1.9e2, -1.6e2, -3.8, -2.3, -1.0 \pm i\, 0.12, -1.1, -1.97, -1.94, -0.89, -6.1e{-6}, 0\}$. To ensure that relative errors are equally weighted, the model matrices are scaled such that the components of the nominal solution are roughly of unit size in $L_2$. The reduction of the differential algebraic equations to a model of type (2) by elimination of the algebraic variables has to be paid by the loss of the typically very sparse structure in the model matrices as depicted Figure 2. Moreover, the grey scale in the sparsity patterns of $\boldsymbol{A}$ and $\boldsymbol{C}$ indicate the absolute values of the matrix entries after normalization by means of the nominal solution. As in the first example, we incorporate model correction terms in all

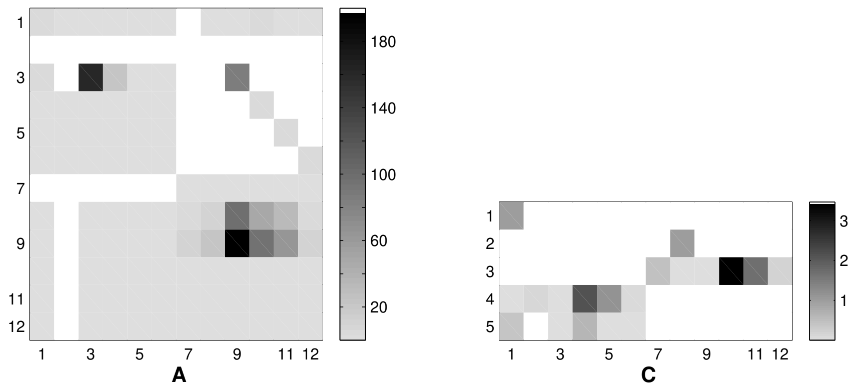


Figure 2: Model matrices.

state and output equations, and deliberately overparameterize the problem.

The observability measures for the test examples are given in Table 1. We also calculated the measures for the special choice $\boldsymbol{C} = \boldsymbol{I}$, which means that all states can be measured directly. This obviously reflects the case of perfect observability and serves as a nominal value for a comparison. The numbers have been calculated for the unscaled problems. Unfortunately, the observability measures applied are not scale invariant. Scaling is crucial for the numerical algorithm and will be discussed in more detail in Section 4.3.

The computed numbers somehow reflect the difficulty a numerical scheme is confronted

Table 1: Observability measures for the test examples.

| Test example | Kalman matrix measure $\rho$ | Kalman matrix measure $\sigma_{\min}(O_1^\rho)$ | Popov–Belevitch Hautus measure $\bar{\sigma}(O_2)$ | Gramian based measure $\sigma_{\min}(O_3)$ | Modal transformation based measure $\kappa_{\min}$ |
|---|---|---|---|---|---|
| Example 2.1 | 4 | 7.0e-1 | 7.4e-2 | 5.5e-6 | 5.5e-3 |
| perfect observable | 1 | 1 | 1 | 2.4e-2 | 1 |
| Example 2.2 | 3 | 3.3e-3 | 1.2e-4 | not applicable | 1.9-5 |
| perfect observable | 1 | 1 | 1 | not applicable | 1 |

with when reconstructing the states from the measurements. Varying the entries $k$ and $c$ in *Example 2.1*, leads to significantly different observability measures. This indicates that although the example has only four states and one output function the reconstruction can face severe problems. The observability measure for *Example 2.2* is clearly worse than for *Example 2.1* with respect to all measures. It is rather difficult to draw precise quantitative conclusions from the measures since the numbers differ significantly depending on the respective concept. However, the measures can be used to qualitatively appraise the numerical difficulty one has to expect when solving the reconstruction problem.

# 3 Wavelet–Galerkin Discretization

We discretize the saddle point problem corresponding to the optimization problem (1)–(3) by a wavelet–Galerkin ansatz with a chosen, finite index set $\Lambda$ of wavelet basis functions. A detailed description of the discretization is given in Part 1 (Ref. 1). In addition, we refer readers who are not familiar with wavelets to Refs.8–10. We recall here only briefly the main notational ingredients. The functions $\boldsymbol{x}, \boldsymbol{z}, \boldsymbol{y}, \boldsymbol{v}$ are discretized with globally continuous, piecewise linear spline wavelets $\Psi_{j,k}$, $(j,k) \in \Lambda$ ($x_i = \sum_{j,k} d_{j,k}\Psi_{j,k}$). The dual wavelets $\tilde{\Psi}_{j,k}$, $(j,k) \in \Lambda$, form the basis for the Lagrange multipliers $\boldsymbol{\nu}$ corresponding to the output equation (3). Moreover, the piecewise constant Haar wavelets $\Psi^H_{j,k}$, $(j,k) \in \Lambda'$, are chosen for discretizing $\boldsymbol{w}, \boldsymbol{u}$ and the Lagrange multipliers $\boldsymbol{\mu}$ corresponding to the differential equations (2). Since for a given coarsest level the number of coarse level scaling functions for the Haar basis is by one less than the number of corresponding piecewise linear scaling functions, the index set $\Lambda'$ differs from $\Lambda$ only in one scaling function index. In fact, the necessary variation of initial values requires reserving at least one degree of freedom in each state variable $x_i$ in excess of the test conditions. Let us collect now in $\boldsymbol{d_x}$ all wavelet coefficients of the states $\boldsymbol{x}_\Lambda$ ($\boldsymbol{d_x} = \left(\boldsymbol{d}^\Lambda_{x_i}\right)_{i=1,\ldots n_x}$) and likewise for $\boldsymbol{d_y}, \boldsymbol{d_v}, \boldsymbol{d_w}$ etc. where we have suppressed for convenience the index $\Lambda$. Furthermore we use the shorthand notation of $\langle\langle \boldsymbol{\Psi}_\Lambda, \boldsymbol{\Psi}_{\Lambda'}\rangle\rangle = \left(\langle \psi_{(j,k)}, \psi_{(j',k')}\rangle\right)_{(j,k)\in\Lambda,(j',k')\in\Lambda'}$ for $L_2$–mass matrices, which typically have sparse, so called finger structure. To describe the discretization matrix in a rather condensed form we use the Kronecker product $\boldsymbol{A} \otimes \boldsymbol{B} := (a_{i,j}\boldsymbol{B})_{\substack{i=1,\ldots,n \\ j=1,\ldots,m}}$ of a matrix $\boldsymbol{A} \in I\!R^{n\times m}$ with a matrix $\boldsymbol{B}$. With $\boldsymbol{I}_n$, $\boldsymbol{I}_\Lambda$ we denote the identity matrix of size $n \times n$, respectively $(\#\Lambda) \times (\#\Lambda)$ and define the matrices

