Noname manuscript No.
(will be inserted by the editor)

Sampling solutions of sparse nonlinear systems

Ali Baharev · **Ferenc Domes** · **Arnold Neumaier**

Received: date / Accepted: date

Abstract The algorithm proposed in the present paper aims at robustly sampling from a manifold implicitly defined by an underdetermined system of equations and bound constraints on the n variables. A representative sample of the solution manifold of the input system can have several uses. For example, the sample can be used to generate starting points for solving arbitrary systems obtained by augmenting the original underdetermined system by further equations such that the system becomes well-defined. The sample is also useful for the robust computation of bifurcation diagrams.

Direct sampling of the manifold means sampling in a very thin subset of $\mathbb{R}^n$, with a typical work that grows exponentially with n. Therefore, we first decompose the input problem into a well-chosen sequence of lower-dimensional subsystems. The proposed algorithm then only samples the solution set of these subsystems in order. This leads to substantial computational savings because the work grows exponentially only with the largest subproblem size, which is usually significantly smaller than n.

The robustness of the method is demonstrated on numerically challenging steady-state models of distillation columns where even problem-specific methods were reported to miss one solution. When started from the sample points generated by the proposed method, a general-purpose gradient-based solver finds all solutions to these steady-state models without problem-specific assumptions. There is no theoretical guarantee that all solutions will be found in the general case, although, increasing the sample size is expected to improve robustness.

Keywords large-scale systems of equations · decomposition · tearing · sequential modular · chemical engineering · multiple steady states

The research was funded by the Austrian Science Fund (FWF): P23554. Support by the Austrian Research Promotion Agency (FFG) under project number 846920 is gratefully acknowledged.

Ali Baharev, Ferenc Domes, Arnold Neumaier
Faculty of Mathematics, University of Vienna, Oskar-Morgenstern-Platz 1, 1090 Vienna, Austria
Tel.: +43 681 201 78 626
Fax: +43 1 4277 850601
E-mail: ali.baharev@gmail.com

1 Introduction

1.1 Input of the proposed method

We consider a manifold (possibly with singularities) implicitly defined as the solution set of the bound-constrained nonlinear underdetermined system

$$F(x) = 0, \quad \underline{x} \leq x \leq \overline{x}, \tag{1}$$

where $F : \mathbb{R}^n \mapsto \mathbb{R}^{n-d}$ is a continuously differentiable vector-valued function; $\underline{x}$ and $\overline{x}$ denote the vector of lower and upper bounds on the variables x, respectively. It is assumed that these bounds are reasonable and not big M bounds. From an engineering perspective, this requirement does not spoil the generality of the proposed algorithm: The variables in a typical technical system are double-bounded, although often implicitly. The physical limitations of the devices and the design typically impose minimal and maximal throughput / load of the devices; this implies bounds on the corresponding variables, either explicitly or implicitly. There are also natural physical bounds, e.g., mole fractions of chemical components must be between 0 and 1, etc.

The proposed method assumes that the Jacobian has block lower Hessenberg form, defined by Equations (2)–(3) and shown in Figure 1. The variables are par-

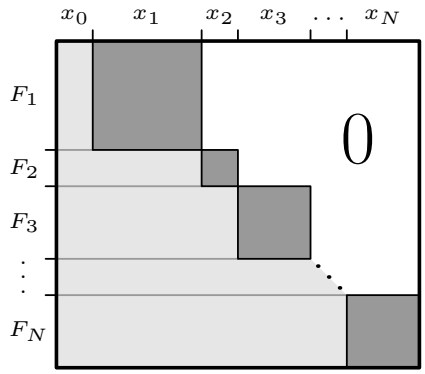

Fig. 1 The desirable block sparsity pattern of the Jacobian, the block lower Hessenberg form.

titioned as

$$x = \begin{pmatrix} x_0 \\ \vdots \\ x_N \end{pmatrix} \tag{2}$$

into $N+1$ subvectors $x_i \in \mathbb{R}^{n_i} \quad (i = 0 \ldots N)$, so that $n = n_0 + \cdots + n_N$. Similarly, F is partitioned as

$$F(x) = \begin{pmatrix} F_1(x) \\ \vdots \\ F_N(x) \end{pmatrix} \tag{3}$$

into N vector-valued subfunctions $F_i(x) \in \mathbb{R}^{m_i} \quad (i = 1 \ldots N)$, and $n - d = m_1 + \cdots + m_N$. The block lower Hessenberg structure guarantees that only variables from the subvectors $x_0, \ldots, x_i$ $(i \leq N)$ can appear in $F_i(x)$:

$$F_i(x) = F_i(x_0, \ldots, x_i) \quad \text{for } i = 1, \ldots, N. \tag{4}$$

In practice, the lower triangle is often sparse.

Any sparse matrix can be automatically ordered to block lower Hessenberg form, c.f. (Hellerman and Rarick 1971; Stadtherr and Wood 1984a,b; Erisman et al 1985; Fletcher and Hall 1993), `MC33` of (HSL 2015), and (Duff et al 1986, pp. 168–175). Discussing sparse matrix ordering algorithms are outside the scope of this paper. It was not necessary to apply an ordering algorithm to the distillation columns considered in Section 4: The block lower Hessenberg form was immediately obtained by partitioning the Jacobian along the devices (called stages) forming the column.

1.2 An informal overview of the proposed method

The proposed method is sketched in the followings. The algorithm essentially builds up a set of solution vectors to Equation (1) step-by-step, processing the blocks of the lower Hessenberg form along the upper envelope; the output of the method is a set of scattered points that can be considered a representative sample of the solution manifold. A sample is considered *representative* if no two points are closer than a user-defined small threshold, and no point on the solution manifold is too far from the sample. The former is trivial to enforce; the latter is accomplished only on a best effort basis by inserting new points into the sample after processing a block, and keeping only the most distant points. This procedure, striving to make the sample representative by improving the point distribution in each iteration, will be referred to as redistribution, and it is the distinguishing feature of the method. The primary (and currently user-defined) parameter of the method is the number of points to be generated in the sample: Increasing the sample size is expected to improve robustness but at the cost of increased computational costs.

Robust sampling is used to generate the new points in the redistribution step. Sampling implicitly defined manifolds in $\mathbb{R}^n$ can be prohibitively expensive if n is large, since the volume to be sampled grows exponentially with n. The proposed method circumvent sampling in $\mathbb{R}^n$ by sampling only the solution manifold of the blocks along the upper envelope, from $i = 1$ to N. This ordering of the blocks ensures that for $i > 1$ the subvector $\{x_0, \ldots, x_{i-1}\}$ in $F_i(x_0, \ldots, x_{i-1}, x_i) = 0$ is available from the previous iteration. Therefore, the volume to be sampled grows exponentially only with the largest block size, which is usually significantly smaller than n.

1.3 Motivation and applications

If the underdetermined system (1) is augmented by further equations such that the system becomes well-defined, the representative sample of the solution manifold of (1) can can be used to generate starting points for solving this well-defined system. This is of great significance for engineering applications.

The problem of solving nonlinear systems of equations arises in the daily engineering practice, e.g., when consistent initial values for differential algebraic equation (DAE) systems are sought (Pantelides 1988; Unger et al 1995), or when solving steady-state models of technical systems. A steady-state solution can be used as a consistent initial set of the DAE system (Kröner et al 1997).

Even though mature equation-based object-oriented modeling environments are available, e.g., Modelica (Mattsson et al 1998; Tiller 2001; Fritzson 2004) for multi-domain modeling of heterogeneous complex technical systems, and gPROMS (gPROMS 2015), ASCEND (Piela et al 1991) and EMSO (de P. Soares and Secchi 2003) for chemical process modeling, simulation and optimization, etc., the steady-state initialization is still not satisfactorily resolved in the general case. Often, steady-state initialization failures can only be resolved in very cumbersome ways (Vieira and Jr 2001; Bachmann et al 2007; Sielemann and Schmitz 2011; Sielemann et al 2013; Ochel and Bachmann 2013), involving user-provided good initial values for the variables.

Finding all solutions to a well-defined system of equations (or proving that the system is infeasible) is a global optimization problem, where the objective function is absent. Global optimization is an NP-hard problem in the general case (Kreinovich and Kearfott 2005). In view of the $P \neq NP$ conjecture (see (Sipser 1997) or (Baker et al 1975) for the definition of the conjecture, and (Papadimitrou 1977) and (Pardalos and Schnitger 1988) for the connection to optimization), it is therefore very likely that for every generic algorithm there will be problem classes for which the algorithm needs an effort exponential in the problem size. In practice, either problem-specific solutions are used, or the guarantees regarding global convergence are relaxed and one resorts to heuristics. This paper presents a fairly generic method that can be used for generating starting points for gradient-based solvers.

