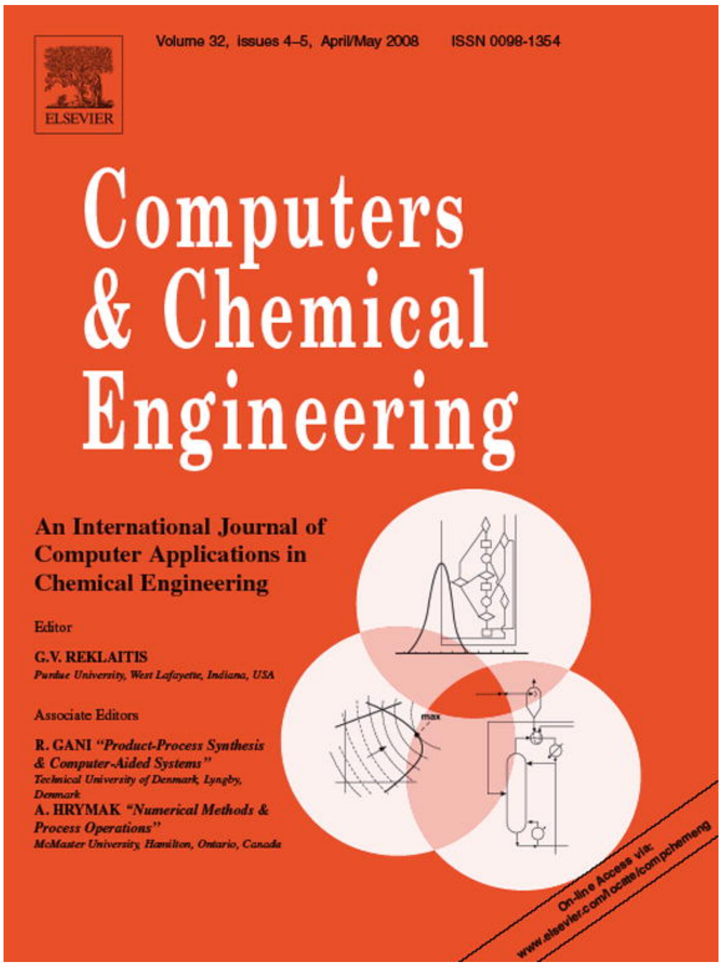
Volume 32, Issues 4–5, April/May 2008 ISSN 0098-1354
ELSEVIER
Computers & Chemical Engineering
An International Journal of Computer Applications in Chemical Engineering
Editor
G.V. REKLAITIS
Purdue University, West Lafayette, Indiana, USA
Associate Editors
R. GANI "Product-Process Synthesis & Computer-Aided Systems"
Technical University of Denmark, Lyngby, Denmark
A. HRYMAK "Numerical Methods & Process Operations"
McMaster University, Hamilton, Ontario, Canada
On-line Access via: www.elsevier.com/locate/compchemeng

Available online at www.sciencedirect.com
ScienceDirect
Computers and Chemical Engineering 32 (2008) 990–999
Computers & Chemical Engineering
www.elsevier.com/locate/compchemeng

Scaled steady state models for effective on-line applications

Tore Lid[a], Sigurd Skogestad[b,*]

[a] *Statoil Mongstad, 5954 Mongstad, Norway*
[b] *Department of Chemical Engineering, Norwegian University of Science and Technology (NTNU), Trondheim, Norway*

Received 19 June 2006; received in revised form 29 March 2007; accepted 9 April 2007
Available online 18 April 2007

Abstract

Applications for on-line data reconciliation and optimization must be efficient and numerically robust. The models in these applications are rarely changed and the same optimization problem is solved thousands of times with only minor changes in the parameters. This paper describes a suitable modeling framework for this type of applications that, with the aim of simplifying the creation of new models, makes the application robust and avoids numerical difficulties. The model is based on a unit model structure where first-order derivatives, scaling and initial values are properties of the unit model. A new scaling procedure is proposed based on equation and variable pairing. The modeling framework and the use of the proposed scaling procedure are demonstrated in two case studies, case 1 is simulation of a simple pipe model, case 2 is simulation, data reconciliation and optimization of a flash process.
© 2007 Elsevier Ltd. All rights reserved.

Keywords: Optimization; Data reconciliation; Simulation; Unit models; Scaling

1. Introduction

Typical process modeling tools are based on a unit model structure library, and then using streams to connect these. Unit models typically included are heaters, flash drums, heat exchangers, distillation columns, reactors and so on (Westerberg, Hutcison, Motard, & Winter, 1979). The resulting model equations are solved sequentially or simultaneously.

Most chemical engineers prefer tools like PRO/II from SIMSCI and Hysys from AspenTech. This may be due to an extensive unit model library, a high quality user interface and a sequential solver that solves one unit model at a time. In this environment it is simple to locate a problem (like a non-converging unit model) and it is simple to do changes to the model on the worksheet level. On the other hand, sequential solvers are ineffective for solving optimization problems, including data reconciliation.

For optimization problems, as well as for simulation of more complex processes with energy and mass recycles, simultaneous solvers are preferred. Examples of tools for process modeling using simultaneous solvers are gProms from PSE, ASCEND from Carnegie Mellon University and Custom Modeler from AspenTech. See Marquardt (1996) for an overview of these tools and others.

The strength of the generic modeling tools mentioned above are the modeling capability, i.e. creation of new models, but this is rarely needed in on-line optimization applications. On-line optimization of a process plant is typically separated into three main tasks: estimation of current state (data reconciliation), optimization and implementation (White, 1997). Models for on-line applications should be derived with the following in mind:

- An optimization problem may be solved thousands of times a year with only small changes in objective functions and specifications and the models are only rarely changed. Changes in the model are only required when the plant is modified which may be only once every 2–10 years.
- The execution of the optimizer is often automated and is generally not monitored by modeling experts. Robust convergence properties of the solver are critical.
- The optimizer must have on-line data exchange with the control and process planning systems. It is therefore often run on computers closely connected to the control system with limited access for changes.

In summary, the requirements for an on-line application are a model with no overhead (unused functionality) to save

* Corresponding author. Tel.: +47 7359 4154.
E-mail address: skoge@chemeng.ntnu.no (S. Skogestad).

0098-1354/$ – see front matter © 2007 Elsevier Ltd. All rights reserved.
doi:10.1016/j.compchemeng.2007.04.003

computation time, an effective and robust solver and simple interfaces to other systems for data transfer. The actual application is typically “tailor made” and programmed in some object oriented programming language (C++ or similar).