$$\boldsymbol{Q}_\Lambda = \boldsymbol{I}_{n_y} \otimes \langle\langle \boldsymbol{\Psi}_\Lambda, \boldsymbol{\Psi}_\Lambda\rangle\rangle, \quad \boldsymbol{C}_\Lambda = \boldsymbol{C} \otimes \boldsymbol{I}_\Lambda, \quad \boldsymbol{B}_\Lambda = \boldsymbol{B} \otimes \boldsymbol{I}_\Lambda,$$
$$\boldsymbol{X}_\Lambda = \boldsymbol{I}_{n_x} \otimes \langle\langle \boldsymbol{\Psi}^H_{\Lambda'}, \dot{\boldsymbol{\Psi}}_\Lambda\rangle\rangle - \boldsymbol{A} \otimes \langle\langle \boldsymbol{\Psi}^H_{\Lambda'}, \boldsymbol{\Psi}_\Lambda\rangle\rangle\ .$$

Then, the Galerkin discretization leads to the following linear system of equations

$$\begin{pmatrix} & & & \boldsymbol{X}_\Lambda{}^T & -\boldsymbol{C}_\Lambda{}^T \\ 2\boldsymbol{Q}_\Lambda & & & & \boldsymbol{I} \\ & 2\boldsymbol{Q}_\Lambda & & & -\boldsymbol{I} \\ & & 2\varepsilon \boldsymbol{I} & -\boldsymbol{I} & \\ \boldsymbol{X}_\Lambda & & -\boldsymbol{I} & & \\ -\boldsymbol{C}_\Lambda & \boldsymbol{I} & -\boldsymbol{I} & & \end{pmatrix} \begin{pmatrix} \boldsymbol{d_x} \\ \boldsymbol{d_y} \\ \boldsymbol{d_v} \\ \boldsymbol{d_w} \\ \boldsymbol{d_\mu} \\ \boldsymbol{d_\nu} \end{pmatrix} = \begin{pmatrix} 0 \\ 2\boldsymbol{Q}_\Lambda \boldsymbol{d_z} \\ 0 \\ 0 \\ \boldsymbol{B}_\Lambda \boldsymbol{d_u} \\ 0 \end{pmatrix}. \tag{8}$$

Note that all the wavelet stiffness matrices appearing in $\boldsymbol{X}_\Lambda$ are singular, since the number of rows is by one less than the number of columns. Moreover, in general, $\boldsymbol{C}_\Lambda$ is also singular due to the singularity of $\boldsymbol{C}$.

# 4 Numerical Results

In this section we examine the performance of several iterative methods for the solution of the KKT system (8). *Examples 2.1* and *2.2* serve as test problems. It will be seen that in particular in *Example 2.2* extremely large condition numbers are encountered. Here, normalization of the states and the judicious choice of the regularization parameter $\varepsilon$ in (1) have to complement an efficient preconditioner. Therefore, unless stated otherwise, the experiments are carried out for *Example 2.2* with state normalization and the choice $\varepsilon = 10^{-4}$, which will be seen to be optimal, while for *Example 2.1* we set $\varepsilon = 1$.

While one expects an adaptive selection of trial spaces, that is the adaptive choice of the index sets $\Lambda$ to be important for real time computation (Ref. 7), we confine our tests of the iterative solution of the KKT–matrix to uniform discretizations of mesh size $2^{-J}$, i.e. the index set $\Lambda_J$ consists of all possible wavelet indices $(j,k)$ up to scale $J$. For nonuniform meshes we expect similar results. Nevertheless, since the linear system (8) is part of a multiscale approach (see Part 1), we have to consider mesh refinement and its effect on the iteration numbers. The objective is to keep the number of required iterations low and uniformly bounded with increasing scale $J$. The choice of the coarsest scale depends, among other things, on the number of outputs relative to the number of states. For *Example 2.2* we can start with scale 1, while scale 2 is necessary for *Example 2.1*. As maximal scale we choose 7 in both examples which corresponds to the mesh size $2^{-7}$. Of course, the corresponding systems have different dimensions, which should be taken into account when interpreting the iteration numbers.

Due to the nested iteration ansatz in the multiscale approach (see Part 1), we use the approximate solution from the preceding scale $j-1$ as initial guess for the current discretization with $\Lambda_j$. Here the hierarchical structure of wavelet bases implies, that one only has to append zero coefficients to the array of wavelet coefficients from the previous solution.

A typical stopping criterion requires that the residual $\boldsymbol{res} = \boldsymbol{K} \cdot \boldsymbol{sd} - \boldsymbol{rhs}$ is smaller than some target tolerance $tol$ unless some maximal number $max_{it}$ is exceeded. Here $\boldsymbol{K}$ is the (preconditioned) system matrix under consideration and $\boldsymbol{sd}, \boldsymbol{rhs}$ denote the computed approximate solution and right hand side, respectively. Of course, these quantities depend on the particular preconditioner $\boldsymbol{M}_\Lambda$, on the current index set $\Lambda$ and on the regularization parameter $\varepsilon$. However, due to the large condition numbers, the residuals are not very reliable. Therefore we monitor here for testing purposes the approximation to the exact error $\boldsymbol{e}_{it} = \boldsymbol{sd}_{it} - \boldsymbol{sd}_{dir}$, the difference between the iterative solution $\boldsymbol{sd}_{it}$ and the solution $\boldsymbol{sd}_{dir}$ which for *Example 2.1* is produced with the aid of a direct solver. Because of the extremely high condition numbers in *Example 2.2* even the results of a direct solver are too inaccurate. Therefore $\boldsymbol{sd}_{dir}$ is in this case chosen as the solution produced by the routine *symmlq*[1] until the residual is of the order of machine accuracy. We are primarily interested in the $L_2$ error of the states. In view of the Riesz basis property (Ref. 9–10), this simply amounts to taking the Euclidean norm $||\boldsymbol{e}_{it}||_{\ell_2}$. Since our primary interest at this stage is not a time optimal complete code but rather a possibly thorough answer to the questions stated in the introduction, we deviate in this paper from the original nested iteration concept suggested in Part 1 and iterate on each level until a certain fixed accuracy is attained. Specifically, we choose the stopping criteria

$$||\boldsymbol{e}_{it}||_{\ell_2} \leq tol, \qquad k = max_{it}$$

with $max_{it} = 3000$ and $tol = 10^{-6}$ for *Example 2.1* and $tol = 10^{-4}$ for *Example 2.2*. Moreover, we choose Matlab (Version 5.2) as a convenient computational environment, and partly include C and C++ codes. For a description of the iterative methods used below we refer to Refs. 11–12 and the references cited there.