1.4 Structure of the present paper

An alternative to robust sampling would be to properly parameterize the solution manifold of Equation (1). The published methods for the parameterization of implicitly defined manifolds are reviewed in Section 2.1. This paper focuses on computing multiple steady-states as an application of the manifold-based starting point generation with the proposed method. The common general-purpose methods for finding multiple solutions are discussed in Section 2.2, and a particular starting point generation heuristic, successful in chemical process flowsheeting, in Section 2.3. The proposed method for underdetermined systems is presented in Section 3; it is extended to square systems in Section 3.3. If the distinguishing feature of the proposed method (i.e. redistribution) is disabled, then the method resembles the heuristic of Section 2.3.

Numerically challenging distillation columns are computed with the new algorithm in Section 4 to demonstrate the robustness of the proposed method.

2 Literature review

2.1 Generic manifold parameterization methods

In the absence of degeneracy, the solution set of (1) is a d-dimensional subset of $\mathbb{R}^n$ which can be locally parameterized by d parameters. Ignoring the 1, 2, and 3 dimensional cases, which are well-studied for obvious reasons, it appears that four general-purpose continuation methods for parameterizing implicitly defined manifolds have been published: a piecewise-linear continuation (Allgower and Schmidt 1985), a moving-frame algorithm (Rheinboldt 1988), a simplicial continuation (Brodzik and Rheinboldt 1994; Brodzik 1998), and a covering algorithm using balls (Henderson 2002). All of these algorithms are higher-dimensional continuation methods. Each uses different representation of either the solution manifold or the neighborhood of a boundary point. Despite the differences, on a higher level of abstraction, they all look the same (Henderson 2007): An approximately spherical neighborhood of a point (or face) on the boundary is constructed, and then merged into the already resolved part of the solution manifold. As a result, they may miss complete branches if these are disconnected from the starting point. To the best of our knowledge, these type of higher dimensional continuation methods have never gained popularity in chemical process simulation.

Popular methods for building metamodels (surrogate models) of computer simulation include kriging (Davis and Ierapetritou 2007; Caballero and Grossmann 2008; Boukouvala and Ierapetritou 2013), and artificial neural networks (Morris et al 1994; Henao and Maravelias 2011). Another, recent development in this area is the software tool ALAMO (Cozad et al 2014): It allows for a large set of potential basis functions, and by combining optimization techniques and statistical methods, the best subset of these basis functions is found that accurately model the data without overfitting or using unnecessary terms.

2.2 General-purpose methods for finding multiple solutions

Multistart methods try to find all solutions by starting a gradient-based local solver from multiple starting points. Guarantees regarding finding all solutions depend on the placement of the starting points. Perhaps the simplest method for bound constrained problems is the grid search: The constraint violation is evaluated at each point of a fine grid, and the best points are used as starting points for local optimization. Finding all solutions with probability one can be achieved by making the grid sufficiently dense. This naive approach is only effective in very low dimensions as the number of grid points grows exponentially with the dimension of the problem.

Multistart methods are applicable to large-scale systems of equations by giving up on the strong guarantees that, for example, grid search would provide. In practice,

sophisticated approaches are used to place the starting points, for example constraint consensus (to reduce infeasibility in a computationally cheap way) and clustering (to separate basins of attraction) by Smith et al. (Smith 2011; Smith et al 2013b,a) or stochastic methods (especially population-based metaheuristics) (Ugray et al 2007). These methods strike a balance between speed and robustness.

Another set of methods that are often successfully used to solve large scale problems - especially if the objective function can be cheaply computed - are *evolutionary algorithms*, e.g. CMA-ES (Auger and Hansen 2005), if they are combined with local optimization starting from the most promising points found. Other methods that are based on similarities to natural processes (ant colony (Dorigo et al 2006), particle swarm (Eberhart and Kennedy 2002), etc.) can be used in a similar way.

Probability-one homotopy methods, especially with problem-specific homotopy maps, are successfully applicable to large-scale industrial problems (Vadapalli and Seader 2001; Doherty et al 2008; Malinen and Tanskanen 2010; Jiménez-Islas et al 2013). This approach is suitable for component-based (also referred to as object-oriented or modular) modeling of large, complex, and heterogeneous technical systems (Sielemann et al 2013): Once the models of the devices are implemented and put into a library, practically no understanding of the homotopy methods is required from the end-user. The critical part is the implementation of the model of the devices. Unless special care is taken, the convergence with probability one is usually not guaranteed (Seydel 2010). Although significant effort and research have been made in this field, robustness still remains an issue in the general case (Sielemann and Schmitz 2011; Sielemann et al 2013). The construction of problem-specific homotopy maps requires expert understanding of the homotopy methods and also considerable engineering skills in the application domain.

Branch-and-bound methods recursively split the search space into smaller parts and eliminate those parts that cannot lead to a solution better than the currently best known one. Unfortunately, their worst-case performance tends to grow exponentially with the dimension of the problem since they perform exhaustive search. (For non-convex functions, global optimization is NP-hard.) The applicability of branch-and-bound methods currently seems to be limited to fairly low-dimensional problems in the general case. Successful enclosure methods in chemical engineering include interval arithmetic (Gwaltney et al 2008), McCormick relaxations (Mitsos et al 2009; Scott et al 2011), affine arithmetic (Baharev et al 2011; Soares 2013), and αBB (Guzman et al 2014).

2.3 A starting point generation heuristic from chemical process flowsheeting

There is a traditional heuristic, summarized in (Biegler et al 1997, Ch. 8), that is based on partitioning, precedence ordering, and tearing, and is used in professional chemical process simulators when run in sequential-modular mode. The complete or partial sequential-modular solution can be used for initializing equation-oriented models (Aspen Technology, Inc. 2009).

With the notation of this paper, the heuristic proceeds *roughly* as follows. The blocks are determined by the modules of the chemical process flowsheet, and are

ordered into bordered block lower triangular form with an appropriate, usually greedy heuristic; see Figure 2. The objective of the ordering heuristic is to minimize $\dim x_0$,

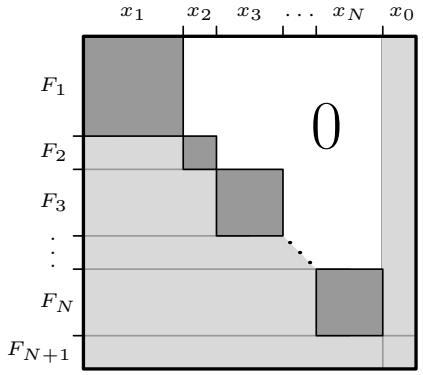

Fig. 2 Bordered block lower triangular form.

since it influences the computational work, and the user has to provide an initial guess for these but only for these variables. Then, the diagonal blocks are eliminated from $i = 1$ to N, and $F_{N+1}(x)$ is considered as a function of x_0 only: $G(x_0)$, where $G : \mathbb{R}^d \mapsto \mathbb{R}^d$; finally, $G(x_0) = 0$ is solved for x_0, the only variables not eliminated. The stage-by-stage method (Lewis and Matheson 1932; Thiele and Geddes 1933) for distillation columns also fits this pattern: The blocks correspond to the stages of the distillation column, and no ordering heuristic is needed because the natural order of the stages already gives a bordered block lower triangular form.

The approach has its merits: For example, it is relatively easy to implement in a module-oriented simulator, and it is usually robust if $G(x_0)$ is well-conditioned. Unfortunately, the method can show extreme sensitivity to the initial guess for x_0, since numerical sensitivity can build up while the blocks are eliminated along the diagonal. In such cases, even if the Jacobian $F'(x)$ of the original system is well-conditioned, the Jacobian $G'(x_0)$ of the reduced system is extremely ill-conditioned. This has a negative impact on the convergence properties of the methods used for solving $G(x_0) = 0$. The sensitivity issue can become so severe that, with all the intermediate variables $x_1, \ldots x_N$ eliminated, there may not be any machine representable vector for x_0 such that $G(x_0) = 0$ is satisfied with acceptable numerical accuracy. This can happen even for well-conditioned problems; the distillation column computed in Section 4.1 is unsolvable with the stage-by-stage approach.

3 Proposed algorithm

The proposed algorithm was sketched in Section 1.2; further details are given here, before the pseudo-code of the algorithm is presented. The algorithm essentially builds up a set of approximate solution vectors step-by-step, processing the blocks of the

lower Hessenberg form along the upper envelope. The algorithm starts with a well-distributed set of points $S^{(0)}$ for $\{x_0\}$; Latin hypercube sampling is used to generate $S^{(0)}$ in the current implementation. In iteration i, in order $1,2,\ldots,N$, the subsystem

$$F_i(x_0,x_1,\ldots,x_{i-1},x_i) = 0 \tag{5}$$

is solved for x_i for each subvector $\{x_0,\ldots,x_{i-1}\}$ in the set of propagated points $S^{(i-1)}$. The block lower Hessenberg form guarantees that only x_i is unknown in iteration i; the values of $\{x_0,\ldots,x_{i-1}\}$ have already been computed in the previous iterations, that is, they are in $S^{(i-1)}$.

