Improving Convergence of Derivative-Based Parameter Estimation with Multistart Parameter Clustering Based on DAE Decomposition

Atya Elsheikh* Katharina Nöh Eric von Lieres†
Research Center Jülich, Institute of Biotechnology 2
{a.elsheikh, k.noeh, e.von.lieres}@fz-juelich.de

Abstract

Derivative-based optimization methods for parameter estimation require good start values in order to converge to the global optimum. A conventional multistart strategy is often not practical for identifying such start values, especially for high dimensional problems. Moreover, the computational efforts for each iteration of the optimizer are significantly increased by the computation of parameter sensitivities. We hence present a multistart recursive clustering strategy that utilizes DAE decomposition algorithms, in particular Tarjan's and tearing algorithms. These algorithms are also used by standard Modelica compilers for improving the performance of solving large DAE systems. Our key concept is to provide a natural decomposition of the parameter estimation problem into smaller clusters (i.e. subproblems), each of which requires fewer start values and less computation. The resulting local minima are taken as start values for enlarged subproblems, and so forth until good start values for the original problem are found. This approach serves to improve global convergence and computational speed of multistart derivative-based optimization strategies for large sparse DAE systems.

Keywords: Parameter estimation, global optimization, cluster methods, DAE decomposition algorithms

1 Introduction

1.1 Problem Specification

Differential algebraic equations (DAE) are widely used in modeling and simulation applications, for example in Electrical Engineering, Biochemical Engineering, Mechanics and Thermodynamics. In most applications the values of some model parameters are not known a priori and must hence be estimated from measured data. A parametrized DAE system is formally given by:

$$F(\dot{x}, x, p, t) = 0, \quad x(0) = x_0 \tag{1}$$

with a function $F : R^{2N+M+1} \rightarrow R^N$ that is sufficiently smooth with respect to the state variables $x \in R^N$ and parameters $p \in R^M$. Typical parameter estimation problems aim at minimizing the distance between simulation results $x(p,t)$ and measurement data $\tilde{x}(t_j) \in R^N$ at discrete time points t_j with $j = 1..T$ in the sense of least squares:

$$r = \frac{1}{2}||Q||_2^2 \in R \tag{2a}$$

$$Q = [q_1, .., q_T] \in R^{N \cdot T} \tag{2b}$$

$$q_j = \tilde{x}(t_j) - x(p, t_j) \in R^N, \quad j = 1..T \tag{2c}$$

Note that $Q \notin R^{N \times T}$ is a vector and not a matrix. For start values $p^0 \in R^M$ that are chosen sufficiently close to a local optimum $p^*_{loc} \in R^M$, the Gauss-Newton algorithm converges to p^*_{loc} by iterating the following scheme [1, 2]:

$$\begin{aligned} p^{i+1} &= p^i - (r^i_{pp})^{-1} \cdot r^i_p \\ &\approx p^i - \left(\left[x^i_p\right]^T \cdot x^i_p \right)^{-1} \cdot \left[x^i_p\right]^T \cdot Q \end{aligned}$$

where

$$\begin{aligned} r^i_p &= \frac{\partial r}{\partial p}(p^i) \in R^M \\ r^i_{pp} &= \frac{\partial^2 r}{\partial p^2}(p^i) \in R^{M \times M} \end{aligned}$$

are first and second order partial derivatives of the residual r with respect to the unknown parameters, and

$$x^i_p = \frac{\partial x}{\partial p}(p^i) \in R^{N \cdot T \times M}$$

*Evonik Industries are acknowledged for financial support within the BMBF co-funded project SysMAP (project no. 0313704)

†To whom correspondence should be addressed.

DOI: 10.3384/ecp09430039

are first order partial derivatives of the model solution x with respect to the unknown parameters, also referred to as parameter sensitivities.

1.2 Common Problems in Derivative-Based Optimization

Practical application of derivative-based optimization methods (for instance Gauss-Newton) for obtaining a global optimum

$$p^* = \underset{p \in \{p^*_{loc}\}}{\arg\min}\, r(p)$$

is typically hindered by the following problems:

1. *Good start values are hard to obtain:* The resulting optimization problem can be efficiently solved when good start values are known. However, from most arbitrarily chosen start values the algorithms either diverge or converge only to a local optimum. This problem will be discussed in more depth in section 1.3.