## 4.1 Comparison of Different System Formulations

As a first step for analyzing the iterative solvers we investigate possible inexpensive system reductions. Specifically, we denote the KKT system (8) by $\text{red}_1$. The elimination of $\boldsymbol{d_y}, \boldsymbol{d_v}, \boldsymbol{d_w}$ with their representation given by the Lagrange parameters $\boldsymbol{d_\mu}$ and $\boldsymbol{d_\nu}$ yields the system $\text{red}_2$. Additional elimination of $\boldsymbol{d_\nu}$ gives the system $\text{red}_3$ which corresponds to the variables $\boldsymbol{d_x}$ and $\boldsymbol{d_\mu}$. The last reduction step ($\text{red}_4$) eliminates $\boldsymbol{d_\mu}$ and supplies a positive definite matrix, which is the Schur complement $\boldsymbol{\mathcal{S}}_\Lambda$ of $\text{red}_2$

$$\boldsymbol{\mathcal{S}}_\Lambda = \boldsymbol{C}_\Lambda^T \left( \boldsymbol{I}_{n_y} \otimes \langle\langle \boldsymbol{\Psi}_{\Lambda_y}, \boldsymbol{\Psi}_{\Lambda_y} \rangle\rangle \right) \boldsymbol{C}_\Lambda + 2\varepsilon \boldsymbol{X}_\Lambda^T \boldsymbol{X}_\Lambda \,. \tag{9}$$

The different formulations of the KKT system (8) exhibit different features, as for example condition numbers, dimensions and definiteness. Hence, we investigate first the influence of these differences on the performance of an iterative scheme. In particular, since the number of nonzero entries remains roughly the same for all reductions, we compare the numbers of iterations. As an iterative solver we choose *symmlq* because it is designed for symmetric not necessarily definite systems and therefore applies to each formulation. Like all iterative methods tested below it is based on the construction of orthogonal bases for Krylov subspaces. In the case of *symmlq* a three term recurrence relation is used (Refs. 11, 13). We use the diagonal, left hand side preconditioner $\boldsymbol{M}^{l_2}$ with the $l_2$–norm of the corresponding matrix rows as entries. The justification for this choice which can be applied to all system formulations will be given later in Section 4.4. In the Tables 2 and 3 the iteration numbers corresponding to the above stopping criteria are recorded for both test examples.

Obviously, even though the reduced systems may have larger condition numbers (e.g. tests have confirmed that $\text{cond}_2(\text{red}_4) \approx (\text{cond}_2(\text{red}_1))^2$), the iteration numbers clearly indicate, that the use of the smaller system is preferable. On the other hand, it would be desirable not to assemble the Schur complement $\boldsymbol{\mathcal{S}}_\Lambda$.

[1]The version *symmlq* employed here has been kindly provided by Michael Saunders.

Table 2: Iteration numbers for different system formulations, *Example 2.1.*

| Scale | $\text{red}_1$ | $\text{red}_2$ | $\text{red}_3$ | $\text{red}_4$ |
|---|---|---|---|---|
| 2 | 64 | 43 | 36 | 21 |
| 3 | 103 | 80 | 70 | 38 |
| 4 | 138 | 112 | 94 | 53 |
| 5 | 163 | 137 | 112 | 62 |
| 6 | 177 | 161 | 132 | 73 |
| 7 | 201 | 182 | 155 | 85 |

Table 3: Iteration numbers for different system formulations, *Example 2.2.*

| Scale | $\text{red}_2$ | $\text{red}_4$ |
|---|---|---|
| 1 | 492 | 130 |
| 2 | 906 | 224 |
| 3 | 1432 | 359 |
| 4 | 2192 | 463 |
| 5 | 2772 | 555 |
| 6 | max | 578 |
| 7 | max | 596 |

## 4.2 Comparison of Different Iterative Methods

Exploiting the specific problem structure we developed in Part 1 a *simplified Uzawa* scheme. Essentially it is based on the *pcg*–method applied to the Schur complement $\boldsymbol{\mathcal{S}}_\Lambda$ but uses the system $\text{red}_2$. However, the difference is that the *simplified Uzawa* algorithm calculates the solution $\boldsymbol{d_x}, \boldsymbol{d_\mu}$ and $\boldsymbol{d_\nu}$ iteratively without setting up $\boldsymbol{\mathcal{S}}_\Lambda$ and at nearly the same cost as *pcg* needs for calculating only $\boldsymbol{d_x}$. Hence it is not surprising that the respective numbers of iterations are the same as all tests confirm (not shown here).

Therefore, to compare the performance of different iterative methods with the *simplified Uzawa* scheme we choose the system $\text{red}_2$. As left hand side preconditioner we use again $\boldsymbol{M}^{l_2}$. In addition to *symmlq* we apply *gmres(k)* and *bicg* although they do not exploit symmetry as well as the *simplified Uzawa* scheme. Methods like *cgne* and *cgnr* are inappropriate since they square the already high condition numbers. The advantage of using *symmlq, bicg* or *gmres* could be, that they apply conceptually to matrices whose condition numbers are smaller than the condition numbers of the Schur complements, which are relevant for the simplified Uzawa algorithm. It is remarkable that *gmres* only converged without restart, to which all listed results refer to. In view of the enormous cost, this method turns out not to be suitable.

The Tables 4 and 5 show that *symmlq* and *bicg* require nearly the same number of iterations. Nevertheless, the computational work for one iteration in the *bicg* method roughly doubles that required by *symmlq*, since *bicg* biorthogonalizes two sequences. For *Example 2.1 gmres* without restart is slowest, while the *simplified Uzawa* algorithm yields the best results. For *Example 2.2*, *gmres* requires the lowest iteration numbers for the lower scales. However, since *gmres* employs orthogonalization with respect to all previous directions, significantly more work is needed for one iteration when progressing to higher scales. For scale 6 and 7, the required storage is even too large to allow simple testing.