The algorithm described so far would have the same numerical sensitivity issues that tearing and the stage-by-stage approach have, see Section 2.3. The only difference up to this point is that the proposed algorithm propagates not just a single point but a set of points; Equation (5) is solved for a set of points in each iteration. The proposed algorithm avoids the severe ill-conditioning characteristic of tearing and the stage-by-stage approach by improving the point distribution after each block, inspired by our earlier results for the univariate case (Baharev and Neumaier 2014). Therefore, the distinguishing feature of the proposed algorithm is the redistribution step, performed in each iteration.

The redistribution strives to ensure that the sample of the solution manifold of Equation (5) is *representative:* No two points are closer than a user-defined small threshold, and no point on the solution manifold is too far from the sample. The former requirement is trivial to enforce, the latter is accomplished only on a best effort basis as follows. Additional points are inserted into the point set in each iteration by robust sampling and sensitivity analysis; then, points are discarded until the most distant M points remain, where M is a user-defined parameter.

In particular, new values for a selected subset of x_i are picked with robust sampling and are kept fixed (currently QR factorization with pivoting for subset selection, and Latin hypercube sampling). The missing part of the partial solution is completed with the following heuristic. With a given user-defined small integer h, the best approximating subvector $\{x_0,x_1,\ldots,x_{i-h-1}\}$ is chosen from the point set $S^{(i-1)}$, and $\{x_{i-h},\ldots,x_i\}$ is computed by re-solving the last h blocks $F_{i-h}(x) = 0,\ldots,F_i(x) = 0$, with the selected subset of x_i kept fixed according to the robust sampling. Finally, only the most distant M points are kept.

3.1 Pseudo-code of the proposed algorithm

The algorithm is presented in high-level pseudo-code in Algorithm 1; the reader is referred to Section 3.4 for practical matters, such as the choice and effect of the used-provided parameters. The Java source code is available as online supplementary material (Baharev et al 2015). In the pseudo-code that follows, the shorthand $p{:}q$ is used for the index set $p, p+1,\ldots,q$, similarly to the Matlab notation.

The actual norm on line 4 depends on the solver being used to solve the optimization problem. In our implementation, the user can choose between IPOPT (Wächter and Biegler 2006) and LMBOPT (Neumaier and Azmi 2015); the former uses the L^1

Algorithm 1: The proposed algorithm

Input: A problem instance in the form of Equations (1)–(3)
Parameters : M: at most M_i points are kept in iteration i
ε: tolerated constraint violation
Output: A representative sample of the solution manifold of Eq. (1)
1 Populate the set $S^{(0)}$: Pick M_0 vectors for x_0 with (e.g. Latin hypercube) sampling
2 **for** $i = 1$ **to** N **do**
3 **foreach** $x_{0:i-1} \in S^{(i-1)}$ **do**
4 Solve $\min_{x_i} \|F_i(x_{0:i-1}, x_i)\|$ s.t. $\underline{x}_i \leq x_i \leq \bar{x}_i$
5 Add $\{x_{0:i-1}, x_i\}$ to $S^{(i)}$
6 **end**
7 Call **Redistribution**(i, $S^{(i-1)}$, $S^{(i)}$), then add the result to $S^{(i)}$
8 Remove all $x_{0:i}$ from $S^{(i)}$ for which $\|F_{1:i}(x_{0:i})\|_\infty \geq \varepsilon$
9 Keep only the most distant M_i points from the remaining $S^{(i)}$
10 **end**
11 **return** $S^{(N)}$

norm, the latter the L^2 norm. The choice of the norm had no effect on the algorithm in our numerical experience.

3.2 Redistribution

The pseudo-code for the redistribution is given in Algorithm 2. The goal of this algorithm is to keep the limited number of sample points apart as much as possible. The newly introduced $x_{i-h:i}$ parts of the forced new points are separated from each other on a best effort basis. The new values for x_i are obtained with Latin hypercube sampling on line 1, which is expected to give good separation of these new x_i parts. Local sensitivity analysis is performed to find the index set J of the least influential variables (more on this shortly). Only these least influential variables of x_i are ultimately fixed on line 12 (s.t. $(y_i)_J = (x_i)_J$), so that infeasibility can be most likely repaired on line 12 by minimizing the constraint violation. The fixed $(x_i)_J$ parts are also separated from each sample point $x_i^{(S)}$ (computed in Algorithm 1 on line 4): those $(x_i)_J$ parts that already have nearby neighbors in the sample $S^{(i)}$ are discarded on line 8 of Algorithm 2.

The entire $x_{0:i-1}$ part of the partial solution should not be recomputed on line 12, not even with inter- or extrapolations: That would potentially make the complexity of the entire algorithm $O(n^2)$ where n is the number of variables in Equation (1). This is the reason why $x_{0:i-h-1}$ is left unchanged, and only the last h subproblems are considered on line 12.

The goal of the local sensitivity analysis performed on line 4 is to order the variables x_i according to their influence on $\|F_i(x_{0:i})\|_2$ with $x_{0:i-1}$ fixed. QR factorization with column pivoting is performed on the Jacobian of $F_i(x_{0:i})$; the resulting permutation vector gives the desired ordering (Golub and van Loan 1996). The index set J on line 4 denotes the indices of the least influential variables; the cardinality of J is chosen to be $\dim x_{i-h:i} - \dim F_{i-h:i} + 1$, that is, the least influential variables are

Algorithm 2: Redistribution

Input: index i of the current subproblem; for $k = i-1, i$, the set $S^{(k)}$ of sample points $x_{0:k}$ for which $F_{1:k}(x_{0:k}) \approx 0$
Parameters : p: number of forced new points
δ: threshold for two points being too close
h: maximal number of subproblems to be considered
Output: A set of points U containing new values for $x_{0:i}$

```
// Force new values for a subvector of x_i
// Latin hypercube sampling is used in the current implementation
```

Pick p vectors for x_i with robust sampling, and add them to set R
$T \leftarrow \{\ \}$
foreach $x_{0:i-1} \in S^{(i-1)}, x_i \in R$ **do**

```
    // In the current implementation the index set J is
    // identified by QR factorization of F_i'(x_{0:i}) with pivoting
```

    Apply sensitivity analysis to find the *subvector* $(x_i)_J$ of the least influential variables
    Add $(x_i)_J$ to T
end

```
// Discard those (x_i)_J that already have nearby neighbors in S^(i)
```

foreach $(x_i)_J \in T$, $x_i^{(S)} \in S^{(i)}$ **do**
    Remove $(x_i)_J$ from T if $\|(x_i)_J - (x_i^{(S)})_J\|_\infty < \delta$
end

```
// Calculate the missing history of (x_i)_J
// Brute-force search for the best approximating x_{0:i-h-1} ∈ S^(i-1)
```

$U \leftarrow \{\ \}$
foreach $(x_i)_J \in T$, $x_{0:i-h-1} \in S^{(i-1)}$ **do**

```
    // Resolve the last h subproblems to find the missing part x_{i-h:i}
```

    Solve $\min_{y_{i-h:i}} \|F_{i-h:i}(x_{0:i-h-1}, y_{i-h:i})\|$ s.t. $(y_i)_J = (x_i)_J$, $\underline{x}_{i-h:i} \le y_{i-h,i} \le \overline{x}_{i-h:i}$
    Add $\{x_{0:i-h-1}, y_{i-h:i}\}$ to U
end
return U

fixed until the subsystem $F_{i-h:i}$ becomes overdetermined by one degrees of freedom. Making this subsystem overdetermined by fixing the well-separated least influential variables $(x_i)_J$ proved to be necessary, otherwise the gradient-based solver may place many of the solution vectors near to each other in a particular subspace. (This was observed more often with LMBOPT than with IPOPT.)

3.3 Manifold-based starting point generation for square systems

The proposed method is extended in this section to robustly compute all solutions to square systems of equations. A square system is obtained if d additional equations $F_{N+1}(x) = 0$ augment Equations (1)–(3). (These additional equations must not introduce new variables.) The algorithm presented in Section 3.1 is applied to generate the set of approximate solutions $S^{(N)}$, satisfying the underdetermined part defined by Equations (1)–(3) up to the user-defined threshold: $\|F_{1:N}(x)\|_\infty < \varepsilon$ for each $x \in S^{(N)}$.