2. *The Hessian r_{pp} is usually approximated by $(x^i_p)^T \cdot x^i_p$ which is often semi-singular:* In this case the inverse cannot be exactly computed but is usually approximated by a pseudo-inverse using singular value decomposition. This approximation is however inaccurate, in particular for high dimensional matrices in large optimization problems.

3. *Parameter sensitivities x_p are expensive to compute:* These sensitivities are usually computed by either solving the original DAE system (equation 1) together with the associated computationally expensive sensitivity equations:

$$F_{\dot{x}} \cdot \dot{x}_p + F_x \cdot x_p + F_p = 0, \quad x_p(0) = 0 \qquad (3)$$

   or by using less precise finite difference methods.

1.3 Identification of Successful Start Values

The determination of suitable start values for high dimensional parameter estimation problems that are located within the global convergence area of a derivative-based optimization algorithm is not trivial because:

1. The space $S_p \subseteq R^M$ of parameter values that are physically admissible is typically large.

2. The DAE system extended with parameter sensitivities (equations 1 and 3) is usually solvable only in a subspace $S_{sol} \subseteq S_p$.

3. Some parameter values can cause numerical difficulties that are associated with the numerical solver. We denote the subspace covering such parameter values by $S_{diff} \subseteq S_{sol}$. For example, assume that the DAE system (equation 1) together with the sensitivity equations 3 are not stiff in the global optimum p^*, but the optimizer may iterate over parameter values for which these equations are stiff, and hence numerically harder to solve.

4. Nonlinear optimization problems usually have numerous local optima, and global convergence is guaranteed only for start values $p_0 \in N_\varepsilon(p^*)$ from a subspace $S_{N_\varepsilon(p^*)} \subseteq S_p$ around the global optimum p^*.

Start values should hence ideally be chosen within the subspace $S_{conv} = S_{N_\varepsilon(p^*)} \bigcap (S_{sol}/S_{diff})$ in order to efficiently find the global optimum p^*. Conventional multistart optimization strategies are not practical for high dimensional problems in which S_{conv} is notably smaller than S_p. Moreover, computational efforts are significantly increased by the computation of parameter sensitivities.

1.4 Multistart Recursive Clustering

We here present a multistart recursive clustering strategy that is specifically designed to reduce the number of starts required for identifying values in S_{conv} from which the applied derivative-based optimization method globally converges. This multistart strategy heuristically decomposes the optimization problem into smaller clusters (i.e. subproblems), and has similarities with cluster optimization methods. Each cluster C_i defines a parameter estimation subproblem that is characterized by:

- a parameter subset $p_i \in R^{M_i}$ with $\sum_i M_i = M$,
- a variable subset $x_i \in R^{N_i}$ with $\sum_i N_i = N$,
- the corresponding subset of measurement data $\tilde{x}_i \in R^{N_i \times T}$,
- a residual r that is a function only of p_i, x_i and $\tilde{x}_i$ (compare equation 2).

Each cluster C_i is recursively decomposed into subclusters $C_{i,j}$ and so forth. In this way, rather small clusters are created and our strategy begins with separately estimating the respective parameter subsets. Once optima are found, the resulting parameters are taken as new start values and the cluster size is enlarged, and so forth until the complete original system is reconsidered.

In the present study we aim to optimally decompose the parameter estimation problem into smaller subproblems specifically for DAE systems. A related problem has already been researched in the context of speeding up the solution of high dimensional DAE systems, namely finding optimal decompositions of a DAE system into smaller DAE subsystems, each of which solves a subset of state variables. We here determine the clusters C_i with the same DAE decomposition algorithms [7, 8]. These algorithms use intuitive but powerful clustering heuristics to naturally decompose the optimization problem into smaller subproblems with well defined dependencies. The details of this approach will be explained in section 3.

1.5 Practical Aspects

The proposed strategy can effectively solve the previously discussed problems of parameter estimation with derivative-based optimization algorithms:

1. *Fewer starts are required.* The recursive multistart parameter clustering strategy helps to rationally identify good start values from which derivative-based optimization algorithms converge to the global optimum. This optimum can hence be identified with significantly fewer starts for the full system as compared to conventional multistart strategies.