This paper demonstrates a modeling procedure for this type of on-line applications. Our experience is that too much time in such projects is spent on finding model errors and avoiding numerical difficulties and too little time on result analysis. This modeling guideline will hopefully improve this. The models are based on a unit model structure and solved simultaneously using a general NLP (nonlinear programming) solver. The equations and variables are organized such that the same process model is used for simulation, data reconciliation and optimization of the process.

Model residuals, first-order derivatives of the models, scaling factors and initial values, are properties of the unit model. The model equations and numerical properties of each unit model are verified before they are added to the process model. The unit model equations are standardized to reduce the possibility of errors and simplify the modeling work. For example, all mass balances have the same structure, similar scaling and same engineering units. This simplifies the development of new unit models and reduces the possibility of errors.

The examples given in this paper are simple, but the procedure has been applied industrially on a crude unit pre-heater train (Lid, Skogestad, & Strand, 2002) where the resulting on-line application is still operating after several years. It has also successfully been applied to a naphtha reformer model with more than 500 equations and variables.

The model representation in this paper is very simple and a comprehensive definition, more suited for commercial use, can be found in Bogusch and Marquardt (1995).

In this paper all models are steady state, which is suitable for most process plants with continuous operation. In the case of processes where dynamic changes are central, the use of a dynamic model should be considered.

The most important notation is summarized in Table 1.

Table 1
Nomenclature

	Description	Dimension
z	Process model variables	$n_z \times 1$
$\tilde{z}$	Scaled process model variables	$n_z \times 1$
n_z	Number of process model variables	
f	Process model equations	$n_f \times 1$
$\tilde{f}$	Scaled process model equations	$n_f \times 1$
r	Residual vector	$n_f \times 1$
n_f	Number of process model equations	
J	Objective to be minimized	
y	Measurement vector	$n_y \times 1$
Q	Measurement weighting matrix	$n_y \times n_y$
U	Measurement incident matrix	$n_y \times n_z$
P_n	Equation and variable pairing matrix	$n_f \times n_z$
P_s	Equation and variable pairing matrix	$n_\mathrm{s} \times n_z$
n_y	Number of measurements	
A_s	Fixed values matrix	$n_\mathrm{s} \times n_z$
b_s	Vector of fixed values	$n_\mathrm{s} \times 1$
n_s	Number of specified variables	
A_r	Fixed values matrix	$n_\mathrm{r} \times n_z$
b_r	Vector of fixed values	$n_\mathrm{r} \times 1$
n_r	Number of specified variables	
A_opt	Fixed values matrix	$n_\mathrm{opt} \times n_z$
b_opt	Vector of fixed values	$n_\mathrm{opt} \times 1$
n_opt	Number of specified variables	
z	Model variables	$n_z \times 1$
z_s	Simulation result	$n_z \times 1$
z_r	Data reconciliation result	$n_z \times 1$
z_opt	Optimization result	$n_z \times 1$
z_0	Initial value	$n_z \times 1$
p	Cost vector	$n_z \times 1$
S_n	Nonlinear equations scaling matrix	$n_f \times n_f$
S_l	Linear equations scaling matrix	$n_\mathrm{s} \times n_\mathrm{s}$
S_v	Variable scaling matrix	$n_z \times n_z$
S_o	Objective scaling factor	
H	Linearized equality constraints	
$\tilde{H}$	Scaled linearized equality constraints	
d_est	Estimation error	
Init.	Initial values	
Sim.	Simulation results	
Rec.	Data reconciliation results	
Opt.	Optimization results	

2. Simulation, data reconciliation and optimization problems

This section defines the simulation, data reconciliation and optimization problems considered in this work.

All three problems use a nonlinear steady state model of the process, which is incorporated as a set of nonlinear equality constraints $f(z) = 0$. In addition, known variables are specified by linear equality constraints $Az = b$. For each specification i, the matrix A has a row $A(i)$ with a single non-zero element $A(i, j) = 1$, such that the value of $z(j)$ is specified to equal $b(i)$.

The number of equations in the process model ($f(z) = 0$, $Az = b$) should be less than the number of variables, i.e. $n_f < n_z$. The difference $n_z - n_f - n_\mathrm{s}$ is the number of degrees of freedom for the problem.

2.1. Simulation

In the simulation case, specifications are added in A_s such that there are zero degrees of freedom, i.e. $n_z - n_f - n_\mathrm{s} = 0$. The simulation problem is formally defined as

$$\min_z \quad J_\mathrm{s}(z) \qquad \text{s.t.} \quad f(z) = 0, \quad A_\mathrm{s} z = b_\mathrm{s} \tag{1}$$

where the “dummy” objective function is chosen as $J_\mathrm{s}(z) = 0$. This is because with no degrees of freedom the objective function has no influence on the solution. Note that the specifications in $A_\mathrm{s} z = b_\mathrm{s}$ must be selected such that there are no dependent equations in $f(z)$ and A_s, that is such that the matrix

$$\begin{bmatrix} \dfrac{\partial f}{\partial z} \\ A_\mathrm{s} \end{bmatrix} \tag{2}$$

has full rank.

2.2. Data reconciliation

Data reconciliation is used to estimate the actual condition of the process and is obtained as the solution of

$$\min_z \quad J(z) \qquad \text{s.t.} \quad f(z)=0, \quad A_r z = b_r, \qquad z_{r\,\min} \le z \le z_{r\,\max} \tag{3}$$

where $J=(y-Uz)^{\mathrm{T}}Q(y-Uz)$. All n_y measurements are collected in the measurement vector y. The "selection" matrix U gives a mapping of the variables z into the measurements, such that Uz represents the estimated value of the measurements y. The matrix U has n_y rows and in each row there is only one non-zero element $U(i,j)=1$, that is $y(i)$ corresponds to $z(j)$.

The diagonal weighting matrix Q has elements $Q(i,i)$ equal to $1/\sigma(i)^2$, where $\sigma(i)^2$ is the variance of the measurement noise of measurement number i. Minimizing the objective function is the same as maximizing the Gaussian frequency function, $\sum_i f_i = 1/(\sigma(i)\sqrt{2\pi})\exp(-0.5(y(i)-U(i)z)^2/\sigma(i)^2)$, which results in a least squares or maximum likelihood estimate of the process state. More about this and other objective functions can be found in Tjoa and Biegler (1991) and Chen, Pike, Hertwig, and Hopper (1998).

Upper and lower bounds on variables are used to limit the solution to acceptable values. For example all flows, temperatures and pressures must satisfy $z(j)\ge 0$.