The sample points $x \in S^{(N)}$ are then sorted by constraint violation $\|F_{1:N+1}(x)\|_\infty$, in increasing order. In our numerical experience, the algorithm can meet the $\|F_{1:N}(x)\|_\infty < \varepsilon$ requirement with ease for the underdetermined part of the square system. As the

constraints $F_{N+1}(x) = 0$ are unknown to the algorithm while the sample points are computed, it is accidental how far the sample points end up from the solutions to the square system $F_{1:N+1}(x) = 0$. Therefore, $\|F_{N+1}(x)\|_\infty \gg \|F_{1:N}(x)\|_\infty$ typically holds for most of the sample points. Since the number of sample points is limited to M_N, even the closest sample points are usually approximate solutions only, see Figure 5.

There are several options after the sample has been sorted by increasing order of constraint violation. The simplest way is to try all sample points in order; this order makes it very likely to find the solutions early. The search procedure can be terminated early at the user's discretion, if the expected number of solutions have already been found. Another possibility is to try only the first k sample points as starting points (with e.g., $k = 10$, or $k = \lceil M_N/4 \rceil$ for example). It is also possible to try only those sample points whose constraint violation is below a user-defined threshold. Partly because the most difficult test example in Section 4 requires only $M_N = 75$ points, and partly to gain insight, we run the gradient-based solver IPOPT (Wächter and Biegler 2006) from each sample point in the current implementation. In our numerical experience, the gradient-based solver converges to the nearest solution in a few iteration steps when started from the corresponding sample point, see also Figure 5.

If the final $F_{N+1}(x) = 0$ equations of the square system are changed, the set of approximate solutions $S^{(N)}$ to the unchanged Equations (1)–(3) can be reused at no cost. In some sense, the sample $S^{(N)}$ of the solution manifold acts as a surrogate model for the underdetermined part of the equations.

In case of distillation columns, the final equations often correspond to the specifications either at the reboiler or at the condenser. Computing new starting points only requires evaluating the final $F_{N+1}(x) = 0$ equations for each $x \in S^{(N)}$ after an arbitrary – even structural – change in the specifications. This makes the proposed method practical for computing bifurcation diagrams. Unlike continuation methods, this approach does not rely on the connectedness of branches or connectedness to previously computed points.

3.4 Notes on practical matters

The primary parameter of the algorithm is the sample size M_i in iteration i. In the current implementation, $M_i = c \cdot i + M_0$ is used, where c and M_0 are constants; the choice $M_i = i + 25$ proved to be appropriate for the most difficult distillation column considered in this paper, and for all other, easier problems in our test set. In our numerical experience, the probability of finding all solutions vs. the final sample size M_N (with either M_0 or c fixed, and the other varied) is a sigmoid-like function: For small M_N values, the generated sample is not representative, and some of the solutions are usually lost; all solutions are consistently found if M_N is large enough; and there is a transient range in between, where it is accidental whether all solutions are found or not. The transient range is absent if the test problem is easy. Increasing M_i (for $i = 0 : N$) is desired from a robustness point of view; the downside of setting M_i large is performance-wise, due to the large number of sample points that are computed unnecessarily.

In the current implementation, it is left to the user to specify M_0 and c. One can probably find a parametric formula that gives a reasonable default value for M_i as a function of the key characteristics such as the size of the largest block, the size of block i, the number of blocks N, and the dimension of the solution manifold (denoted by d in Equation (1)). The parameters of such a formula with the right qualitative form should be fitted and cross-validated by running the algorithm on a large benchmark set, consisting of diverse test problems. This is the subject of future research.

The second most important parameter of the proposed method is h in the redistribution step: After forcing a new point into the sample, the last h blocks are simultaneously re-solved to minimize the constraint violation resulting from the forceful insertion. If h is chosen too small (for example $h = 1$), we may fail to reduce the constraint violation sufficiently, and the new forced points will be discarded in Algorithm 1 on line 8; this can lead to poor resolution of the solution manifold, and some of the solutions will be lost eventually. Setting h to a large value ($h \gg 5$) spoils the performance, since the last h blocks are resolved simultaneously, and the increased computational effort does not yield any visible improvement in robustness. The $h = 4$ choice works for all the test problems in our test set.

In the current implementation, it is left to the user to specify h. Similarly to M, it is subject of future research to work out a rule for choosing a default value for h. In particular, the following adaptive rule seems appealing. In our numerical experience, the remaining constraint violation in Algorithm 2, line 12 decreases rapidly as h is increased. Therefore, the algorithm could start with a small value for h at each block (for example $h = 2$), and increment it until either the tolerated constraint violation ε of Algorithm 1 or a user-defined cutoff for h (for example $h = 5$) is reached. The optimization problem in Algorithm 2 on line 12 can be quickly re-solved after incrementing h if the gradient-based solver is started from the previous solution. Other, more sophisticated adaptive rules are also possible; elaborating one is subject of future research.

With M and h fixed, the other parameters have negligible effect on the robustness and performance of the method in our experience; therefore, practically no effort was made to tune these other parameters. For the purposes of documentation only, the following settings were used: $\varepsilon = 2/M_0$ in Algorithm 1, and $p = \left|S^{(i)}\right|$, and $\delta = 2/M_0$ in Algorithm 2.

Both Algorithm 1 (on line 4) and Algorithm 2 (on line 12) requires solving bound constrained nonlinear optimization problems. Currently, the user can choose between IPOPT (Wächter and Biegler 2006) and LMBOPT (Neumaier and Azmi 2015). These solvers are gradient-based solvers that guarantee only local optimality (under certain assumptions). Global optimization is not performed in the current implementation; this is a trade-off between computation time and convergence guarantees. In practice, this trade-off did not result in loosing solutions of any of the considered test problems. The individual blocks of these problems are easy enough for gradient-based (local) solvers; it is only the original (non-decomposed) problem that is difficult to solve, see Subsections 4.1.1 and 4.2.1.

4 Numerical results and discussion

The numerical examples are steady-state models of distillation columns; to understand the displayed results, we recall how chemical engineers traditionally plot the solutions. A distillation column consists of stages; partitioning the Jacobian along the stage boundaries gives the blocks, and the natural order of the stages directly yields the block lower Hessenberg form by virtue of the internal physical layout of distillation columns. Solving the steady-state model of a distillation column is equivalent to solving a square system of equations. Once a solution to this system is available, it is plotted as follows. A slice of the solution vector is created first: A subset of variables is selected that has the same physical meaning in each block. For example, if the variables corresponding to the temperature are selected at each block, a slice is obtained, the so-called temperature column profile, etc. Then, the block index is plotted against the selected value of the variable in this block, see on the left of Figure 3. In

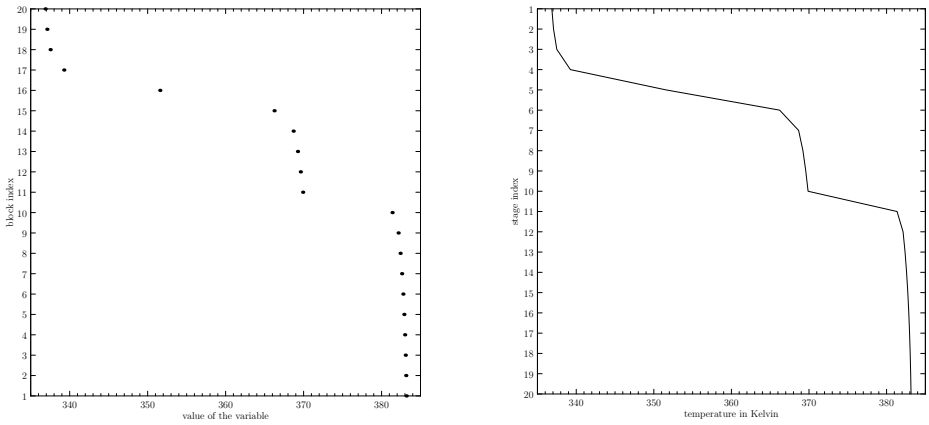

Fig. 3 Plot of a solution vector slice. Left: Block index vs. the value of the variable in the slice. Right: Plot of the same slice as seen in the chemical engineering literature, the so-called temperature column profile. It must be emphasized that the slice is a vector of real numbers, and not a continuous curve.

the chemical engineering literature, not a sequence of discrete points are plotted but a piecewise linear curve passing through these points, and not the block but the stage index is used, see on the right of Figure 3. For the columns considered in this paper, block i corresponds to stage $N-i+1$. It must be emphasized that the solution is a vector of real numbers, and not a continuous curve.