2. *Smaller subproblems are processed most of the time.* The parameter estimation problem for each individual cluster C_i involves fewer parameters p_i, fewer state variables x_i, and mostly also fewer model and sensitivity equations. The full set of system equations is solved only for few iterations from start values that are already close to the global optimum. Moreover, the approximations to the Hessian are often more accurate and numerically better conditioned for small clusters.

3. *Automatic differentiation enables efficient computation of parameter sensitivities.* We apply ADModelica (Automatic Differentiation of Modelica), a self developed tool that exploits high level Modelica compiler techniques for generating Modelica code for efficiently and precisely computing the required parameter sensitivities. Automatic differentiation involves significantly less equations as explicit differentiation and finite difference methods [3, 4, 5]. Moreover, we exploit the known structure of the sensitivity equations for reducing compilation and simulation time on both serial and parallel computers [6].

1.6 Outline

The remainder of this contribution is structured as follows: In section 2 we give an overview of the DAE decomposition algorithms that we reuse in the context of parameter estimation. The implementation of the proposed strategy is then presented in section 3 in a simplified form without low level tricks, although fine tuning could further improve the results that are reported in section 4. In section 5 we conclude with discussion of potential limitations as well as future developments and applications of the presented method.

2 DAE Decomposition

2.1 Structure Digraph of DAE Systems

Directed graphs are used as intermediate representations for symbolical handling and simulation of DAE systems that are automatically generated from Modelica models. Figure 1 shows the simple example of a linear reaction chain, that is mathematically represented by equation 4 with initial values $X_i(0) = X_i^0$ for $i = 1..5$.

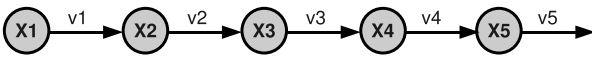

Figure 1: Five nodes with capacities X_i that are connected by flows v_i and subjected to mass conservation.

$$\dot{X}_1 = -v_1 \tag{4a}$$
$$\dot{X}_i = v_{i-1} - v_i, \quad i = 2..5 \tag{4b}$$
$$v_j = v_{j,max} \cdot \frac{X_j}{k_j + X_j}, \quad j = 1..4 \tag{4c}$$
$$v_5 = d \cdot X_5 \tag{4d}$$

The differential equations 4a to 4b are also referred to as rate equations, and the algebraic equations 4c

to 4d determine the flow rates v_j as functions of the maximal flow rates $v_{j,max}$, saturation parameters k_j with $j = 1..4$, and of the degradation rate d. Figure 2a shows the *bipartite graph representation* of the DAE system 4 according to the following definition:

Definition (Bipartite Graph Representation) A bipartite graph representation $G = (V,E)$ of a DAE system (equation 1) consists of a set of vertices:

$$\begin{aligned} V &= V_e \cup V_x \\ V_e &= \{e_i : \textit{equation number } i\} \\ V_x &= \{x_i : \textit{variable number } i\} \end{aligned}$$

and a set of edges:

$$E = \{(e_i, x_j) : x_j \textit{ occurs in equation } e_i\}$$

with $i, j \in \{1, ..., N\}$.

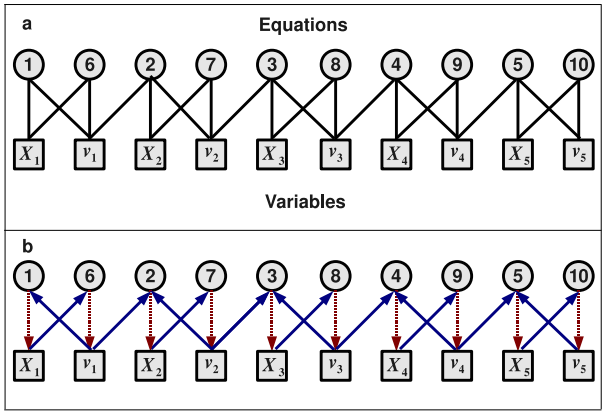

Figure 2: Bipartite graph representation (a) and structure digraph (b) of the example DAE system (equation 4). Red dashed arrow: equation e is explicitly solved for variable x. Blue arrow: variable x is required for solving equation e for another variable.