If the value of a variable is known it can be specified using the linear constraints.

The variables must be observable based on the measured values and the process model (Stanley & Mah, 1981). A minimal requirement is that the number of measurements satisfies $n_y > n_z - n_f - n_r$, where n_r is the number of rows in A_r. If some variables are not observable then measurements must be added or the actual variable value must be specified.

2.3. Optimization

Optimal operation is calculated by minimization of a cost function subject to the process model, specified values and operating constrains:

$$\min_z \quad J(z) \qquad \text{s.t.} \quad f(z)=0, \quad A_{\mathrm{opt}} z = b_{\mathrm{opt}}, \qquad z_{\mathrm{opt\,min}} \le z \le z_{\mathrm{opt\,max}} \tag{4}$$

where $J(z)=p(z)^{\mathrm{T}}z$. In most cases p is a vector of fixed prices related to feed cost, energy cost and product values.

Values for variables like model parameters, feed conditions and other variables, not available for optimization, are specified using the linear equality constraints $A_{\mathrm{opt}}z=b_{\mathrm{opt}}$. These variables are set equal to the reconciled variable, $b_{\mathrm{opt}}=A_{\mathrm{opt}}z_{\mathrm{r}}$.

Operating constraints are added as upper and lower bounds on variables, $z_{\mathrm{opt\,min}}$ and $z_{\mathrm{opt\,max}}$.

2.4. NLP solver

A NLP solver is used for solving the simulation, data reconciliation and optimization problems. In this paper a general NLP solver is required to at least handle the following optimization problem definition:

Objective to be minimized	$J(z)$
Linear equality constraints	$Az=b$
Nonlinear equality constraints	$f(z)=0$
Variable bounds	$z_{\min} < z < z_{\max}$

In addition it is expected to be able to utilize user specified first-order derivatives of the objective and of the nonlinear constraint functions.

Objective first-order derivatives	$\dfrac{\partial J(z)}{\partial z}$
NL constraints first-order derivatives	$\dfrac{\partial f(z)}{\partial z}$

The linearized equality constraints

$$H=\begin{bmatrix} \dfrac{\partial f(z)}{\partial z} \\ A \end{bmatrix} \tag{5}$$

are used for analysis of the numerical properties of the optimization problem. If the condition number of H is large, then the problem is said to be ill-conditioned and numerical problems may be expected (here large means $>10^6$).

In this paper the solver `fmincon` from the Matlab Optimization Toolbox® is used.

3. Modeling framework

3.1. Model structure

In the suggested model structure, a process model is a collection of one or more unit models. A unit model describes a small part of the process like a flash drum, heater or a reactor. The boundary of the unit model is selected such that the connection to other unit models is by process streams. A general unit model, as shown in Fig. 1, can have one or more input and output streams, shown as $S_1 - S_N$ and internal variables shown as Θ_i. A process stream, connecting two unit models, is simply a set of shared variables describing the properties of the process stream. A process model with three unit models and seven process streams is shown in Fig. 2. Each unit model has a set of equations $f_i(z)=0$ and the overall process model is a collection of equations from these unit models:

$$r=f(z)=\begin{bmatrix} f_1(z) \\ f_2(z) \\ f_3(z) \end{bmatrix} \tag{6}$$

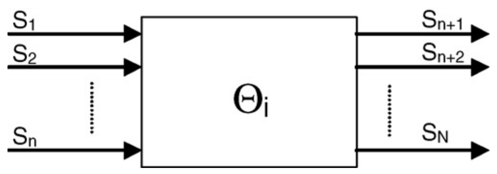

Fig. 1. Unit model.

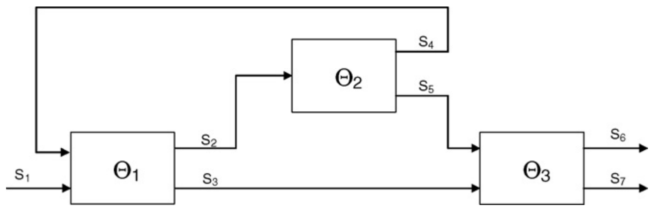

Fig. 2. Process model.

The process model, $r = f(z)$, as shown in Eq. (6), is a collection of unit models where each unit model is represented by equations written as $r_i = f_i(z)$.

All unit models share the variable vector z. This variable vector contains variables from all process streams and internal variables from all unit models:

$$z = \begin{bmatrix} S_1 \\ S_2 \\ \vdots \\ S_7 \\ \Theta_1 \\ \Theta_2 \\ \Theta_3 \end{bmatrix} \tag{7}$$

Variables describing a process stream are typically component molar fractions, flow, temperature and pressure. Some cases may require other variables. For example, in units with two-phase streams, enthalpy may replace temperature as a variable. In this paper process stream variables are selected as $S_i = [\boldsymbol{x}_j^{\mathrm{T}} \quad F_j \quad T_j \quad P_j]^{\mathrm{T}}$

The unit model internal variables Θ_i, can be heater duty, heat transfer coefficient and compressor efficiency.

With this fixed ordering of the variables in the variable vector z, a variable mapping is created. The variable mapping is used to obtain the values of input and output stream variables and internal variables from the variable vector z, within each unit model. This requires that the stream number of the input and output streams is known within each unit model. The stream numbers can be passed to the unit model as parameters in the actual function call.

The first-order derivatives of the process model are also calculated on a unit model basis:

$$\frac{\partial f(z)}{\partial z} = \begin{bmatrix} \dfrac{\partial f_1(z)}{\partial z} \\ \dfrac{\partial f_2(z)}{\partial z} \\ \dfrac{\partial f_3(z)}{\partial z} \end{bmatrix} \tag{8}$$

where $\partial f_i(z)/\partial z$ is a $nf_i \times n_z$ matrix. The above mentioned variable mapping is used in the column mapping of the individual elements in $\partial f_i(z)/\partial z$.

3.2. Unit models

A unit model describes the behavior of some process unit or process equipment and is based on equations of mass balance, energy balance and pressure-flow relations. Even if the individual units may be different, the equations describing their behavior is very similar and there are benefits of standardization of these equations.

The simplest unit model possible is a unit model with one inlet stream and one outlet stream. The unit model has no holdup, no reactions, no heat loss or pressure drop. It is visualized as a "pipe model" (Fig. 3) and is stated in Eq. (21).