4.1 Multiple steady-states in homogeneous azeotropic distillation

The model equations are the MESH equations: The component material balance (M), vapor-liquid equilibrium (E), summation (S), and heat balance (H) equations are solved. The liquid phase activity coefficient is computed from the Wilson equations. The model and its parameters correspond to the Auto model (Güttinger et al

1997), except for the number of stages N and the feed stage location $N_F = N/2$. The specifications are the feed composition (methanol–methyl butyrate–toluene), the reflux ratio, and the vapor flow rate. The model has been coded in the AMPL (Fourer et al 2003) modeling language; the model files are available in the online supplementary material (Baharev et al 2015). The sparsity pattern of the Jacobian is shown in Figure 4. The ordering was obtained by partitioning the Jacobian along the stage

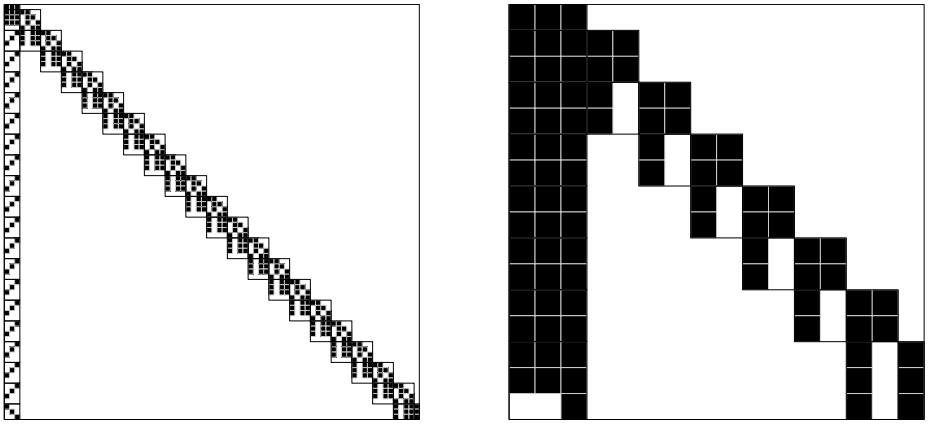

Fig. 4 Sparsity pattern of the Jacobian with the natural block structure as defined by the stages. Left: The column described in Section 4.1, the figure prepared for $N = 20$ stages. Right: The column described in Section 4.2, this column has $N = 8$ stages.

boundaries, which gives the block lower Hessenberg form by virtue of the internal physical layout of distillation columns.

There are three steady-state branches: two stable steady-state branches and an unstable branch; this was experimentally verified in an industrial pilot column operated at finite reflux (Güttinger et al 1997; Dorn et al 1998). Multiple steady-states can be predicted by analyzing columns with infinite reflux and infinite length (Bekiaris et al 1993; Güttinger and Morari 1996; Petlyuk 2004). These predictions for infinite columns have relevant implications for columns of finite length operated at finite reflux.

4.1.1 Published numerical results with continuation methods

Both the conventional inside-out procedure (Boston and Sullivan 1974) and the simultaneous correction procedure (Naphthali and Sandholm 1971) were reported to miss the unstable steady-state solution (Vadapalli and Seader 2001; Kannan et al 2005) (all input variables specified; output multiplicity). However, all steady-state branches were computed (Güttinger et al 1997; Vadapalli and Seader 2001; Kannan et al 2005) either with the AUTO software package (Doedel et al 1995) or with an appropriate continuation method. The initial estimates were carefully chosen with the ∞/∞ analysis (Bekiaris et al 1993; Güttinger and Morari 1996), and special attention was paid to the turning points and branch switching.

4.1.2 Numerical results with the proposed algorithm

All three steady-state solutions of the column with $N = 50$ stages are found when the gradient-based solver IPOPT is run from the starting points generated with the proposed method. This is considered the main achievement of the paper, especially in the light of the earlier results and difficulties published in the literature. As for parameter tuning of the proposed algorithm, the reader is referred to Section 3.4

For the purposes of demonstration, several plots are given for the column with $N = 20$ stages; the column with 50 stages is too long to be appropriate for illustration. Figure 5 shows the three steady-state solutions and those starting points that are the closest to them.

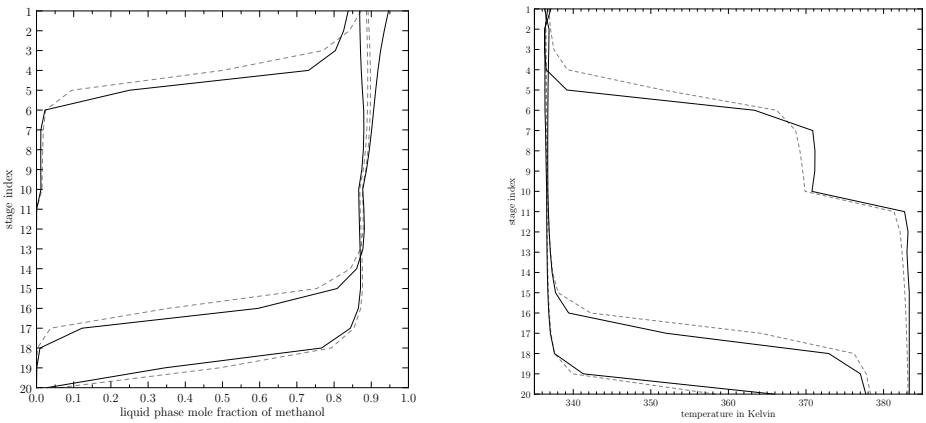

Fig. 5 The three steady-state solutions (dashed gray lines) and those generated starting points (solid black lines) that are the closest to them. The gradient-based solver IPOPT converges to the nearest solution when started from the corresponding starting point. Column description in Section 4.1.

Figure 6 illustrates the extension of the partial column profiles when moving from stage 11 to stage 10, that is, the result of executing the nested loop in Algorithm 1 for each point in the sample. Block i corresponds to stage $N - i + 1$; the computations are performed bottom up, starting at the reboiler (stage 20).

Figure 7 shows the effect of the distinguishing feature of the method, the redistribution. If the redistribution is disabled, the proposed method boils down to the traditional stage-by-stage method. Figure 7 is computed stage-by-stage, exactly from the same bulk composition used for Figure 5. Two out of the three steady-state solutions are lost without the redistribution step.

Figure 8 shows one execution of the redistribution Algorithm 2 at stage 10. New points are forced into the sample; then, only the most distant points are kept.

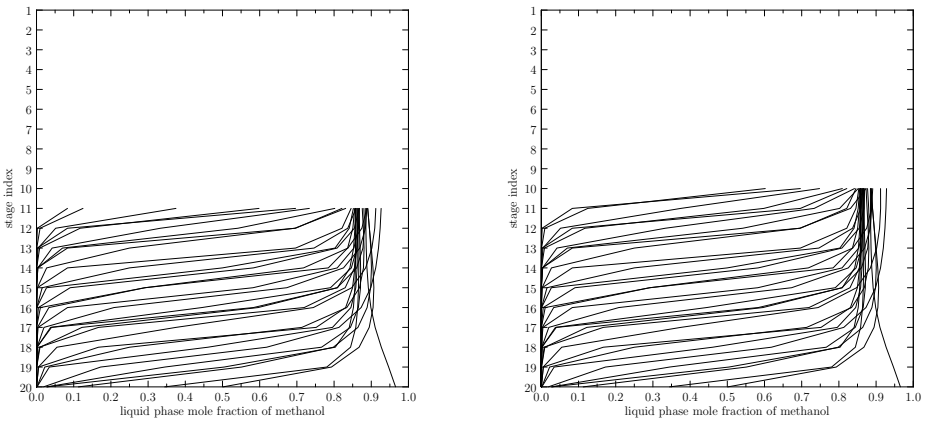

Fig. 6 Extending the partial solutions as moving from stage 11 (left) to stage 10 (right); see the loop starting at line 3 in Algorithm 1. Several partial column profiles are built stage-by-stage, starting from a variety of bulk compositions. Column description in Section 4.1.

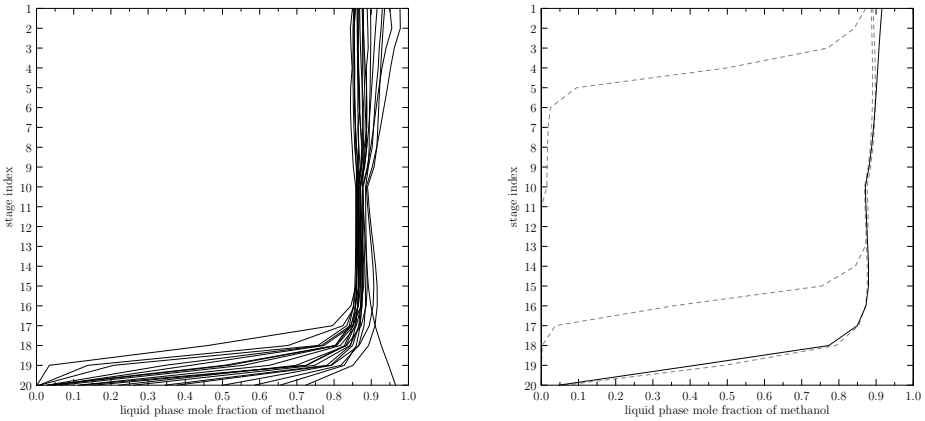

Fig. 7 The lack of redistribution. Left: Several column profiles are built as in the traditional stage-by-stage calculation, starting from a variety of bulk compositions. Right: The dashed gray lines show the 3 different steady-state solutions, two of them do not have any good approximation and therefore are not discovered. Column description in Section 4.1.