The bipartite graph representation can be transformed into a *structure digraph* [7], as illustrated in figure 2b for the DAE system from equation 4. This directed graph reveals the causality relations among variables and equations. Ideally each equation e is explicitly solved for one variable x. In practice, however, algebraic loops can occur when a group of equations must be concurrently solved for a group of variables, for instance the ODE system $\dot{x} = y$ and $\dot{y} = x$.

2.2 DAE Decomposition Algorithms

DAE systems that are generated from descriptive Modelica models usually consist of many components and connectors. They are hence high-dimensional and sparse, meaning that only few variables appear in each equation. Instead of solving these equations at once, one can often decompose such systems into smaller blocks of equations. Sequential solution of these blocks reduces overall complexity and consequently speeds up the computation.

The example DAE system (equation 4) can be decomposed into sorted blocks of equations. Application of Tarjan's algorithm [9] to the directed graph in figure 2b yields a set of Strongly Connected Components (SCCs) as shown in figure 3. Each SCC C_i represents an equation block that is solved for a subset of the variables. The SCCs are subjected to a *topological sorting* "<", imposing an order in which the blocks must be solved. $C_i < C_j$ if x_i is required for determining x_j. In figure 3, $C_i < C_j$ if $i < j$, resulting in the solution scheme shown in figure 4.

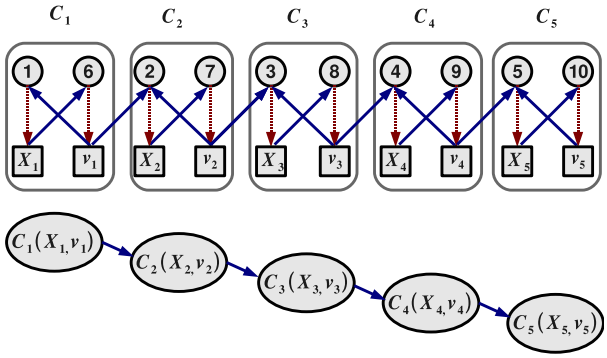

Figure 3: Strongly Connected Components (SCCs) of the structure digraph of the example DAE system.

The block C_1 in figure 3, representing the equations 4a and 4c for $j = 1$, is solved for X_1 and v_1. The value of v_1 is then used to solve the equations represented by the block C_2, and so forth until all variables are determined. A sequential system decomposition as in figure 3 cannot always be achieved. For example consider figure 1 with an additional flow connection from X_5 to X_1. In contrast to figure 3, the resulting digraph now has a cyclic structure, as shown in figure 5.

In this example all equations must be concurrently solved for all variables. Such problems can be solved with Tearing methods [10, 11]: First initial guesses are provided for certain state variables in order to decouple the corresponding SCCs. For example, specification of X_5 would make C_5 independent from C_4. The SCC structure is made acyclic by tearing the graph apart through the *tear variable* X_5 at the connection between C_4 and C_5. The DAE system is then decom-

posed as before, and the remaining state variables X_i with $i = 1..4$ are sequentially determined. With these solutions the initial guess of X_5 is corrected through v_4, and this process is repeated until X_5 does not significantly change any more. The choice of suitable tear variables generally is a key question of tearing methods. In the present study we apply physical knowledge about the modeled system.

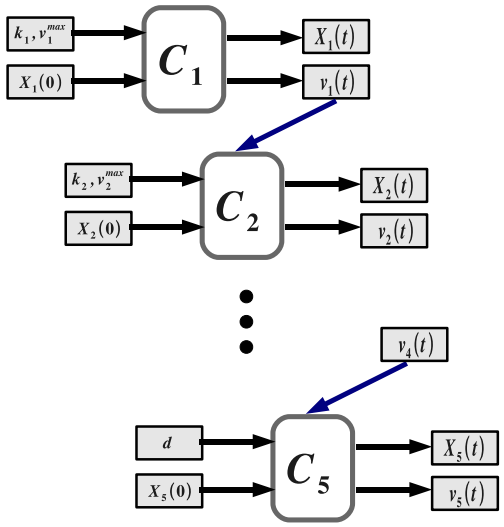

Figure 4: Solution scheme of the equation blocks for the example DAE system (equation 4).