This "pipe model" is of no practical use as a unit model but works well as a basic template for other unit models. Some examples:

- A heater unit model can be made by adding a simple heat input term Q in the energy balance. The heat input is an internal variable in the model.
- A heat exchanger can be made by combining two pipes. The energy balance in the two models is modified by adding a heat term, one negative and one positive. One additional equation is added in the models describing the heat transfer. This can be based on log mean temperature difference (LMTD), ϵ-Ntu or other.
- A CSTR reactor can be made by adding a reaction term, $VN^{\mathrm{T}}r$, in the pipe model mass balance, where V is the reactor volume, N the reaction stoichiometric matrix and r is a vector of reaction rates.
- A flash drum can be made as a pipe with two outlet streams, one vapor flow and one liquid flow. Equations for vapor–liquid equilibrium ($y - K(T, P)x = 0$), sum of vapor components, equal vapor–liquid pressure and temperature have to be added.

The idea in Section 4 is to develop a "pipe model" with good numerical properties to serve as a template. Other unit models will then inherit these properties and only small adjustments will be necessary.

In formulating models, it is easy to miss an equation. A general recommendation or rule in modeling is to use set assignment and formally pair equations and variables. Since most variables appear in more than one equation this pairing is not unique. Nevertheless this rule gives a valuable overview of the model and the pairing turns out to be useful in adding proper variable specifications and scaling of the variables and equations.

A systematic approach to the equation–variable pairing is found in Maurya, Rengaswamy, and Venkatasubramanian (2003) and Mah (1990) where the equations and variable are defined as nodes in a graph. The equations and variables are grouped into two disjoint subsets where arcs connect the variables and equations. If all equation nodes are connected to only one variable node and no node is left unmatched, the set of equations and variables is said to have perfect matching.

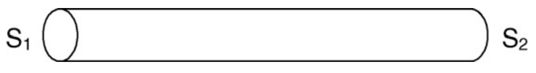

Fig. 3. Pipe.

3.3. Initial values

When creating a large process model it is reasonable to start with a small part of the process, verify the results, and then add more process streams and process units until the total model is finalized.

In this construction process the variable vector z will vary in size and the position of the individual variables in z will change and the generation of initial values, z_0, will be a tedious task.

A simple solution is to let all unit models generate initial values for all unit model internal variables and for variables related to unit model exit streams. This may not result in initial values close to a solution but it may still be sufficient as a starting value for generation of an improved set of initial values by simulation. In addition, initial values for feed streams must be generated.

In a steady state model, assuming no multiple steady-states, the initial value may influence convergence properties but has no influence on the final solution itself. This is different for differential algebraic (DAE) systems where the initial value affects the solution and must be a valid solution of the DAE system at $t=0$ (Pantelides, 1988). In this case the method described above may be an initial value for solving the DAE system at $t=0$.

3.4. Scaling

The performance of the NLP solver depends crucially on how the problem is formulated and an important issue is proper scaling. Note that the scaling is performed off-line. Thus, the computational complexity of the scaling itself is not important. Rather, the objective of the scaling is to minimize the computation time and robustness for the subsequent on-line computations.

An unconstrained optimization problem is said to be poorly scaled if a change in x in one direction produces a much larger change in $f=f(x)$ than in another direction (Nocedal & Wright, 1999). The measure of poor scaling is not so clear in constrained optimization. Some of the methods are said to be scaling invariant, like the SQP algorithm with BFGS update of the Hessian (as used in this paper), but they are still influenced by scaling (Biegler & Cuthrell, 1985). This is related to two issues. First, the initial value of the Hessian is normally set equal to the identity matrix. If the true Hessian of the scaled problem is closer to the identity matrix than the unscaled model this should result in an improved estimate of the Hessian and improved performance of the algorithm. Second, a poorly scaled model is likely to generate larger rounding errors which may degrade the performance of the algorithm.

Scaling methods used within or as a part of a NLP solver are in general based on properties of the estimated Hessian (Roma, 2005; Zhu, 2005). The scaling methods related to the process model or constraints are based on residuals, variable values and first-order derivatives (Jacobian).

A scaled process model is written as

$$\tilde{f}(\tilde{z}) = 0 \tag{9}$$

$$\tilde{A}_s\tilde{z} = \tilde{b} \tag{10}$$

where the scaled variable $\tilde{z} = S_v^{-1}z$. The scaled model $\tilde{f}(\tilde{z}) = S_n f(S_v\tilde{z})$ and for the scaled specification $\tilde{A}_s = S_l A_s$ and $\tilde{b} = S_l b_s$. S_l, S_n and S_v are fixed diagonal scaling matrices.

The scaled objective function $\tilde{J}(\tilde{z}) = S_o J(S_v\tilde{z})$ where S_o is a fixed factor. Three methods for scaling found in literature are in the following sections.

3.4.1. Method 1

Scaling based on variable bounds and initial equation residual (Biegler & Cuthrell, 1985):

$$S_{v_{jj}} = 2^{a_j} \quad \text{where } a_j = \text{int}[\log_2(z_{\max_i} - z_{\min_i})] \tag{11}$$

$$S_{n_{ii}} = 2^{-a_i} \quad \text{where } a_i = \text{int}[\log_2(|f(z_0)|_i)] \tag{12}$$

$$S_{l_{ii}} = 2^{-a_i} \quad \text{where } a_i = \text{int}[\log_2(|A_s z_0 - b_s|_i)] \tag{13}$$

where z_0 is the initial value. The equation scaling factor is limited to some maximum value in case the equation residual is close to zero. More details and suggested improvements can be found in the reference.

3.4.2. Method 2

Scaling based on first-order derivatives (Kelly, 2004):

$$\mathbb{C} = \begin{bmatrix} \dfrac{\partial f(z_0)}{\partial z} \\ A_s \end{bmatrix} \tag{14}$$

$$S_{v_{jj}} = ||\mathbb{C}_j||_2^{-1} \quad \text{where } j = 1, \ldots, n_z \tag{15}$$

$$S_{n_{ii}} = ||\mathbb{C}_i||_2^{-1} \quad \text{where } i = 1, \ldots, n_f \tag{16}$$

$$S_{l_{ii}} = ||\mathbb{C}_i||_2^{-1} \quad \text{where } j = n_f + 1, \ldots, n_f + n_s \tag{17}$$

where $\mathbb{C}_j$ and $\mathbb{C}_i$ denotes the columns and rows of $\mathbb{C}$, respectively. Other norms like the 1-norm ($||\cdot||_1$) or the infinity norm ($||\cdot||_\infty$) may also be used.