4.2 Multiple steady-states in ideal two-product distillation

Multiple steady-states can arise in simple distillation columns with ideal vapor-liquid equilibrium (Jacobsen and Skogestad 1991a). The model equations are taken from the paper of Jacobsen and Skogestad; specifications are: methanol-propanol feed composition, *mass* reflux flow rate and vapor molar flow rate of the boilup. Heat balances are included in the model; the column has $N = 8$ stages. The model has been coded in the AMPL (Fourer et al 2003) modeling language; the model files are available in the online supplementary material (Baharev et al 2015). The sparsity pattern of the Ja-

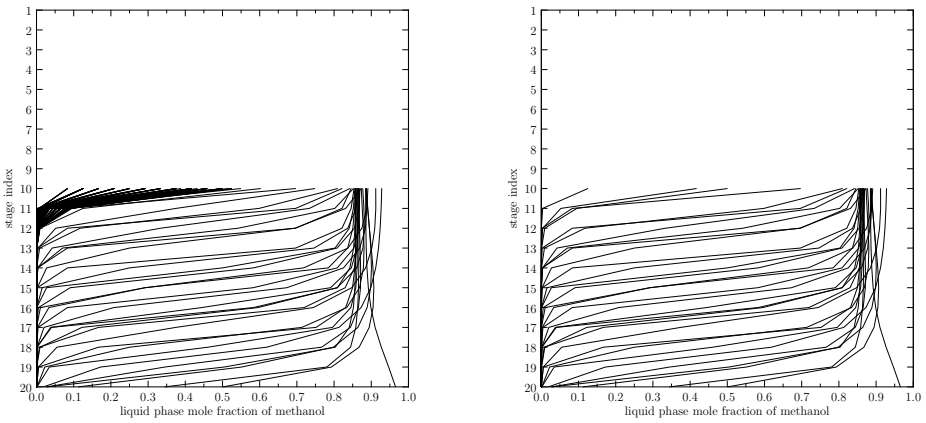

Fig. 8 The redistribution step at stage 10. Left: New values are inserted for mole fractions of methanol, cf. Figure 6. Right: Only the most distant column profile extensions are kept. Column description in Section 4.1.

cobian is shown in Figure 4. The ordering was obtained by partitioning the Jacobian along the stage boundaries, which gives the block lower Hessenberg form.

Two types of multiplicities can be distinguished. The first type can occur when some flows are specified on a mass basis and are due to the nonlinear mass to molar flow rate transformation (Jacobsen and Skogestad 1991a). Figure 9 shows an example for this. There are three steady-state branches: two stable steady-state branches with an unstable branch in between.

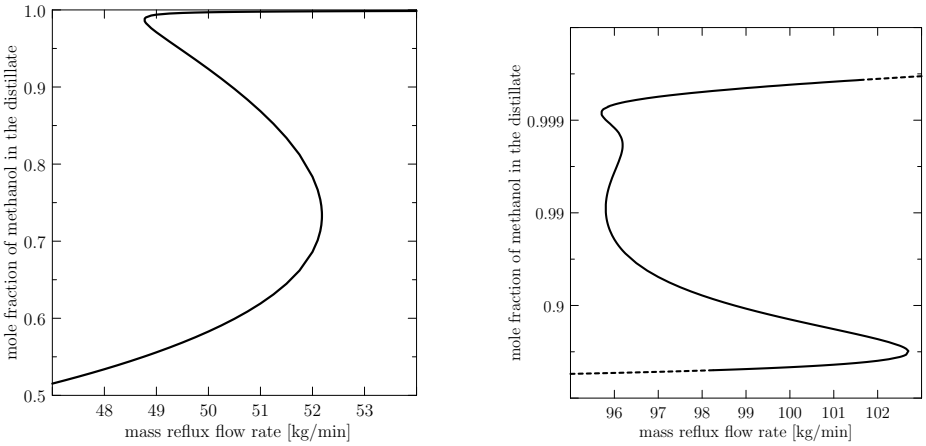

Fig. 9 Bifurcation diagrams, mole fraction of the methanol in the distillate as a function of the mass reflux flow rate. Column description in Section 4.2. Left: Boilup $V_N = 2.0$ kmol/min, constant molar overflow. Right: Boilup $V_N = 3.0$ kmol/min, heat balance included in the model. The dashed part corresponds to negative flow rates at stage 8 (infeasible).

The second type of multiplicities occurs when energy balances are included in the model. This type of multiplicity does not occur for the case of constant molar flows. The region of the different type of multiplicities can overlap (Jacobsen and Skogestad 1991a). Figure 9 illustrates this phenomenon. The model equations have five distinct solutions in a certain range of the reflux flow rate. One of the solutions is infeasible in practice because it would result in negative flow rates. Nevertheless, the proposed algorithm finds all the five solutions; the infeasible one is represented by a dashed line in the bifurcation diagram in Figure 9.

4.2.1 Published numerical results with dynamic simulation

Judging from the conference paper (Jacobsen and Skogestad 1991b), the multiple steady-state solutions were found with dynamic simulation and systematic bifurcation analysis. In general, dynamic simulation and relaxation techniques (at least one set of the MESH equations is cast to unsteady-state form) are recommended for problems which are notoriously difficult to converge (Doherty et al 2008) because of their extreme stability. A disadvantage of these methods is that convergence is generally a very slow process, slowing down even more as the steady-state is approached. A single steady-state solution is found if convergence is reached.

4.2.2 Numerical results with the proposed algorithm

Figure 10 shows the five steady-state solutions, and those starting points that are the closest to them. All five steady-state solutions are found when the gradient-based solver IPOPT is run from these starting points. As for parameter tuning of the proposed algorithm, the reader is referred to Section 3.4

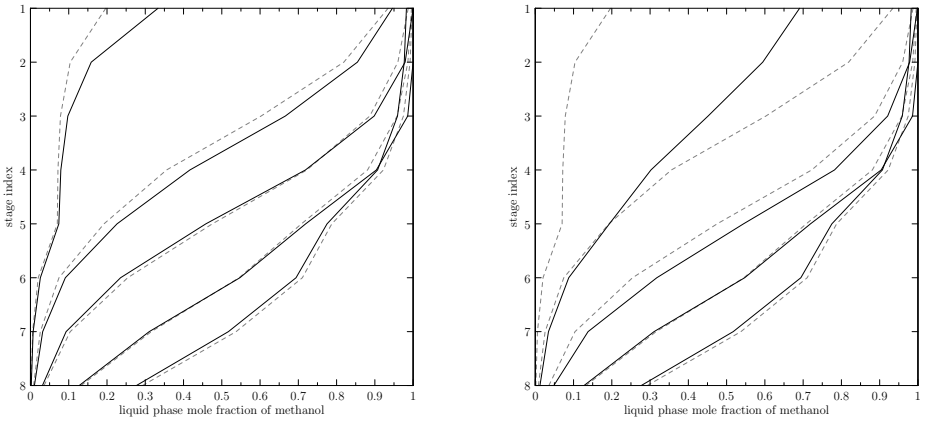

Fig. 10 Left: The five steady-state solutions (dashed gray lines) and those generated starting points (solid black lines) that are the closest to them. The gradient-based solver IPOPT converges to the nearest solution when started from the corresponding starting point. Column description in Section 4.2. Specifications: boilup $V_N = 3.0$ kmol/min, reflux flowrate 96.0 kg/min, cf. Fig. 9. Right: The same with identical initial points for the bulk composition but without redistribution; one solution is not discovered.

The column profiles on the right of Figure 10 are obtained from exactly the same initial bulk composition as before (altogether 25 initial points, only the closest 4 profiles are shown) but without redistribution (stage-by-stage computation). There is one solution that is not approximated well by any of the column profiles; this solutions is eventually lost. Again, this shows the importance of the redistribution step.

5 Conclusions

5.1 Summary of the proposed method

When the proposed method is applied to an underdetermined system of equations in block lower Hessenberg form, it returns a representative sample of the solution manifold. If the underdetermined system is then extended appropriately with additional equations such that it becomes a square system, the sample of the underdetermined part provides a computationally cheap way to generate starting points for gradient-based solvers. If the newly appended equations are changed or even completely replaced, the sample of the (unchanged) underdetermined part can be reused at no cost. This enables the robust computation of bifurcation diagrams; unlike continuation methods, this approach does not rely on the connectedness of branches or connectedness to previously computed points.