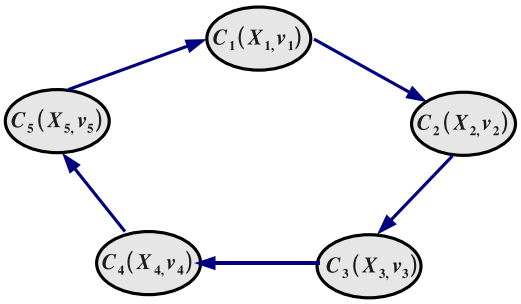

Figure 5: SCCs of the structure digraph of an example DAE system with cyclic structure.

3 Parameter Estimation

We now address the estimation of unknown model parameters by minimizing the distance between simulation results and measurement data. Our basic idea is to decompose the DAE system into SCCs as described in the previous section, and to treat the individual clusters C_i as separate parameter estimation subproblems.

3.1 Example with Linear Structure

Reconsider the model from equation 4 (figure 1), and suppose that $\tilde{X}(t_j) \in R^5$ are corresponding measurement data at discrete time points t_j with $j = 1..T$. Each of the parameter estimation subproblems is characterized by:

- the parameters k_i and $v_{i,max}$ for $i = 1..4$ or, respectively, d for $i = 5$,
- the variables $X_i(t)$ and $v_i(t)$ that equation block C_i is solved for,
- the corresponding measurements $\tilde{X}_i(t_j)$ at times t_j with $j = 1..T$.

The dependency between the subproblems is well-defined, because the fluxes $v_{i-1}(t)$ that are required in an equation block C_i are always determined from the previous equation block C_{i-1} (see figure 4). The relation $C_i < C_j$ for $i < j$ enables to estimate parameters k_j and $v_{j,max}$ from measurements X_j after k_i and $v_{i,max}$ have been estimated for $i < j$.

3.2 Algorithm

Algorithms 1 and 2 formally describe the procedure of our multistart recursive parameter clustering strategy. Three inputs are required:

1. C: A cluster specifying the processed parameter estimation subproblem, in particular the involved parameter, variable and data subsets, as well as the space of feasible parameter values (see section 1.3).
2. *OptAlg*: The optimization strategy to be applied, for example Gauss-Newton algorithm.
3. N: The number of start values for estimating the involved parameter set with a multistart strategy. Alternatively, algorithm 2 can be called with a specific set of start values.

Algorithm 1 first attempts to optimize the parameters in a given cluster C. Algorithm 2 is called for processing a conventional multistart strategy (line 1). If an acceptable residual r is achieved, the solution is returned (line 2), and otherwise the optimization problem is further decomposed.

Algorithm 1 Multistart Clustering Strategy

ClusterOptAlg(C, $OptAlg$, N)
input Cluster: C
Optimization Algorithm: $OptAlg$
Number of Start Values: N
output Optima: $p_C^{1..M}$ with $M \leq N$

1: $p_C^{1..M}$ = Multistart(C, $OptAlg$, N)
2: **if** IsAcceptable($p_C^{1..M}$) **then**
3: return
4: **end if**
5: **if** IsDecomposable(C) **then**
6: $[C_k, <]$ = SCC(C)
7: **for all** k **do**
8: $p_{C_k}^{1..M_k}$ = ClusterOptAlg(C_k, $OptAlg$, N)
9: **for all** j **do**
10: **if** $C_k < C_j$ **and** adjacent(C_k,C_j) **then**
11: $C_j \Leftarrow x_k(p_{C_k}^{1..M_k})$
12: **end if**
13: **end for**
14: **end for**
15: ▷ construct start values from local optima
16: $p_0^{1..N} = \cup_k p_{C_k}^{1..M_k}$
17: $p_C^{1..M}$ = Multistart(C, $OptAlg$, $p_0^{1..N}$)
18: **else**
19: ▷ Decompose using a tear variable x_t
20: $[C_k, <]$ = Tearing(C,x_t)
21: ▷ Start values of all parameters
22: $C_1 \Leftarrow p_0^{1..N}$
23: ITR = 0
24: **while** ITR < MAX_ITERATIONS **do**
25: **for all** k **do**
26: $p_{C_k}^{1..M_k}$=Multistart(C_k, $OptAlg$, N)
27: **for all** j **do**
28: $C_j \Leftarrow p_{C_k}^{1..M_k}$
29: **end for**
30: **end for**
31: $p_0^{1..N} = \cup_k p_{C_k}^{1..M_k}$
32: $p_C^{1..M}$ = Multistart(C, $OptAlg$, $p_0^{1..N}$)
33: **if** IsAcceptable($p_C^{1..M}$) **then**
34: return
35: **end if**
36: ITR = ITR + 1
37: **end while**
38: **end if**