Table 4: Iteration numbers for different iterative methods, *Example 2.1*.

| Scale | symmlq | bicg | simplified Uzawa | gmres |
|---|---|---|---|---|
| 2 | 43 | 44 | 23 | 41 |
| 3 | 80 | 80 | 40 | 77 |
| 4 | 112 | 112 | 64 | 149 |
| 5 | 137 | 137 | 90 | 288 |
| 6 | 161 | 163 | 123 | 527 |
| 7 | 182 | 182 | 166 | 927 |

Table 5: Iteration numbers for different iterative methods, *Example 2.2*.

| Scale | symmlq | bicg | simplified Uzawa | gmres |
|---|---|---|---|---|
| 1 | 492 | 488 | 146 | 75 |
| 2 | 906 | 877 | 263 | 133 |
| 3 | 1432 | 1580 | 404 | 249 |
| 4 | 2192 | 2427 | 613 | 478 |
| 5 | 2772 | 2963 | 709 | 899 |
| 6 | max | max | 743 | – |
| 7 | max | max | 689 | – |

Overall, the *simplified Uzawa* algorithm seems to perform best. The above comparison of different iterative schemes motivates now to explore different ways of improving the efficiency of the simplified Uzawa algorithm. Since for this method the condition number and the eigenvalue distribution of $\boldsymbol{\mathcal{S}}_\Lambda$ is as relevant as for the corresponding pcg–method, we aim at decreasing the condition numbers. Three possible major influences are analyzed in the following: normalizing the states and outputs, preconditioning $\boldsymbol{\mathcal{S}}_\Lambda$ and varying the parameter $\varepsilon$.

## 4.3 Normalization

Normalizing the states, outputs and measurements using possibly available information about the nominal solution can also be viewed as preconditioning with an additional change of the previously chosen regularization. However, normalized variables ($||x_i||_{L_2} \approx 1$ etc.) should be introduced before regularization is fixed. The fact that normalization can have a significant effect on condition numbers is illustrated by Table 6 which shows the original and normalized system for *Example 2.2* with $\varepsilon = 1$. It reveals that proper normalization is necessary for this example. Aside from the effect on conditioning the system, normalization also weighs the state errors equally, which is advantageous for the error analysis.

## 4.4 Comparison of Different Preconditioners

Before investigating different preconditioners it is important to point out two characteristic features of $\boldsymbol{\mathcal{S}}_\Lambda$. On one hand, one faces the problem of a scale wise increase of the largest eigenvalue of $\boldsymbol{\mathcal{S}}_\Lambda$, i.e. $\lim \sigma_{\max}(\Lambda) \to \infty$ as $\#\Lambda \to \infty$, which is a consequence of the involved differential operator. On the other hand, it may happen that the smallest eigenvalue

Table 6: cond($\boldsymbol{\mathcal{S}}_\Lambda$) without and with normalizing of $\boldsymbol{x}$ and $\boldsymbol{y}$, *Example 2.2*.

| Scale | without | with |
|---|---|---|
| 1 | 8.1694e+19 | 9.2508e+10 |
| 2 | 1.4066e+20 | 1.0107e+11 |
| 3 | 1.2968e+19 | 5.4481e+10 |
| 4 | 3.8306e+19 | 3.1693e+10 |
| 5 | 6.4281e+19 | 1.9665e+10 |
| 6 | 1.0396e+20 | 1.4867e+10 |
| 7 | 3.2429e+19 | 2.2272e+10 |

$\sigma_{\min}(\Lambda)$ of $\boldsymbol{\mathcal{S}}_\Lambda$ is very close to 0 for all $\Lambda$. This is mainly caused by system inherent features, like possibly low observability measure and high stiffness of the differential state equations. Here, the choice of regularization plays an important role and the matrices $\boldsymbol{\mathcal{S}}_\Lambda$ are often very ill–conditioned. This is indeed the case for both test examples. Nevertheless, while we increase the smallest eigenvalue $\sigma_{\min}(\Lambda)$ by varying $\varepsilon$ (see Subsection 4.5), the maximal eigenvalue $\sigma_{\max}(\Lambda)$ increasing with the scale of refinement can be bounded independently of $\Lambda$ by wavelet based preconditioners (see Part 1). More specifically, symmetric, scale depending diagonal preconditioning $\boldsymbol{M}^{1/2}$ with entries of order $2^{-j}$ for scale $j$ provide condition numbers bounded independently of $\Lambda$. Hence, in the following we compare various choices of diagonal entries with this feature.

The simplest preconditioner is the *diagonal matrix*

$$(\boldsymbol{M}^w_\Lambda)_{i,i'} = 2^{-2j}\delta_{i,i'} \tag{10}$$

with $i \widehat{=} (l,j,k)$ and $l = 1,\ldots,n_x$ and $(j,k) \in \Lambda$.

Note that the above preconditioning corresponds to a properly scaled wavelet basis in the Sobolev space $H^1$. One expects that the involved Riesz constants can be improved by normalizing the wavelet basis in the relevant energy space defined by the Schur complement. This suggests to employ as a diagonal preconditioner the *Jacobi preconditioner*, namely the matrix $\mathcal{M}^J_\Lambda$ obtained by taking the inverses of the diagonal entries of the Schur complement

$$\left(\boldsymbol{M}^J_\Lambda\right)_{i,i'} := \left((\mathcal{S}_\Lambda)_{i,i}\right)^{-1}\delta_{i,i'}\ . \tag{11}$$

Recall that we wish to avoid expensive matrix multiplications. Therefore, we consider the additional option of *approximating* the diagonal entries of the Schur complement while avoiding its explicit computation. This can be accomplished by exploiting that the involved inverses have, on account of the chosen wavelet discretization, their maximal entries appearing on the diagonal. Therefore, we approximate the diagonal of $\boldsymbol{\mathcal{S}}_\Lambda$ by using their diagonals only. This yields the *approximate Jacobi* preconditioner $\boldsymbol{M}^{aJ}_\Lambda$ with

$$\begin{aligned} \left(\boldsymbol{M}^{aJ}_\Lambda\right)_{i,i'} \;=\; & \Bigg(\sum_s \left(\boldsymbol{Q}_\Lambda^{-1} + \boldsymbol{V}_\Lambda \boldsymbol{R}_{\boldsymbol{v}_\Lambda}{}^{-1} \boldsymbol{V}_\Lambda^T\right)^{-1}_{(s,s)} \boldsymbol{C}^2_{\Lambda,(s,i)} + \\ & \sum_s \left(\boldsymbol{W}_\Lambda \boldsymbol{R}_{\boldsymbol{w}_\Lambda}{}^{-1} \boldsymbol{W}_\Lambda^T\right)^{-1}_{(s,s)} \boldsymbol{X}^2_{\Lambda,(s,i)}\Bigg)^{-1} \delta_{i,i'}\ . \end{aligned} \tag{12}$$

Only 4 very sparse vector multiplications are necessary for each entry to determine $\boldsymbol{M}^{aJ}_\Lambda$ (Part 1).