5.2 Progress compared to our earlier results

The proposed method was inspired by our recent paper (Baharev and Neumaier 2014). The method published in that paper assumes the availability of an algorithm for parameterizing implicitly defined d-dimensional manifolds ($d > 3$). Such algorithms exist as discussed in Section 2.1; although it is a major undertaking to implement one that is robust, and requires practically no end-user tuning. The redistribution algorithm in Section 3.2 is our attempt to implement an alternative to manifold parameterization. In comparison with higher-dimensional continuation methods, the redistribution algorithm tries to strike a balance between implementation effort and accurate resolution of the solution manifold. When compared to response surfaces like in kriging, the trade-off is between strong resolution guarantees and computational speed: Building surrogate models usually involve sampling, followed by minimizing or maximizing an auxiliary function *globally* to check whether the sample indeed provides sufficient resolution (Jones 2001). Solving the latter global optimization problem is a computationally expensive task. The redistribution algorithm performs the corresponding step only on a best effort basis.

5.3 Parameter tuning

The primary parameter of the algorithm is the sample size M_i in iteration $i = 0, 1, \ldots, N$, as discussed in Section 3.4. Increasing the sample size increases the robustness of the

method; although, a large number of sample points are computed unnecessarily if the sample size is set much larger than what would otherwise be sufficient. Currently, it is left to the user to choose the sample size. It is subject of future research to develop an algorithm that decides on the sample size on the fly, based on the properties of the input problem.

The second most important parameter of the algorithm is h, the number of blocks re-solved after inserting a new point into the sample in the redistribution step. If h is chosen too small (for example $h = 1$), it can eventually lead to loosing some of the solutions. Setting h to a large value ($h \gg 5$) spoils the performance. Currently, it is left to the user to set h. Our future research will start with the adaptive rule outlined in Section 3.4, setting h at each block on the fly.

The same set of parameters were used when computing the distillation columns in this paper; there was no need to re-tune any of the parameters between solving different problems or after changing the number of stages in the column, etc. In our numerical experience, the parameters other than M and h had negligible effects.

5.4 Comparison to alternative state-of-the-art approaches

A numerically challenging distillation column with $N = 50$ stages was considered in Section 4.1. The conventional inside-out and the simultaneous correction procedure were reported to miss the unstable steady-state solution. Results published in the literature show that computing this column requires an appropriate continuation method, carefully chosen initial estimates, and special attention to turning points and branch switching. All three steady-state solutions of this column are found when gradient-based solver IPOPT (Wächter and Biegler 2006) is run from the starting points generated with the proposed method; no user-provided initial estimates are required. Finding all solutions to this challenging problem is considered the main achievement of the paper.

5.5 Future work

The current Java implementation is available online (Baharev et al 2015) together with the models written in AMPL (Fourer et al 2003). This implementation is a research prototype: There is room for improvements both on the algorithmic and on the implementation level. An example for algorithmic improvement is to apply linear interpolation between the generated sample points to produce a starting point that has less constraint violation than any of the sample points, see Section 3.3. Further algorithmic improvement opportunities are creating more sophisticated and adaptive rules for choosing the sample size and the number of blocks to be re-solved in the redistribution step, and more elaborate sampling procedures again in the redistribution step. Examples for the implementation improvements include the followings. The array slicing syntax and the array-oriented approach of NumPy seems more suitable for implementing the proposed algorithm than Java. Although AMPL has its merits, it was not mean to be used for block-oriented modeling; the component-based

modeling methodology in Modelica is well-suited for recovering the block structure automatically (Baharev and Neumaier 2012). The algorithm is currently being reimplemented in Python, and the models in Modelica. The block lower Hessenberg form can be automatically generated with the new Python implementation already (Baharev 2015).

References

Allgower EL, Schmidt PH (1985) An algorithm for piecewise-linear approximation of an implicitly defined manifold. SIAM J Numer Analysis 22(2):322–346

Aspen Technology, Inc (2009) Aspen Simulation Workbook, Version Number: V7.1. Burlington, MA, USA. EO and SM Variables and Synchronization, p. 110.

Auger A, Hansen N (2005) A restart CMA evolution strategy with increasing population size. In: Evolutionary Computation, 2005. The 2005 IEEE Congress on, IEEE, vol 2, pp 1769–1776

Bachmann B, Aronßon P, Fritzson P (2007) Robust initialization of differential algebraic equations. In: 1st International Workshop on Equation-Based Object-Oriented Modeling Languages and Tools (Berlin; Germany; July 30; 2007), Linköping University Electronic Press; Linköpings universitet, Linköping Electronic Conference Proceedings, pp 151–163

Baharev A (2015) URL `http://sdopt-tearing.readthedocs.org`, Exact and heuristic methods for tearing

Baharev A, Neumaier A (2012) Chemical Process Modeling in Modelica. In: Proceedings of the 9th International Modelica Conference, Munich, Germany, pp 955–962

Baharev A, Neumaier A (2014) A globally convergent method for finding all steady-state solutions of distillation columns. AIChE J 60:410–414

Baharev A, Kolev L, Rév E (2011) Computing multiple steady states in homogeneous azeotropic and ideal two-product distillation. AIChE Journal 57:1485–1495

Baharev A, Domes F, Neumaier A (2015) URL `http://www.mat.univie.ac.at/~neum/ms/maniSolSuppl/`, Online supplementary material of the present manuscript; it will be available from the journal upon acceptance

Baker T, Gill J, RSolovay (1975) Relativizations of the P =? NP question. SIAM J of Computing 4:431–442

Bekiaris N, Meski GA, Radu CM, Morari M (1993) Multiple steady states in homogeneous azeotropic distillation. Ind Eng Chem Res 32:2023–2038

Biegler LT, Grossmann IE, Westerberg AW (1997) Systematic Methods of Chemical Process Design. Prentice Hall PTR, Upper Saddle River, NJ

Boston JF, Sullivan SL (1974) A new class of solution methods for multicomponent, multistage separation processes. Can J Chem Eng 52:52–63

Boukouvala F, Ierapetritou M (2013) Surrogate-based optimization of expensive flowsheet modeling for continuous pharmaceutical manufacturing. Journal of Pharmaceutical Innovation 8(2):131–145

Brodzik ML (1998) The computation of simplicial approximations of implicity defined p-dimensional manifolds. Computers Math Applic 36(6):93–113

Brodzik ML, Rheinboldt WC (1994) The computation of simplicial approximations of implicitly defined two-dimensional manifolds. Computers Math Applic 28(9):9–21

Caballero JA, Grossmann IE (2008) An algorithm for the use of surrogate models in modular flowsheet optimization. AIChE Journal 54(10):2633–2650

Cozad A, Sahinidis NV, Miller DC (2014) Learning surrogate models for simulation-based optimization. AIChE Journal 60(6):2211–2227

Davis E, Ierapetritou M (2007) A kriging method for the solution of nonlinear programs with black-box functions. AIChE Journal 53(8):2001–2012

Doedel EJ, Wang XJ, Fairgrieve TF (1995) AUTO94: Software for continuation and bifurcation problems in ordinary differential equations. Tech. Rep. CRPC-95-1, Center for Research on Parallel Computing, California Institute of Technology, Pasadena CA 91125

Doherty MF, Fidkowski ZT, Malone MF, Taylor R (2008) Perry's Chemical Engineers' Handbook, 8th edn, McGraw-Hill Professional, chap 13, p 33

Dorigo M, Birattari M, Stützle T (2006) Ant colony optimization. IEEE Computational Intelligence Magazine 1(4):28–39

Dorn C, Güttinger TE, Wells GJ, Morari M (1998) Stabilization of an unstable distillation column. Ind Eng Chem Res 37:506–515

Duff IS, Erisman AM, Reid JK (1986) Direct Methods for Sparse Matrices. Clarendon Press, Oxford

Eberhart R, Kennedy J (2002) A new optimizer using particle swarm theory. In: Micro Machine and Human Science, 1995. MHS'95., Proceedings of the Sixth International Symposium on, IEEE, pp 39–43

Erisman AM, Grimes RG, Lewis JG, Poole WGJ (1985) A structurally stable modification of Hellerman-Rarick's P^4 algorithm for reordering unsymmetric sparse matrices. SIAM J Numer Anal 22:369–385

Fletcher R, Hall JAJ (1993) Ordering algorithms for irreducible sparse linear systems. Annals of Operations Research 43:15–32

Fourer R, Gay DM, Kernighan BW (2003) AMPL: A Modeling Language for Mathematical Programming. Brooks/Cole USA