3.4.3. Method 3

Scaling based on order of magnitude (Rodriguez-Toral, Morton, & Mitchell, 2001):

$$S_{v_{jj}} = 10^{-a_j} \quad \text{where } a_j = \text{int}[\log_{10}(z_0)_j] \tag{18}$$

The equation scaling factor is the reciprocal of an integer power of 10 of the value of a given term or group of terms, normally related to the scale factor of a relevant variable. As an example, let a typical value of a mass balance term x_iF be $0.5 \times 0.3 = 0.15$. The scaling factor for the mass balance equation is then $10^{(-\text{int}(\log_{10}(0.15)))} = 10$. The objective scaling factor is divided by an integer power of 10 close to its typical value.

3.4.4. Method 4

New proposed scaling method based on variable and equation pairing.

This new scaling method is similar to method number 3 but uses to a larger extent the structure of the model. The equation scaling factors are not based on the constraint term values but on values of the first-order derivatives matrix. The proposed scaling procedure is

1. Make a pairing of equations and variables:
   The equation and variable pairing is given in the matrix P where $P(i, j) = 1$ if variable number j is paired with equation number i. All other elements in P are zero.
   The equation and variable pairing for a unit model $f_i(z)$ is given in a matrix P_{ni} of dimension $n_{f_i} \times n_z$ and variable pairing for the specifications $A_s z = b$ is stated in P_s of dimension $n_s \times n_z$.
2. Scale all variables such that the scaled variable has a value close to one.
   The variable scaling matrix $S_v(j, j) = \bar{z}_j$ where $\bar{z}_j$ is a typical value of variable number j. The initial value, z_0, is used in this case.
3. Scale all equations such that the absolute value of the elements of the first-order derivatives, corresponding to the equation and variable pairing, is close to one:
   $$S_{ni} = \left| \left[I \times \left(\frac{\partial f_i(z)}{\partial z} S_v P_{ni}^T \right) \right]^{-1} \right| \tag{19}$$
   $$S_l = |[I \times (A_s S_v P_l^T)]^{-1}| \tag{20}$$
   where $\times$ denotes element by element multiplication so that S_{ni} and S_l are diagonal matrices.
4. The scaling factor for the objective function S_0 is selected such that the largest element of the first-order derivative $\tilde{J}(\tilde{z})$ has an absolute value close to one.
5. If any of the elements in the matrix $\tilde{H} = [\tilde{f}(\tilde{z})^T \tilde{A}_s^T]^T$ have large absolute values (where large is >100) then the equation and variable pairing or variable scaling should be revised. A possible solution is to pair the equation with the variable corresponding to the large value in $\tilde{H}$.

In order to illustrate the idea of this scaling strategy, assume that the variables and equations are reordered such that the elements along the diagonal of the first-order derivatives correspond to the selected equation–variable pairing. The diagonal elements of this matrix are now all equal to one and the off-diagonal elements are preferably smaller than one. With this scaling the set of constraints will be balanced where a change in one variable will result in a change of same magnitude in the equation residual.

The condition number of $\tilde{H}$ is used as a measure of improved scaling. This measure is based on the definition of poor scaling in the unconstrained case where a change in the variable vector $\boldsymbol{z}$ in one direction produces a much larger change in the residual $r = f(z)$ than in another direction.

A process model with a large condition number of the first-order derivatives will have larger rounding errors. If the matrix of first-order derivatives $\partial f/\partial z$ has a high condition number a small change in Δz, caused by rounding errors, may cause a large change in $\Delta \boldsymbol{r}$.

The objective function scaling factor, S_o, has a large influence on the solution path of the solver during the iterations. A large scaling factor gives large deviations in the model equations in the solution path and rapid decrease in the objective. In case of numerical problems, like temporarily negative values of flows and compositions, the scaling factor of the objective function should be reduced.

4. Case study 1: "Pipe model"

A simple model of a pipe, as described in Section 3.2, demonstrates the use of the suggested modeling procedures. This model has two process streams, one inlet stream and one outlet stream. The fluid is a mixture of two components, propane and butane (NC = 2). The variables are the composition, flow, temperature and pressure of the two process streams. The variable vector organized as $z^T = [\, S_1^T \quad S_2^T \,]$ where $S_i^T = [\, \boldsymbol{x}_i^T \quad F_i \quad T_i \quad P_i \,]$. There are no internal variables in this model.

The equations of the pipe model are written as

$$F_1 \boldsymbol{x}_1 - F_2 \boldsymbol{x}_2 = 0, \qquad \Sigma \boldsymbol{x}_2 - 1 = 0,$$
$$F_1 h(T_1, \boldsymbol{x}_1) - F_2 h(T_2, \boldsymbol{x}_2) = 0, \qquad P_1 - P_2 = 0 \tag{21}$$

These equations represent the mass balance, mole fraction summation, energy balance and pressure-flow relation (with no pressure drop in this case).

The pipe model is in this case unit model number 1 and is in short-hand notation written as $f_1(z) = 0$.

The number of variables in the variable vector z is $2(\mathrm{NC} + 3) = 10$, with NC = 2 and the number of equations in the pipe model is NC + 3 = 5. In order to solve the model equations, as in the simulation case, NC + 3 = 5 variables have to be specified. In this case the inlet stream molar fraction, flow, temperature and outlet stream pressure are specified:

$$\boldsymbol{x}_1 = \boldsymbol{x}_s \tag{22}$$
$$F_1 = F_s \tag{23}$$
$$T_1 = T_s \tag{24}$$
$$P_2 = P_s \tag{25}$$

The specifications are implemented as linear constrains $A_s z = b_s$, where A_s has n_z columns and $n_s = 5$ rows, one row for each specification. A_s is written as

$$A_s = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{26}$$

The values of the specific variables are collected in b_s and $b_s^T = [\, \boldsymbol{x}_s^T \quad F_s \quad T_s \quad P_s \,]$. The specification values $F_s = 0.27$ kmol/s, $T_s = 285$ K, $P_s = 30$ bar and $x_s = [\, 0.5 \quad 0.5 \,]^T$ which gives $b_s = [\, 0.5 \quad 0.5 \quad 0.27 \quad 285 \quad 30 \,]^T$.

The selected equation–variable pairings are listed in Table 2. The equation–variable pairing is not unique and other valid combinations exist. An obvious requirement is that, if an equation is paired with a variable, this variable must exist in the actual equation. In the pipe model this leaves two choices for pairing of the outlet stream F_2, component balance one (propane) or component balance two (butane). In this case the recommendation is to pair F_2 with the component balance of the component with the largest molar fraction. This will in fact simplify the variable and equation scaling and remove the need for "extreme" scaling factors.