The type of decomposition in algorithm 1 depends on the structure of the underlying DAE system: Linear structures are processed in lines 6-17, and cyclic structures in lines 19-37. The constructed subproblems are sequentially solved (lines 8,26), according to the operator "<". The results of solved clusters are passed to dependent clusters (lines 9,27). Finally, the original problem is reconsidered with the resulting local optima as start values (lines 15-17,31-32).

Algorithm 2 Multistart Strategy

Multistart(C, $OptAlg$, N or $p_0^{1..N}$)
input Cluster: C
Optimization Algorithm: $OptAlg$
(Number of) Start Values: N or $p_0^{1..N}$
output Optima: $p_C^{1..M}$ with $M \leq N$

1: **if** 3rd Input is N **then**
2: $p_0^{1..N}$ = generateStartValues(C, N)
3: **end if**
4: ▷ compute optima from start values
5: $p_C^{1..N}$ = OptAlg(C, $p_0^{1..N}$)
6: ▷ Remove poor local optima
7: $p_C^{1..M}$ = Filter($p_C^{1..N}$)

3.3 Implementation Details

Algorithm 2 applies the optimization algorithm $OptAlg$ to compute local optima $p_C^{1..N}$ of a cluster C_i from N start values that are either self-generated or passed as $p_0^{1..N}$. In our actual implementation, the resulting local optima of each cluster are sorted with respect to the quality of their corresponding residuals r_i. A χ^2-test is then performed, and the worst local optima are removed such that the remaining optima follow a uniform distribution with prespecified confidence level. DAE systems with linear structure are processed with higher confidence levels than DAE systems with cyclic structure.

Moreover, we apply tearing heuristics not for seeking decompositions that are most optimal for efficiently solving the DAE systems, but rather aim at constructing clusters that are small and require only few start values. However, automatic recognition of decompositions that are optimal for the parameter estimation problem is not yet implemented, and tear variables are manually chosen on the basis of physical knowledge about the model topology.

3.4 Example with Linear Structure (continued)

Figure 6 illustrates a run of our algorithm for the example DAE system with linear structure (equation 4, figure 1). First, the parameters of cluster C_1 are estimated with the Gauss-Newton algorithm and a conventional multistart strategy. This strategy generates

N start values for k_1 and $v_{1,max}$, respectively. From $M \leq N$ of these start values the optimization algorithm converges to local optima $k_1^{1..M}$ and $v_{1,max}^{1..M}$.

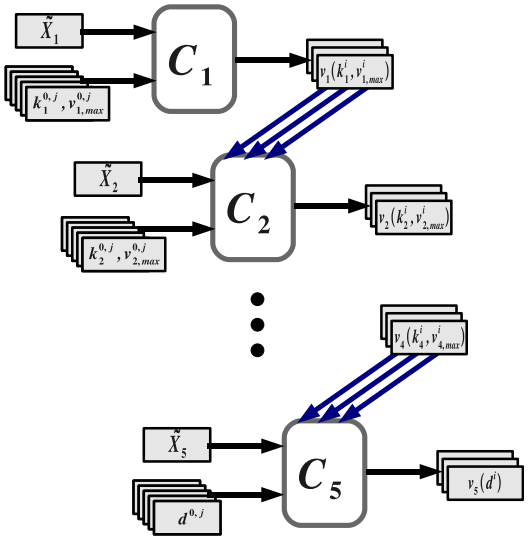

Figure 6: Multistart clustering strategy for the example DAE system with linear structure (equation 4).

In case the optimization algorithm has not converged for all start values, $N-M$ optima are randomly chosen and copied in order to maintain a full set of N optima. The corresponding N solutions for the state variable v_1 are passed to the next cluster, C_2. The same process is then repeated for C_2 with N start values for k_2 and $v_{2,max}$, respectively, that are randomly paired with the N solutions for v_1, and so on until all parameters are estimated.