Fritzson P (2004) Principles of Object-Oriented Modeling and Simulation with Modelica 2.1. Wiley-IEEE Press

Golub GH, van Loan CF (1996) Matrix Computations, 3rd edn, The Johns Hopkins University Press, Baltimore, USA, p 591

gPROMS (2015) Process Systems Enterprise Limited, gPROMS. `http://www.psenterprise.com`, [Online; accessed 17-November-2015]

Güttinger TE, Morari M (1996) Comments on "multiple steady states in homogeneous azeotropic distillation". Ind Eng Chem Res 35:2816–2816

Güttinger TE, Dorn C, Morari M (1997) Experimental study of multiple steady states in homogeneous azeotropic distillation. Ind Eng Chem Res 36:794–802

Guzman YA, Hasan MMF, Floudas CA (2014) Computational comparison of convex underestimators for use in a branch-and-bound global optimization framework. In: Rassias TM, Floudas CA, Butenko S (eds) Optimization in Science and Engineering, Springer New York, USA, pp 229–246

Gwaltney CR, Lin Y, Simoni LD, Stadtherr MA (2008) Interval Methods for Nonlinear Equation Solving Applications. John Wiley & Sons Ltd., Chichester, UK

Hellerman E, Rarick DC (1971) Reinversion with preassigned pivot procedure. Math Programming 1:195–216

Henao CA, Maravelias CT (2011) Surrogate-based superstructure optimization framework. AIChE Journal 57(5):1216–1232

Henderson ME (2002) Multiple parameter continuation: Computing implicitly defined k-manifolds. International Journal of Bifurcation and Chaos 12(3):451–476

Henderson ME (2007) Higher-dimensional continuation. In: Krauskopf B, Osinga HM, Galán-Vioque J (eds) Numerical Continuation Methods for Dynamical Systems, Dordrecht, The Netherlands: Springer, chap 3, pp 77–115

HSL (2015) A collection of Fortran codes for large scale scientific computation. URL `http://www.hsl.rl.ac.uk`

Jacobsen E, Skogestad S (1991a) Multiple steady states in ideal two-product distillation. AIChE Journal 37:499–511

Jacobsen EW, Skogestad S (1991b) Control of unstable distillation columns. In: Proceedings of the 1991 American Control Conference, IEEE, Boston, MA, USA, pp 773–778

Jiménez-Islas H, Martínez-González GM, Navarrete-Bolaños JL, Botello-Álvarez JE, Oliveros-Muñoz JM (2013) Nonlinear homotopic continuation methods: A chemical engineering perspective review. Ind Eng Chem Res 52(42):14,729–14,742, DOI 10.1021/ie402418e

Jones DR (2001) A taxonomy of global optimization methods based on response surfaces. Journal of Global Optimization 21(4):345–383

Kannan A, Joshi MR, Reddy GR, Shah DM (2005) Multiple-steady-states identification in homogeneous azeotropic distillation using a process simulator. Ind Eng Chem Res 44:4386–4399

Kreinovich V, Kearfott RB (2005) Beyond Convex? Global Optimization is Feasible Only for Convex Objective Functions: A Theorem. Journal of Global Optimization 33(4):617–624

Kröner A, Marquardt W, Gilles E (1997) Getting around consistent initialization of DAE systems? Computers & Chemical Engineering 21(2):145–158

Lewis WK, Matheson GL (1932) Studies in distillation. Ind Eng Chem 24:494–498

Malinen I, Tanskanen J (2010) Homotopy parameter bounding in increasing the robustness of homotopy continuation methods in multiplicity studies. Computers & Chemical Engineering 34(11):1761–1774

Mattsson S, Elmqvist H, Otter M (1998) Physical system modeling with Modelica. Control Eng Pract 6:501–510

Mitsos A, Chachuat B, Barton PI (2009) McCormick-based relaxations of algorithms. SIAM Journal on Optimization 20(2):573–601

Morris A, Montague G, Willis M (1994) Artificial neural networks: studies in process modelling and control. Chemical Engineering Research and Design 72(A1):3–19

Naphthali LM, Sandholm DP (1971) Multicomponent separation calculations by linearization. AIChE J 17:148–153

Neumaier A, Azmi B (2015) LMBOPT – A limited memory method for bound-constrained optimization, URL `http://www.mat.univie.ac.at/~neum/ms/lmbopt.pdf`, in preparation

Ochel LA, Bachmann B (2013) Initialization of equation-based hybrid models within OpenModelica. In: 5th International Workshop on Equation-Based Object-Oriented Modeling Languages and Tools (University of Nottingham; Nottingham, UK; April 19, 2013), Linköping University Electronic Press; Linköpings universitet, Linköping Electronic Conference Proceedings, pp 97–103

de P Soares R, Secchi AR (2003) EMSO: A new environment for modelling, simulation and optimisation. In: Computer Aided Chemical Engineering, vol 14, Elsevier, pp 947–952

Pantelides CC (1988) The consistent initialization of differential-algebraic systems. SIAM Journal on Scientific and Statistical Computing 9(2):213–231

Papadimitrou C (1977) The Euclidean traveling salesman problem is NP-complete. Theoretical Computer Science 4 3:237–244

Pardalos P, Schnitger G (1988) Checking local optimality in constrained quadratic programming is NP-hard. Operations Research Letters 7(1):33–35

Petlyuk FB (2004) Distillation Theory and Its Application to Optimal Design of Separation Units. Cambridge University Press, Cambridge, UK

Piela PC, Epperly TG, Westerberg KM, Westerberg AW (1991) ASCEND: An object-oriented computer environment for modeling and analysis: The modeling language. Computers & Chemical Engineering 15(1):53–72

Rheinboldt WC (1988) On the computation of multi-dimensional solution manifolds of parameterized equations. Numerische Mathematik 53:165–181

Scott JK, Stuber MD, Barton PI (2011) Generalized mccormick relaxations. Journal of Global Optimization 51(4):569–606

Seydel R (2010) Practical Bifurcation and Stability Analysis. In: Antman S, Sirovich L, Marsden J (eds) Interdisciplinary Applied Mathematics, vol 5, Dordrecht, The Netherlands: Springer

Sielemann M, Schmitz G (2011) A quantitative metric for robustness of nonlinear algebraic equation solvers. Mathematics and Computers in Simulation 81(12):2673–2687

Sielemann M, Casella F, Otter M (2013) Robustness of declarative modeling languages: Improvements via probability-one homotopy. Simulation Modelling Practice and Theory 38:38–57

Sipser M (1997) Introduction to the theory of computation. PWS Publ. Comp.

Smith L (2011) Improved placement of local solver launch points for large-scale global optimization. PhD thesis, Ottawa-Carleton Institute for Electrical and Computer Engineering (OCIECE), Carleton University, Ottawa, Ontario, Canada

Smith L, Chinneck J, Aitken V (2013a) Constraint consensus concentration for identifying disjoint feasible regions in nonlinear programmes. Optimization Methods and Software 28(2):339–363

Smith L, Chinneck J, Aitken V (2013b) Improved constraint consensus methods for seeking feasibility in nonlinear programs. Computational Optimization and Applications 54(3):555–578

Soares RP (2013) Finding all real solutions of nonlinear systems of equations with discontinuities by a modified affine arithmetic. Computers & Chemical Engineering 48:48–57

Stadtherr MA, Wood ES (1984a) Sparse matrix methods for equation-based chemical process flowsheeting–I: Reordering phase. Computers & Chemical Engineering 8(1):9–18

Stadtherr MA, Wood ES (1984b) Sparse matrix methods for equation-based chemical process flowsheeting–II: Numerical Phase. Computers & Chemical Engineering 8(1):19–33

Thiele E, Geddes R (1933) Computation of distillation apparatus for hydrocarbon mixtures. Ind Eng Chem 25:289–295

Tiller M (2001) Introduction to physical modeling with Modelica. Springer Science & Business Media

Ugray Z, Lasdon L, Plummer J, Glover F, Kelly J, Martí R (2007) Scatter Search and Local NLP Solvers: A Multistart Framework for Global Optimization. INFORMS Journal on Computing 19(3):328–340, DOI 10.1287/ijoc.1060.0175

Unger J, Kröner A, Marquardt W (1995) Structural analysis of differential-algebraic equation systems – theory and applications. Computers & Chemical Engineering 19(8):867–882

Vadapalli A, Seader JD (2001) A generalized framework for computing bifurcation diagrams using process simulation programs. Comput Chem Eng 25:445–464

Vieira R, Jr EB (2001) Direct methods for consistent initialization of DAE systems. Computers & Chemical Engineering 25(910):1299–1311

Wächter A, Biegler LT (2006) On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. Mathematical Programming 106:25–57