The first-order derivatives of the pipe unit model, $\partial f_1(z)/\partial z$ is written as a $n_{f_1} \times n_z$ matrix where n_{f_1} is the number of equations in unit model number 1 and n_z is the total number of variables in the process model.

$$\frac{\partial f_1(z)}{\partial z} = \begin{bmatrix} F_1 & 0 & x_1(1) & 0 & 0 & -F_2 & 0 & -x_2(1) & 0 & 0 \\ 0 & F_1 & x_1(2) & 0 & 0 & 0 & -F_2 & -x_2(2) & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ F_1\dfrac{\partial h(x_1,T_1)}{x_1(1)} & F_1\dfrac{\partial h(x_1,T_1)}{x_1(2)} & h(x_1,T_1) & F_1\dfrac{\partial h(x_1,T_1)}{T_1} & 0 & -F_2\dfrac{\partial h(x_2,T_2)}{x_2(1)} & -F_2\dfrac{\partial h(x_2,T_2)}{x_2(2)} & -h(x_2,T_2) & -F_2\dfrac{\partial h(x_2,T_2)}{T_0} & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & -1 \end{bmatrix} \quad (27)$$

A simple verification of the model equations and calculation of first-order derivatives is recommended:

- Compare $\partial f_i(z)/\partial z$ with numerically calculated derivatives.
- Verify that equations are linearly independent. The rank of the first-order derivative $\partial f_i(z)/\partial z$ must equal the number of equations n_{f_i}.
- Specifications added in A (ref. Eqs. (1)–(4)) must be linearly independent of any unit model equation, i.e. the matrix $[\,\partial f_i(z)^{\mathrm{T}}/\partial z \quad A^{\mathrm{T}}\,]^{\mathrm{T}}$ must have full rank.

The matrix of the first-order derivatives of the specifications and pipe model, where $\partial f(z)/\partial z = \partial f_1(z)/\partial z$, are shown in Eq. (28):

$$H = \begin{bmatrix} \dfrac{\partial f(z)}{\partial z} \\ A_s \end{bmatrix} = \begin{bmatrix} 0.27 & 0 & 0.50 & 0 & 0 & -0.27 & 0 & \mathbf{-0.50} & 0 & 0 \\ 0 & 0.27 & 0.50 & 0 & 0 & 0 & \mathbf{-0.27} & -0.50 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \mathbf{1} & 1 & 0 & 0 & 0 \\ 9203 & 10190 & 35913 & 34 & 0 & -9203 & -10190 & -35913 & \mathbf{-34} & 0 \\ 0 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 & 0 & -1 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{1} \end{bmatrix} \quad (28)$$

Table 2
Equation–variable assignment for the pipe unit model

Description	Equation	Pairing
Unit model MB Eq. (1)	$x_1(1)F_1 - x_2(1)F_2 = 0$	F_2
Unit model MB Eq. (2)	$x_1(2)F_1 - x_2(2)F_2 = 0$	$x_2(2)$
Sum of compositions	$\sum x_2 - 1 = 0$	$x_2(1)$
Energy balance	$F_1 h(T_1, \boldsymbol{x}_1) - F_2 h(T_2, \boldsymbol{x}_2) = 0$	T_2
Pressure-flow relation	$P_1 - P_2 = 0$	P_1
Specification no. 1	$A(1)z = b_1$	$x_1(1)$
Specification no. 2	$A(2)z = b_2$	$x_2(2)$
Specification no. 3	$A(3)z = b_3$	F_1
Specification no. 4	$A(4)z = b_4$	T_1
Specification no. 5	$A(5)z = b_5$	P_2

The condition number of H is 8.13×10^7. The main cause of the high condition number is the energy balance equation. This equation has large values compared to the other equation and changes in the paired variable, T_2, has the least significant influence on the equation residual.

In order to reduce the condition number, the model is scaled according to the proposed method (Section 3.4.4).

The matrix of equation and variable pairing, P, is derived from Table 2 and the pairing is shown in Eq. (28) using bold font.

The values of flow, temperature and pressure variables are approximately 0.25 kmol/s, 280 K and 30 bar and the variable scaling matrix $S_v = \mathrm{diag}([\,1 \quad 1 \quad 0.25 \quad 280 \quad 30 \quad 1 \quad 1 \quad 0.25 \quad 280 \quad 30\,])$.

The equation scaling matrices S_n and S_l are computed according to step 3 in the proposed scaling procedure. This gives $S_n = \mathrm{diag}([\,10.0 \quad 2.0 \quad 1.0 \quad 0.00056 \quad 0.033\,])$ and $S_l = \mathrm{diag}([\,1.0 \quad 1.0 \quad 4.0 \quad 0.0036 \quad 0.033\,])$.

The matrix of specifications and first-order derivatives are written as:

$$\tilde{H} = \begin{bmatrix} \widetilde{\dfrac{\partial f(z)}{\partial z}} \\ \tilde{A}_s \end{bmatrix} = \begin{bmatrix} S_n & 0 \\ 0 & S_l \end{bmatrix} \begin{bmatrix} \dfrac{\partial f(z)}{\partial z} \\ A_s \end{bmatrix} S_v = \begin{bmatrix} 2.0 & 0 & 1.75 & 0 & 0 & -5.0 & 0 & \mathbf{-1.0} & 0 & 0 \\ 0 & 0.4 & 0.15 & 0 & 0 & 0 & \mathbf{-1.0} & -0.3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \mathbf{1.0} & 1.0 & 0 & 0 & 0 \\ 0.37 & 0.41 & 0.48 & 0.39 & 0 & -0.91 & -1.0 & -0.48 & \mathbf{-1.0} & 0 \\ 0 & 0 & 0 & 0 & \mathbf{1.0} & 0 & 0 & 0 & 0 & -1.0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \mathbf{1.0} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \mathbf{1.0} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \mathbf{1.0} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \mathbf{1.0} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{1.0} \end{bmatrix} \quad (29)$$

As a result of the applied scaling the condition number of H is reduced from 8.1×10^7 to 6.8.

The pipe model is solved using Matlabs `fsolve` and `fmincon`. `fsolve` is based on a nonlinear least squares algorithm and `fmincon` is a SQP algorithm with BFGS Hessian update (Matlab, 2000). The initial values $z_0 = [0.7 \quad 0.3 \quad 0.2 \quad 278 \quad 20 \quad 0.4 \quad 0.6 \quad 0.5 \quad 270 \quad 25]^T$ is used as a starting point for both solvers.