Next, larger clusters are constructed by merging smaller clusters. For instance C_1 is merged with C_2, and C_3 with C_4. We apply a binary tree data-structure in order to always merge clusters that are adjacent to each other. Start values are not newly generated at this stage, but the optima from the merged clusters are used. For instance, the optima that were individually estimated for clusters C_1 and C_2 are now together used as start values for the cluster that is merged from C_1 and C_2. This process is recursively performed until the original problem is reconsidered with very good start values for global optimization.

3.5 Example with Cyclic Structure

Figure 7 illustrates a run of our algorithm for the example DAE system with cyclic structure (figure 5). The clusters are defined as before, but the clustering strategy is somewhat different, because the whole DAE system must be concurrently solved for the state variables of all clusters when the parameters of one cluster are estimated. This explains the difference between lines 9 and 27 in algorithm 1.

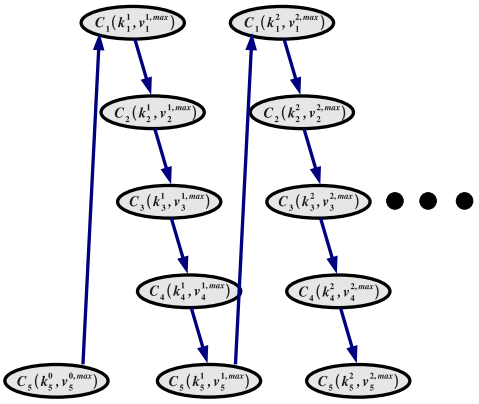

Figure 7: Multistart clustering strategy for the example DAE system with cyclic structure.

We treat parameter estimation problems with cyclic structure similarly to the decomposition of DAE systems with cyclic structures for fast solution. Again, N start values are chosen for each unknown parameter (line 21). The N initial guesses for one cluster are fixed, for instance k_5 and $v_{5,max}$ in cluster C_5. Then the parameters of all clusters are sequentially estimated with a conventional multistart strategy. The resulting optima are passed to all clusters (line 27), and the values of those parameters that were originally fixed are refined. Finally the parameters of the original DAE system are estimated with the combined optima of the clusters as start values. This procedure is repeated for several rounds, until either the residual r of the original DAE system drops below a prespecified threshold or the number of rounds exceeds a predefined maximum.

4 Benchmark

Equation 5 defines an abstract example that we use as benchmark for comparing the performance of our multistart clustering strategy with the conventional multistart strategy. The benchmark is again a reaction network model that consists of a linear pathway ($X_1 - X_6$) and a cyclic pathway ($X_6 - X_{12}$). Simulation results without additional noise are used as synthetic data for

re-estimating the model parameters. We applied the Gauss-Newton optimization algorithm, and initially choose the interval $(\frac{p^*}{2}, 2p^*)$ around the known global optimum as admissible region S_p for each parameter.

$$\dot{X}_1 = -v_1 \quad (5a)$$
$$\dot{X}_i = v_{i-1} - v_i, \qquad i = 2..5 \quad (5b)$$
$$\dot{X}_6 = v_5 + v_{12} - v_6 \quad (5c)$$
$$\dot{X}_i = v_{i-1} - v_i, \qquad i = 7..12 \quad (5d)$$
$$v_j = v_{j,max} \cdot \frac{X_j}{k_j + X_j}, \quad j = 1..12 \quad (5e)$$

Table 1 documents that we used ten times fewer attempts for our clustered multistart strategy than for the conventional approach with random start values. However, most attempts of the clustered strategy converged to the global optimum, whereas the conventional multistart strategy did not at all converge from 1000 start values that were arbitrarily chosen in S_p.

Table 1: Conventional vs. clustered multistart strategy

Multistart	Conventional	Clustered
# of attempts	1000	100, $N = 10$
Best Quality of r	∞	10^{-5}
Success Rate	0%	96%
# of Simulations	2223	980 / attempt

Table 2 shows that the success rate decreases when the admitted parameter region S_p is enlarged to $(\frac{p^*}{4}, 4p^*)$, even if the number of attempts per cluster is increased from 10 to 20. More attempts improve the probability of finding successful start values at the cost of computational effort.