Other scaling strategies with the same qualitative behavior are conceivable. In order to see to which extent the exact values of the scaling weights matter we have applied analogous diagonal preconditioners of the following form. Denoting by $\boldsymbol{\mathcal{S}}_\Lambda^{(i)}$ the $i$–th row of $\boldsymbol{\mathcal{S}}_\Lambda$ we set

$$\left(\boldsymbol{M}_\Lambda^{\ell_1}\right)_{i,i'} := \| \boldsymbol{\mathcal{S}}_\Lambda^{(i)} \|_{\ell_1}^{-1} \delta_{i,i'}, \quad \left(\boldsymbol{M}_\Lambda^{\ell_2}\right)_{i,i'} := \| \boldsymbol{\mathcal{S}}_\Lambda^{(i)} \|_{\ell_2}^{-1} \delta_{i,i'} . \tag{13}$$

Their disadvantage is that $\boldsymbol{\mathcal{S}}_\Lambda$ has to be assembled.

The results recorded in the following tables confirm the above statements. Table 7 lists the condition numbers and Table 8 gives the iteration numbers for *Example 2.1.* Exact and approximate Jacobi preconditioning $\boldsymbol{M}_\Lambda^J, \boldsymbol{M}_\Lambda^{aJ}$ yield exactly the same results for condition and iteration numbers and are therefore not listed separately. This reveals that exact determination of the diagonal of the Schur complement is not necessary. Obviously,

Table 7: cond $\left(\boldsymbol{M}_\Lambda^{1/2}\boldsymbol{\mathcal{S}}_\Lambda\boldsymbol{M}_\Lambda^{1/2}\right)$ for various preconditioners, *Example 2.1.*

| Scale | $\boldsymbol{I}$ | $\boldsymbol{M}_\Lambda^{\ell_1}$ | $\boldsymbol{M}_\Lambda^{\ell_2}$ | $\boldsymbol{M}_\Lambda^{aJ}$ | $\boldsymbol{M}_\Lambda^{w}$ |
|---|---|---|---|---|---|
| 2 | 7.5222e+06 | 2.5464e+06 | 2.3050e+06 | 1.8998e+06 | 2.7637e+07 |
| 3 | 6.8613e+06 | 4.2752e+06 | 3.9818e+06 | 4.5289e+06 | 5.1785e+07 |
| 4 | 2.1416e+07 | 9.0804e+06 | 7.5754e+06 | 8.7587e+06 | 7.8658e+07 |
| 5 | 8.2050e+07 | 1.6244e+07 | 1.1043e+07 | 1.1880e+07 | 9.4910e+07 |
| 6 | 3.1474e+08 | 2.6204e+07 | 1.4261e+07 | 1.3354e+07 | 1.0040e+08 |
| 7 | 1.2185e+09 | 3.9772e+07 | 1.8236e+07 | 1.4006e+07 | 1.0194e+08 |

Table 8: Iteration numbers for various preconditioners, *Example 2.1.*

| Scale | $\boldsymbol{I}$ | $\boldsymbol{M}_\Lambda^{\ell_1}$ | $\boldsymbol{M}_\Lambda^{\ell_2}$ | $\boldsymbol{M}_\Lambda^{aJ}$ | $\boldsymbol{M}_\Lambda^{w}$ |
|---|---|---|---|---|---|
| 2 | 28 | 22 | 22 | 23 | 30 |
| 3 | 48 | 39 | 39 | 41 | 56 |
| 4 | 84 | 66 | 54 | 53 | 77 |
| 5 | 143 | 91 | 63 | 58 | 84 |
| 6 | 239 | 112 | 74 | 61 | 89 |
| 7 | 411 | 134 | 86 | 62 | 90 |

the condition numbers of $\boldsymbol{\mathcal{S}}_\Lambda$ grow with increasing scale and the iteration numbers roughly double for each mesh bisection. As predicted the preconditioners keep the condition numbers bounded but make them settle on a very high level. Moreover, as expected by the condition numbers for lower scales $\boldsymbol{M}_\Lambda^w$ does not reduce the number of iterations very much. $\boldsymbol{M}_\Lambda^{aJ}$ (as well as $\boldsymbol{M}_\Lambda^J$) lead to a nearly constant iteration number for higher scales. The approximate Jacobi preconditioner $\boldsymbol{M}_\Lambda^{aJ}$ appears to offer the by far best compromise between computational cost and efficiency.

The condition number would predict much higher iteration numbers. However, here the superconvergent behavior of the cg–method plays an important role (Refs.14–15). Figure 3 displays the distribution of eigenvalues for the Schur complement on scale 7. It confirms the expected scale wise increase of the eigenvalues caused by the derivative in the constraints. It also shows that this increase is prevented by the approximate Jacobi preconditioner. As predicted the preconditioner $\boldsymbol{M}_\Lambda^{aJ}$ only affects the eigenvalues for high frequency components. Figure 3 also confirms that the smallest eigenvalue is always close to zero reflecting poor observability of the problem, the effect of the discretization as well as the insufficient amount of regularization. Unfortunately, the smallest eigenvalue is even decreased further by the preconditioner from roughly $10^{-5}$ to $10^{-7}$. Moreover, apart from a few very small eigenvalues the eigenvalues concentrate around one value. For the simplified Uzawa scheme this leads to the superconvergent convergence behavior. While the error reduction in the first iteration steps may be small, subsequently we have a high convergence order since the iterates are close to a subspace which are well separated from the eigenspace of small eigenvalues.

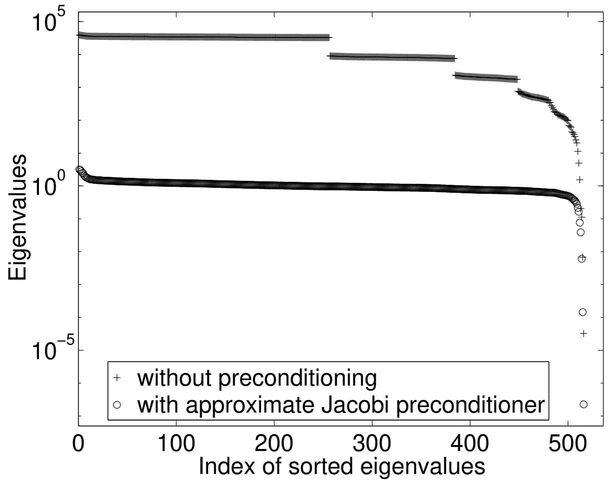


Figure 3: Eigenvalues of $\boldsymbol{\mathcal{S}}_\Lambda$ for the discretization of *Example 2.1*.