The unscaled model was solved using 14 iterations using `fsolve` and the scaled model was solved using 4 iterations. The scaled and unsealed model where both solved in three iterations using `fmincon` and scaling does not seem to have any significant effect in this case. Still, when a unit model becomes part of a larger model the condition number will increase further and the effect of scaling will be significant. To compare, the three scaling methods presented in Section 3.4 where also applied to the pipe model. The results are summarized in Table 3. The condition number for the unsealed model was 8.1×10^7. The smallest condition number for the scaled model, 6.8, was obtained using scaling method 4. Note that `fsolve` did not converge to a valid solution using scaling method 2. The solver terminated (successfully) in 3 iterations at a solution where there was a 2.5 K difference in inlet and outlet temperature. The scaled variables had values in the order of 1×10^4 which may have caused the failure of convergence.

Table 3
Comparison of scaling procedures

Scaling	Iterations		Condition number of $\tilde{H}$
	`fsolve`	`fmincon`	
Unscaled	14	3	8.1×10^7
Method 1	5	3	4.3×10^3
Method 2	*	3	3.7×10^6
Method 3	5	3	30
Method 4	4	3	6.8

* Failed to converge.

5. Case study 2: flash process with preheating

A simple flash process is here studied in order to demonstrate the use of the above modeling guidelines in simulation, data reconciliation and optimization. The process, shown in Fig. 4, has three unit models, a heat exchanger, a heater and a flash drum. The three unit models are connected using six process streams.

The model has three chemical components, propane, butane and pentane (NC = 3).

The process operating constraints are S_1 (flow) < 0.3 kmol/s, S_4 (pressure) > 28 bar and <40 bar, Hi (heat duty) < 5500 kW, S_3 (temperature) < 485 K and S_6 (propane content) < 0.2 mol/mol. The feed and energy price are respectively 100$/kmol and 0.001$/kW and the product price are 50$/kmol for vapor product and 200$/kmol for liquid product.

The variables in the model include 6(NC + 3) = 36 process stream variables (6(NC + 3)), two internal variables in the heat exchanger (duty and heat transfer coefficient) and one internal variable in the heater (duty). This gives a total number of $n_z = 39$ variables.

The heat exchanger unit model has 2NC + 7 = 13 equations, the heater has NC + 3 = 6 equations and the flash drum has 2NC + 6 = 12 equations. This gives the total number of $n_f = 31$ equations. The number of degrees of freedom is then $n_z - n_f = 8$.

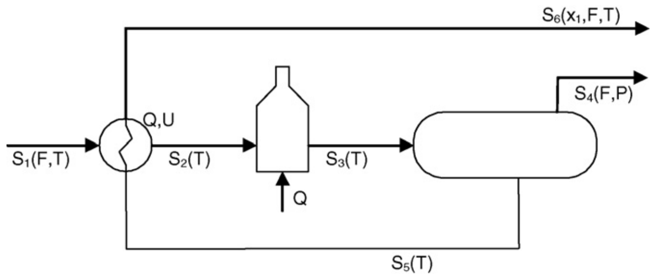

Fig. 4. Flash process.

Table 4
Results for flash process

Variables	y	σ	p	$z_{\min}$	$z_{\max}$	Initial, z_0	Simulation, z_s	Reconciliation, z_r	Optimization, z_{opt}
F_1	0.20	0.025	100		0.30	0.20	0.25	0.20	0.28
T_1	289.92	0.250				280.00	280.00	289.97	289.97
T_2	308.06	0.250				320.00	301.77	308.02	302.78
T_3	474.53	0.250			485.00	370.00	480.00	474.51	438.52
F_4	0.11	0.025	−50			0.10	0.14	0.10	0.09
P_4	31.45	0.500		**28.00**	40.00	30.00	31.50	30.22	28.00
T_6	418.29	0.250				400.00	417.18	418.36	405.81
$x_6(l)$	0.15	0.005			**0.20**	0.20	0.18	0.15	0.20
F_6	0.10	0.025	−200			0.20	0.11	0.10	0.20
T_6	383.90	0.250				380.00	369.75	383.88	387.92
U_{HX}						0.010	0.015	0.010	0.010
Q_{HT}			0.001		**5500**	2000	6264	4856	5500
$J_{opt}(z)$							2.48	−0.206	−9.848

There are 10 measurements: the propane composition, three flows, five temperatures and pressure. The measurements are shown on the figure using the symbols x_1, F, T and P. The measurements are generated by adding normal distributed noise to a simulation result, $y = Uz_y + e_y$ where $e_y = N(0, \sigma)$.

In the simulation problem, as defined by Eq. (1), eight variables have to be specified. In this case feed composition (three variables), feed flow, feed temperature, heater outlet temperature, vapor product pressure and heat transfer coefficient are specified.

In data reconciliation, as defined by Eq. (3), only the feed composition is specified.

The objective of operation is to maximize the profit within constraints. In process optimization, as defined by Eq. (4), feed composition, feed temperature and the heat transfer coefficient are specified. The specification values are set equal to the reconciled values. Table 4 shows the optimization results. Only variables related to measurements and constraints are shown.

The optimal solution has three active constraints (shown as bold values in Table 4): minimum pressure (28.00), maximum liquid product propane content (0.20) and maximum heater duty (5500). The operational profit was increased from a starting point of 0.2$/s (reconciled) to a optimal of 9.8$/s.

To compare the results of solving the scaled (method 4) and unscaled data reconciliation problem, the relative estimation error, $d_{est} = \sum_{j=1}^{n_y} |(Uz_r - y)_j / y_j|$, is used. Ten different sets of normal distributed measurement error were generated and for each set the unscaled and scaled data reconciliation and optimization problem was solved.

In 8 of 10 runs the unscaled data reconciliation problem converges to a local optimum where $d_{est} \approx 3$. In these runs the scaled problem converges with an average of 10 iterations and an average estimation error $d_{est} \approx 0.3$. In the other two runs, the scaled and unscaled problem converges to the same solution. In this case the unscaled problem converged using 45 and 50 iterations.

The optimization problem was solved using the reconciled values as described above. In 10 of 10 runs the scaled optimization problem converged to the optimal solution with pressure, product composition and heater duty as active constraints. In 10 of 10 runs, the unscaled optimization problem failed to converge to the optimal solution and converged to a solution where only the product composition constraint was active. In this case the average objective was $J_{opt} \approx -9.3$.

To compare the four scaling methods presented in Section 3.4 the simulation, data reconciliation and optimization problem is solved for the flash process. The results are summarized in Table 5.