Table 2: Effect of start value region and number

Start Values	$S_p : p_0 \in (\frac{p^*}{4}, 4p^*)$	
# of attempts	100	
N	10	20
Success Rate	16%	44%
# of simulations / attempt	2550	4906

Table 3 illustrates how the residual r improves with each iteration in the cyclic part of the model for $N = 20$ and $S_p \subset (\frac{p^*}{4}, 4p^*)$. The residual r is significantly reduced in the very first iteration, since the average residual for the initial guesses varied around 890. In attempts 1 to 5 start values close to the global optimum were found after several iterations. After each iteration, the strategy attempts to improve the solution of the tear variable (X_{12} in this example). In attempts 2 and 6 the start values diverged, though small residuals were achieved.

Table 3: Residuals in cyclic DAE part

Attempt	1	2	3	4	5	6
Iter 1	2.59	1.76	1.91	4.83	2.85	3.64
Iter 2	1.19	1.47	0.00	0.89	0.90	2.55
Iter 3	0.79	2.66	-	0.41	0.26	1.47
Iter 4	0.33	-	-	0.00	0.00	1.19
Iter 5	0.00	-	-	-	-	0.81

5 Conclusions and Outlook

In this contribution we have presented a multistart recursive clustering strategy for efficiently estimating the parameters of DAE systems with derivative-based optimization algorithms. The algorithm has been illustrated by two simple examples with and without cyclic dependencies in the solved DAE system. A slightly more complex example that combines linear and cyclic structures has been used as benchmark for comparing the performance of our clustering strategy with the conventional multistart strategy.

At some points we have dealt with special cases, and the algorithms and examples can be generalized and extended in various directions:

- In the presented examples each parameter occurs only in one equation, and clear physical causality relation exist between the variables and the parameters. In general, however, the parameters must be included in the causality analysis.
- The benchmark was manually constructed. System analysis and clustering can be implemented using the ADModelica tool, which provides a suitable infra-structure for analyzing Modelica models and for implementing DAE decomposition algorithms.
- Real measurements are afflicted with errors, which complicates the evaluation and comparison of residual values for different optima.
- Redundant computations can be avoided by storing and recognizing local optima that were already found in earlier computations [13].
- Other optimization algorithms can be applied, for example Levenberg-Marquardt. Different clus-

ters might even be optimized with different algorithms, which might be particularly useful for large models with many variables and parameters. Hyper-heuristics [12] can help to identify the suitable algorithms for each subproblem.

References

[1] A. Antoniou, W. Lu. Practical Optimization, Algorithms and Engineering Applications, Springer Verlag, 2007.

[2] J. Nocedal, S. J. Wright. Numerical Optimization, Springer Series in Operations Research, 2000.

[3] A. Elsheikh and W. Wiechert. Automatic sensitivity analysis of DAE systems generated from equation-based modeling languages. Pages 235-246 in C. H. Bischof, et al. (editors). Advances in Automatic Differentiation. Springer, 2008.

[4] A. Elsheikh, S. Noack and W. Wiechert. Sensitivity analysis of Modelica applications via automatic differentiation. In 6th International Modelica Conference, Bielefeld, Germany, March 3-4, 2008.

[5] A. Griewank. Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation, Frontiers in Applied Mathematics, SIAM, 2000.

[6] X. Ke. Tools for sensitivity analysis of Modelica models, Master Thesis, University of Siegen, 2009.

[7] F. E. Cellier. Continuous System Modeling. Springer Verlag, 1991.

[8] K. Murota. Systems Analysis by Graphs and Matroids, Springer Verlag, 1987.

[9] R. Tarjan. Depth-First Search and Linear Graph Algorithms, SIAM Journal on Computing 1 (1972): 146-160.

[10] H. Elmqvist and M. Otter. Methods for tearing systems of equations in object oriented modeling. In ESM'94 European Simulation Multiconference, Barcelona, Spain, June 1-3 1994.

[11] G. Kron. Diakoptics - The piecewise solution of large-scale systems. MacDonald & Co. London, 1963.

[12] E. Özcan, B. Bilgin, and E. E. Korkamz. A comprehensive analysis of hyper-heuristics, Intelligent Data Analysis 12 (2008): 3-23.

[13] C. Voglis and I.E. Lagaris. Towards ideal multistart. A stochastic approach for locating the minima of a continuous function inside a bounded domain. Applied Mathematics and Computation 213 (2009): 216-229.