The condition numbers of *Example 2.2* do not quite exhibit the same scale dependent increase up to the computed scale 7 (see Table 9). The effect of the differential operator is in this case strongly dominated by the system inherent properties. Thus, up to scale 7 $\boldsymbol{M}_\Lambda^w$ does not have the desired effect, while the other preconditioners incorporate $\boldsymbol{\mathcal{S}}_\Lambda$ and therefore lead to a reduction of the condition numbers. Again, we have not seen any difference between using the exact or the approximated Schur complement ($\boldsymbol{M}_\Lambda^J$ or $\boldsymbol{M}_\Lambda^{aJ}$) for the Jacobi preconditioner, so the results are not listed separately. The actual iteration

Table 9: cond $\left(\boldsymbol{M}_\Lambda^{1/2}\boldsymbol{\mathcal{S}}_\Lambda\boldsymbol{M}_\Lambda^{1/2}\right)$ for various preconditioners, *Example 2.2*.

| Scale | $\boldsymbol{I}$ | $\boldsymbol{M}_\Lambda^{\ell_1}$ | $\boldsymbol{M}_\Lambda^{\ell_2}$ | $\boldsymbol{M}_\Lambda^{aJ}$ | $\boldsymbol{M}_\Lambda^{w}$ |
|---|---|---|---|---|---|
| 1 | 6.6811e+08 | 2.0136e+08 | 2.2661e+08 | 3.3955e+08 | 6.6811e+08 |
| 2 | 1.8370e+08 | 7.3924e+06 | 7.4484e+06 | 1.1337e+07 | 2.0925e+08 |
| 3 | 1.8198e+08 | 3.2145e+06 | 3.5570e+06 | 3.9910e+06 | 3.1657e+08 |
| 4 | 1.8383e+08 | 2.5785e+06 | 3.7069e+06 | 1.2942e+06 | 6.2646e+08 |
| 5 | 1.8631e+08 | 2.6227e+06 | 3.8166e+06 | 8.1949e+05 | 1.2605e+09 |
| 6 | 1.8781e+08 | 2.8957e+06 | 3.7576e+06 | 1.3404e+06 | 1.9685e+09 |
| 7 | 2.1555e+08 | 3.1816e+06 | 3.7365e+06 | 2.2054e+06 | 2.5265e+09 |

numbers are much less than indicated by the condition numbers (see Table 10). This feature we attribute again to the superconvergence behavior of the underlying cg–method. Nevertheless, the superconvergent behavior is not as significant as in *Example 2.1* due to the system inherent difficulties which are reflected in the eigenvalue distributions. Starting at scale 4 the iteration numbers of the preconditioned systems begin to settle at a much lower number than without preconditioning.

Table 10: Iteration numbers for various preconditioners, *Example 2.2.*

| Scale | $\boldsymbol{I}$ | $\boldsymbol{M}_\Lambda^{\ell_1}$ | $\boldsymbol{M}_\Lambda^{\ell_2}$ | $\boldsymbol{M}_\Lambda^{aJ}$ | $\boldsymbol{M}_\Lambda^{w}$ |
|---|---|---|---|---|---|
| 1 | 148 | 122 | 118 | 109 | 148 |
| 2 | 277 | 202 | 207 | 191 | 305 |
| 3 | 527 | 357 | 355 | 293 | 639 |
| 4 | 880 | 490 | 473 | 393 | 948 |
| 5 | 1111 | 600 | 560 | 456 | 1124 |
| 6 | 1297 | 695 | 582 | 470 | 1158 |
| 7 | 1555 | 704 | 601 | 429 | 918 |

In summary, among the tested diagonal preconditioners the approximate Jacobi preconditioner performs best although the results are still unsatisfactory. Essentially two further directions suggest themselves as a remedy.

First, one can resort to more sophisticated preconditioners which require of course also more computational effort. First attempts of employing various incomplete versions of Cholesky factorization to the Schur complement have turned out to fail. For example, the classical incomplete Cholesky factorization based on the sparsity pattern of $\boldsymbol{\mathcal{S}}_\Lambda$ breaks down because positivity of the diagonal entries is lost during the process. The same happens with incomplete block–Cholesky factorization, which means that an exact Cholesky factorization is applied to the block structure, while the inversion of the resulting blocks is 'incomplete'. Exact Cholesky factorization applied to an approximated $\boldsymbol{\mathcal{S}}_\Lambda$, where only the diagonals of the involved wavelet matrices are employed, either also breaks down because of a possible loss of positive definiteness of the approximate Schur complement or it is not efficient. Another attempt to have a better grip on the smallest eigenvalues has been to use a Cholesky factorization of $\boldsymbol{\mathcal{S}}_{\Lambda_{j_0}}$ on a lower scale $j_0$ extended by a scale depending diagonal preconditioning on the higher scales for preconditioning $\boldsymbol{\mathcal{S}}_{\Lambda_J}$ on a higher scale $J$. This turned out to produce even worse results than applying the approximate Jacobi preconditioner $\boldsymbol{M}_\Lambda^{aJ}$. Stabilizing incomplete LU–factorization by pivoting is ruled out because one would loose symmetry which is essential for cg based schemes such as the Uzawa algorithm. There remains the class of preconditioners involving parameters like the incomplete Cholesky factorization based on a given threshold parameter, or also, though not leading to a symmetric preconditioner, SPAI. The problem is then to choose the parameters so as to obtain a good preconditioner while keeping the cost of determining and applying the preconditioner acceptable. In principle, one can choose the parameters in such a way that the inverses are determined as preconditioners. In that case we are back to applying a direct solver. Further less established preconditioners that primarily affect the smallest eigenvalues are currently under investigation.

The second direction to improve the condition number is to modify the operator $\boldsymbol{\mathcal{S}}$ by regularization, i.e. in our case by varying $\varepsilon$.

## 4.5 Varying the Regularization Parameter

The key issue for choosing an appropriate $\varepsilon$ is, of course, to attain satisfactory estimation quality. This requires establishing a good compromise between stabilizing the problem and minimizing the resulting bias of the estimate. To avoid bias to a best possible extent, the effect of the regularization of $\boldsymbol{w}$ should be kept at a minimum. Hence, for regularization purposes $\varepsilon$ should be rather small. Typically, those states directly entering the output equations (3) are estimated with little bias for small $\varepsilon$ due to the high weight of the output prediction errors whereas there are significantly larger estimation errors in the other states. If there are no requirements on the relative estimation error quality of the states dictated by the specific estimation problem, it is reasonable to choose $\varepsilon$ so as to maximize the smallest eigenvalue of the Schur complement $\boldsymbol{\mathcal{S}}_\Lambda$ to reduce the condition number. This results in a smaller norm of the inverse operator given by the original optimization problem and leads to the best parameter under these circumstances. The quantitative effect of this strategy is recorded in Tables 11–15.