A valid solution of the data reconciliation and optimization problem are found using scaling methods 1, 3 and 4. A common property of these methods is a reduction of the condition number of the constraint first-order derivatives, $\tilde{H}$. The smallest condition number (5.1) is achieved using method 4 which also solves these problems using the fewest number of iterations. The condition number of a model increases with the model size and large models will benefit more from the use of scaling methods that gives a large reduction in condition number.

Table 5
Scaling methods applied to flash process

	Condition no. $\tilde{H}$	Number of iterations					Active constraints
		Simulation	Reconciliation	Optimization	d_{est}	J_{opt}	
Unscaled	5.0E+09	5	23	11	2.99	−9.34	$x_6(1)$
Method 1	4.8E+05	4	14	7	0.30	−9.85	$P_4, x_6(1), Q_{HT}$
Method 2	7.0E+09	4	9	3	3.49	−5.84	Q_{HT}
Method 3	4.0E+03	4	28	10	0.30	−9.85	$P_4, x_6(1), Q_{HT}$
Method 4	5.1E+01	4	12	5	0.30	−9.85	$P_4, x_6(1), Q_{HT}$

6. Discussion

Multiple steady states are not handled by this method. Rather, the scaling is performed at the desired steady-state.

Nonlinear inequality constraints can be added in this framework by introduction of slack variables according to Luenberger (1984). As a simple example we have that $g(x) < c$ is equivalent to $g(x) - v = 0$, $v < c$ where v is a slack variable.

The simplified thermodynamic relations used in the case studies are all explicit functions and have explicit functions for their first-order derivatives. For example, specific enthalpy is calculated as $h = C_p x^{\mathrm{T}}$. The specific heat of a component, C_{pi}, is a fixed value for the liquid phase and a function of temperature for the vapor phase (sixth order polynomial fitted to data from NIST (2005)). Vapor–liquid equilibrium is based on Raoult's law and Antoine vapor pressure with parameters from the same source.

The described unit model structure is well suited for object oriented programming. A model written in C++ or similar programming language, most commonly used in applications, will be far more effective than the Matlab code used in the examples.

The use of sparse matrices and sparse math in the model and solver code will also give a significant reduction in computational load. In the flash process case study the matrix H has 1521 elements of which only 169 are non-zero.

7. Conclusions

A procedure for building steady state models has been presented. The procedure is based on unit models which interact through a shared variable vector. The unit models and specifications form an "open equation" set, well suited as nonlinear constrains in an optimization problem. In the suggested structure each unit model can be developed, tested and scaled before it is added to the overall process model. This simplifies the modeling work and saves a lot of troubleshooting.

The scaling procedure, which is applied at unit model level, results in a significant improvement in the overall numerical properties of the model.

The numerical examples of the flash process optimization shows that proper scaling reduces the number of iterations used for solving each case. More important though, it makes the results more reliable. In both data reconciliation and optimization, the solver failed in finding the optimal solution when using an unsealed model.

Acknowledgements

The first author acknowledges Statoil Mongstad for the financial support. The authors are grateful to Professor Terje Hertzberg at the Norwegian University of Science and Technology (NTNU) in Trondheim, for his interest in this work, encouragement and helpful comments. The authors are also grateful to Dr. Stig Strand at the Statoil Research Center in Trondheim for many stimulating discussions and helpful comments.

References

Biegler, L. T., & Cuthrell, J. E. (1985). Improved infeasible path optimization for sequential modular simulators. II. The optimization algorithm. *Computers & Chemical Engineering*, *9*(3), 257–267.

Bogusch, R., & Marquardt, W. (1995). A formal representation of process model equations. *Computers & Chemical Engineering*, *19*, S211–S216.

Chen, X., Pike, R. W., Hertwig, T. A., & Hopper, J. R. (1998). Optimal implementation of on-line optimization. In *European symposium on computer aided process engineering* (pp. 435–442).

Kelly, J. D. (2004). Techniques for solving industrial nonlinear data reconciliation problems. *Computers & Chemical Engineering*, *28*, 2837–2843.

Lid, T., Skogestad, S., & Strand, S. (2002). On line optimization of a crude unit heat exchanger network. *Chemical Process Control-6, AIChE Symposium Series*, 326.

Luenberger, D. G. (1984). *Linear and nonlinear programming* (2nd ed.). Reading, MA: Addison-Wesley.

Mah, R. S. H. (1990). *Chemical process structures and information flows. Butterworths series in chemical engineering*. Boston: Butterworths. ISBN 0-409-90175-x.

Marquardt, W. (1996). Trends in computer-aided modelling. *Computers & Chemical Engineering*, *20*, 591–609.

Matlab. *Optimization Toolbox Version 2.1*. The MathWorks Inc., 3 Apple Hill Drive, Natick, MA 01760-2098, United States, 2000.

Maurya, M. R., Rengaswamy, R., & Venkatasubra-manian, V. (2003). A systematic framework for the development and analysis of signed digraphs for chemical processes. 1. Algorithms and analysis. *Industrial & Engineering Chemistry Research*, *42*(20), 4789–4810.

NIST. *NIST Chemistry WebBook*. National Institute of Standards and Technology, http://webbook.nist.gov/chemistry/, 2005

Nocedal, J, & Wright, S. J. (1999). *Numerical optimisation Springer series in operations research*. New York: Springer.

Pantelides, C. C. (1988). The consistent initialisation of differential-algebraic systems. *SIAM Journal on Scientific and Statistical Computing*, *9*(2), 213–231.

Rodriguez-Toral, M. A., Morton, W., & Mitchell, D. R. (2001). The use of new SQP methods for the optimization of utility systems. *Computers & Chemical Engineering*, *25*, 287–300.

Roma, M. (2005). Dynamic scaling based preconditioning for truncated newton methods in large scale unconstrained optimization. *Optimization Methods and Software*, *20*(6), 693–713.

Stanley, G. M., & Mah, R. S. H. (1981). Observability and redundancy in process data estimation. *Chemical Engineering Science*, *36*, 259–272.

Tjoa, I. B., & Biegler, L. T. (1991). Simultaneous strategies for data reconciliation and gross error detection of nonlinear systems. *Computers & Chemical Engineering*, *15*(10), 679–690.

Westerberg, A. W., Hutcison, H. P., Motard, R. L., & Winter, P. (1979). *Process flowsheeting*. London: Cambridge university press.

White, D. C. (1997). Online optimization: What, where and estimating ROI. *Hydrocarbon Processing*, 43–51.

Zhu, D. (2005). An affine scaling projective reduced Hessian algorithm for minimum optimization with nonlinear equality and linear inequality constraints. *Applied Mathematics and Computation*, *166*(1), 131–163.