Comparing Table 11 and Table 12, we see how the different effects of regularization and preconditioning interact. On one hand, for $\varepsilon \to 0$ the condition numbers of the Schur complement first decrease and then increase again (Table 11). The Schur complement

Table 11: Regularization of *Example 2.1* without preconditioning, condition numbers.

| Scale\\$\varepsilon$ | 100 | 1 | $10^{-2}$ | $5 \cdot 10^{-4}$ | $10^{-6}$ | $10^{-8}$ |
|---|---|---|---|---|---|---|
| 2 | 7.430e+08 | 7.522e+06 | 1.476e+05 | 1.349e+05 | 5.873e+07 | 5.872e+09 |
| 3 | 6.800e+08 | 6.861e+06 | 1.091e+05 | 5.690e+04 | 2.100e+07 | 2.098e+09 |
| 4 | 2.124e+09 | 2.141e+07 | 3.251e+05 | 7.491e+04 | 1.455e+07 | 1.451e+09 |
| 5 | 8.136e+09 | 8.205e+07 | 1.253e+06 | 2.200e+05 | 1.316e+07 | 1.309e+09 |
| 6 | 3.120e+10 | 3.147e+08 | 4.841e+06 | 8.641e+05 | 1.292e+07 | 1.275e+09 |
| 7 | 1.208e+11 | 1.218e+09 | 1.878e+07 | 3.381e+06 | 1.302e+07 | 1.267e+09 |

$\boldsymbol{\mathcal{S}}_\Lambda$ defined in (9) tends to the first summand for $\varepsilon \to 0$ and is in general singular in the limit, since the rank of $\boldsymbol{C}$ is typically smaller than the number of columns. On the other hand, assuming that $\boldsymbol{C}$ only selects several states as outputs (as in *Example 2.1*), decreasing $\varepsilon$ enhances the diagonal entries of $\boldsymbol{\mathcal{S}}_\Lambda$ contributed by the diagonal entries of the first term $\boldsymbol{C}_\Lambda^T \left(\boldsymbol{I}_{n_y} \otimes \langle\langle \boldsymbol{\Psi}_{\Lambda_y}, \boldsymbol{\Psi}_{\Lambda_y} \rangle\rangle\right) \boldsymbol{C}_\Lambda$ in (9). Thus, Jacobi preconditioning becomes the more efficient the more $\varepsilon$ is decreased. The fact that, while the Schur complement becomes more and more ill–conditioned when $\varepsilon \to 0$, the preconditioner works better and better, is confirmed by the results recorded in Table 12. Without preconditioning the best result for *Example 2.1* is obtained for $\varepsilon = 5 \cdot 10^{-4}$ (Table 11), which offers a compromise for small condition numbers while keeping the information of the system to a satisfactory extent.

Table 13 records the iteration numbers obtained when varying $\varepsilon$. The last column lists the corresponding dimension of the solved system. We apply again the *simplified Uzawa* algorithm with approximate Jacobi preconditioning $\boldsymbol{M}_\Lambda^{aJ}$. The overlapping effect of regularization and preconditioning is reflected by the respective iteration numbers. While the original system becomes singular with $\varepsilon \to 0$, with preconditioning less iterations are required. The above suggested choice of $\varepsilon = 5 \cdot 10^{-4}$ together with the yet most promising approximate Jacobi preconditioner reduces the initial number 201 of iterations for scale 7 (Table 2) to 30. Moreover, the latter iteration number seems to stay constant when progressing to higher refinement scales.

Table 12: Regularization of *Example 2.1* with approximate Jacobi preconditioning, condition numbers.

| Scale\\$\varepsilon$ | 100 | 1 | $10^{-2}$ | $5 \cdot 10^{-4}$ | $10^{-6}$ | $10^{-8}$ |
|---|---|---|---|---|---|---|
| 2 | 1.903e+08 | 1.899e+06 | 3.216e+04 | 1.225e+04 | 1.096e+04 | 1.096e+04 |
| 3 | 4.540e+08 | 4.528e+06 | 6.349e+04 | 1.434e+04 | 1.079e+04 | 1.078e+04 |
| 4 | 8.774e+08 | 8.758e+06 | 1.235e+05 | 2.141e+04 | 1.204e+04 | 1.202e+04 |
| 5 | 1.184e+09 | 1.188e+07 | 1.733e+05 | 2.923e+04 | 1.272e+04 | 1.265e+04 |
| 6 | 1.326e+09 | 1.335e+07 | 1.982e+05 | 3.429e+04 | 1.336e+04 | 1.318e+04 |
| 7 | 1.389e+09 | 1.400e+07 | 2.095e+05 | 3.679e+04 | 1.403e+04 | 1.366e+04 |

Table 13: Regularization of *Example 2.1* with approximate Jacobi preconditioning, iteration numbers.

| Scale\\$\varepsilon$ | 100 | 1 | $10^{-2}$ | $5 \cdot 10^{-4}$ | $5 \cdot 10^{-6}$ | dim |
|---|---|---|---|---|---|---|
| 2 | 37 | 23 | 20 | 20 | 20 | 41 |
| 3 | 58 | 41 | 35 | 28 | 29 | 77 |
| 4 | 73 | 53 | 37 | 31 | 31 | 149 |
| 5 | 82 | 58 | 38 | 30 | 30 | 293 |
| 6 | 88 | 61 | 38 | 30 | 29 | 581 |
| 7 | 90 | 62 | 39 | 30 | 28 | 1157 |

Tables 14 and 15 list the condition and iteration numbers corresponding to *Example 2.2*. We normalized the states and outputs and applied the preconditioner $\boldsymbol{M}_{\Lambda}^{aJ}$. Here

Table 14: Regularization of *Example 2.2*, condition numbers.

| Scale\\$\varepsilon$ | 100 | 1 | $10^{-3}$ | $10^{-4}$ | $10^{-6}$ |
|---|---|---|---|---|---|
| 1 | 1.5621e+13 | 1.5696e+11 | 5.8125e+08 | 3.3955e+08 | 4.4458e+08 |
| 2 | 4.7847e+12 | 4.7897e+10 | 5.3420e+07 | 1.1337e+07 | 7.3007e+07 |
| 3 | 2.3332e+12 | 2.3347e+10 | 2.6321e+07 | 3.9910e+06 | 7.7545e+07 |
| 4 | 8.6843e+11 | 8.6874e+09 | 8.6592e+06 | 1.2942e+06 | 8.4559e+07 |
| 5 | 4.9074e+11 | 4.9106e+09 | 4.7343e+06 | 8.1949e+05 | 8.7582e+07 |
| 6 | 1.0938e+12 | 1.0946e+10 | 1.0576e+07 | 1.3404e+06 | 8.8499e+07 |
| 7 | 1.9191e+12 | 1.9201e+10 | 1.8251e+07 | 2.2054e+06 | 8.8698e+07 |

regularization and Jacobi preconditioning do not have the same interfering effect as in *Example 2.1*. This is explained by the model matrix $\boldsymbol{C}$ which in this case arises from model reduction and therefore determines the outputs by combinations of several states. Therefore its structure does not lead to a diagonally dominant behavior of $\boldsymbol{\mathcal{S}}_{\Lambda}$ for small $\varepsilon$. The best choice of $\varepsilon$ with respect of the condition numbers has been found in numerical experiments to be roughly $10^{-4}$, while $\varepsilon = 10^{-3}$ turns out to provide the lowest number of iterations. So far, for *Example 2.2* only the *simplified Uzawa* algorithm works reasonably well (Table 5). The combination of normalization, approximate Jacobi preconditioning and simple regularization reduces significantly the condition numbers from initially $10^{20}$ to $10^{6}$. For this extremely ill–conditioned problem, neither iterative nor direct methods

Table 15: Regularization of *Example 2.2*, iteration numbers.

| Scale\\$\varepsilon$ | 100 | 1 | $10^{-3}$ | $10^{-4}$ | $10^{-6}$ | dim |
|---|---|---|---|---|---|---|
| 1 | max | 185 | 98 | 109 | 174 | 75 |
| 2 | max | 394 | 149 | 191 | 384 | 133 |
| 3 | max | 565 | 224 | 293 | 925 | 249 |
| 4 | 1497 | 765 | 306 | 393 | 1779 | 481 |
| 5 | 1606 | 871 | 349 | 456 | 2507 | 945 |
| 6 | 1721 | 911 | 365 | 470 | 2903 | 1873 |
| 7 | 1753 | 970 | 329 | 429 | max | 3729 |

could have been reliably applied without preconditioning. Moreover, the superconvergence of the underlying cg–method also leads for this example to much lower iteration numbers than predicted by the condition numbers. The iteration numbers stay roughly around 350 independently of the refinement scale (see Table 15).

# 5 Conclusions

In summary, for the wavelet discretization of the optimization problem (1)–(3), the most efficient method tested here is the *simplified Uzawa* algorithm with *approximate Jacobi* preconditioner $\boldsymbol{M}_{\Lambda}^{aJ}$ applied to the system red$_2$ in combination with normalization of the states and outputs and with an appropriate choice of the parameter $\varepsilon$. The cost of one iteration of the simplified Uzawa algorithm corresponds to one iteration of the cg–method. Moreover, the number of iterations is less than for the application of various other, more expensive Krylov–space methods like *symmlq, bicg* and *gmres*. The combination of normalization, simple regularization and approximate Jacobi preconditioning gives rise to significantly reduced condition numbers. More precisely, the approximate Jacobi preconditioner eliminates the scale wise increase of the relevant eigenvalues and bounds the condition number independently of the employed index set $\Lambda$. On the other hand, normalization and simple regularization increases the smallest eigenvalues. Depending on the model matrix $\boldsymbol{C}$ we can observe an interfering effect of preconditioning and regularization with respect to the condition numbers. The choice of the regularization parameter $\varepsilon$ should therefore not be guided by the preconditioned systems. In above cases $\varepsilon$ ranging between $10^{-3}$ and $10^{-4}$ turned out to be suitable. Nevertheless, depending on model inherent features the condition numbers may still be too large. More sophisticated methods have to be developed to raise the smallest eigenvalues. Currently, a Schur complement method is under investigation. In spite of the possibly still large condition numbers the superconvergent behavior of the underlying cg–method yields considerably smaller numbers of iterations. This stems from a clustering of eigenvalues, which is caused by the approximated Jacobi preconditioning. Last but not least the application of an iterative solver does not require assembling the discretization matrices. Only their application to a vector has to be carried out by exploiting the Kronecker product rules.

# References


[1] BINDER, T., and BLANK, L., and DAHMEN, W., and MARQUARDT, W., *Iterative Algorithms for Multiscale State Estimation, Part1: Concept,* Journal of Optimization Theory and Applications (to appear), 2001.

[2] BINDER, T., and BLANK, L., and DAHMEN, W., and MARQUARDT, W., *Towards Multiscale Dynamic and Data Reconciliation,* Nonlinear Model Based Process Control, Edited by Berber, R., and Kravaris, C., NATO ASI Series, Kluwer Academic Publishers, Dordrecht, the Netherlands, pp.623-665, 1998.

[3] ALLGÖWER, F., and BADGWELL, T.A., and QIN, J.S., and RAWLINGS, J.B., and WRIGHT, S.J., *Nonlinear Predictive Control and Moving Horizon Estimation – An Introductory Overview,* Advances in Control: Highlights of ECC 1999, Edited by Frank, P.M., Springer, New York, New York, pp.391-449, 1999.

[4] BINDER, T., and BLANK, L., and DAHMEN, W., and MARQUARDT, W., *On the Regularization of Dynamic Data Reconciliation Problems,* Journal of Process Control, (to appear), 2001.

[5] KAILATH, T., *Linear Systems,* Prentice Hall, Englewood Cliffs, New Jersey, 1980.

[6] WALDRAFF, W., and DOCHAIN, D., and BOURREL, S., and MAGNUS, A., *On the Use of Observability Measures for Sensor Location in Tubular Reactors,* Journal of Process Control, Vol.8, pp.497-505, 1998.

[7] BINDER, T., and BLANK, L., and DAHMEN, W., and MARQUARDT, W., *Adaptive Multiscale Approach to Real–Time Dynamic Optimization,* Report LPT–1999–14, RWTH Aachen, Germany 1999.

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

[9] DAHMEN, W., *Wavelet and Multiscale Methods for Operator Equations,* Acta Numerica, Vol.7, pp.55-228, 1997.

[10] DAUBECHIES, I., *Ten Lectures on Wavelets,* SIAM, Philadelphia, Pennsylvania 1992.

[11] BARRETT, R., and BERRY, M., and CHAN, T.F., and DEMMEL, J., and DONATP, J., and EIJKHOUT, V., and POZO, R., and ROMINE, C., and VAN DER VORST, H., *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods,* SIAM, Philadelphia, Pennsylvania, 1994.

[12] FREUND, R.W., and GOLUB, G.H., and NACHTIGAL, N.M., *Iterative Solution of Linear Systems,* Acta Numerica, Vol.2, pp.57-100, 1992.

[13] PAIGE, C.C., and SAUNDERS, M.A., *Solution of Sparse Indefinite Systems of Linear Equations,* SIAM Journal on Numerical Analysis, Vol.12, pp.671-629, 1975.

[14] AXELSSON, O., *Iterative Solution Methods,* Cambridge University Press, New York, New York, 1994.

[15] MEURANT, G., *Computer Solution of Large Linear Systems,* Elsevier Science, Amsterdam, the Netherlands, 1999.