An Efficient Multiple Shooting Based Reduced SQP Strategy for Large-Scale Dynamic Process Optimization

Part II: Software Aspects and Applications

Daniel B. Leineweber

Process Technology, Bayer AG, D-51368 Leverkusen, Germany

Andreas Schäfer Hans Georg Bock* Johannes P. Schlöder

Interdisciplinary Center for Scientific Computing (IWR)
University of Heidelberg, D-69120 Heidelberg, Germany

Abstract

As model based optimization techniques play a more and more important role in the chemical process industries, there is a great demand for ever more efficient and reliable process optimization software. In the first part of this paper, the theoretical aspects of a tailored multiple shooting based solution strategy for dynamic process optimization have been presented (Leineweber *et al.*, 2002 [11]). The current second part describes software aspects of the specific implementation MUSCOD-II and provides numerical results for several application examples. MUSCOD-II has been coupled with the dynamic process modeling software gPROMS via the standard ESO interface of CAPE-OPEN. Thereby, an advanced dynamic optimization platform for integrated batch processes has been created, where each process stage is separately modeled in gPROMS, and the multistage dynamic optimization problem is assembled and solved with MUSCOD-II. The code has also been parallelized based on the portable MPI standard.

It is shown that the use of directional sensitivities becomes very important for larger problems with many algebraic variables, leading to drastically reduced computing times compared to strategies with complete constraint linearization. In addition, gPROMS ESO models are compared with classical Fortran models in terms of computational performance, and it is found that only a moderate loss of performance occurs if so-called in-process ESOs are employed. Finally, it is demonstrated that a significant speed-up can be obtained through parallel function and gradient evaluations.

Keywords

Dynamic Optimization Software, CAPE-OPEN Standard, Equation Set Object (ESO), Multiple Shooting, Structured RSQP Methods, Directional Sensitivities, Integrated Batch Processes, Multistage Optimal Control Problems, Parallel Computing

1 Introduction

The merit of a strategy for dynamic process optimization can be assessed only on the basis of a given implementation. For the reduced-space boundary value problem (BVP) approach discussed in Part I of this paper, a software realization is provided by the modular dynamic optimization package MUSCOD-II (Leineweber, 1999 [10]). Due to its unique capabilities for the treatment of *multistage* dynamic optimization problems, MUSCOD-II enables a straightforward optimization of complete integrated batch processes consisting of several different interconnected process stages, e.g., a sequence of reaction and separation steps. It also allows the direct calculation of optimal cyclic steady states (CSS) for cyclic batch operations, where material and energy are transferred between subsequent batches. Last but not least, the code has been very successfully adapted to nonlinear model predictive control (NMPC), see e.g., Diehl *et al.* (2002 [6]), Diehl (2001 [5]), and Nagy *et al.* (2000 [13]).

However, regardless of its numerical sophistication, a dynamic optimization solver is of little practical value unless it can be interfaced with commonly used modeling environments like SPEEDUP (Pantelides, 1988 [15]) – which has now been superseded by its successor Aspen Custom Modeler (ACM) – or gPROMS (Barton and Pantelides, 1994 [1]). Dynamic process models created within such modeling environments

*corresponding author (Email: bock@iwr.uni-heidelberg.de)

have almost entirely replaced classical Fortran models in industry. The significant progress made over the past 10–15 years is reflected by the latest generation of process modeling software. A prominent example is the gPROMS system which has been commercialized by PSE Ltd., London: gPROMS provides a powerful and elegant language for the hierarchical modeling of complex dynamic processes, efficient and robust numerical solvers, and an open software architecture. The latter adopts several of the interface standards defined by the European CAPE-OPEN project (Braunschweig *et al.*, 2000 [4]).

For instance, the concept of an *Equation Set Object (ESO)* enables gPROMS to expose a DAE process model to an external model based application like MUSCOD-II. The ESO interface allows access to the structure of the system (i.e., the number of variables and equations, and the sparsity pattern of the Jacobian), as well as to information on the variables involved (i.e., their names, current values, and lower and upper bounds). It also allows modifying the current values of the variables and obtaining the corresponding values of the residuals and the Jacobian. The communication between gPROMS (model server) and the external application (client) is based on the industry standard middleware CORBA (Object Management Group, 1997 [14]), and hence it is completely independent of the platforms and programming languages employed. Furthermore, client and server can either be dynamically linked into one single process (in-process ESO), or they can be run as separate processes, possibly on different computers over a network (out-of-process ESO).

Numerical results are presented for two reaction and two distillation problems with state dimensions ranging from less than 10 to more than 600. All the models have been implemented in gPROMS as well as in Fortran in order to allow a comparison of in-process ESO, out-of-process ESO and Fortran in terms of computational performance. One of the examples comprises two coupled batch reaction stages which are described by DAE models of different state and control dimensions, showing that general multistage problems can be solved without introducing dummy variables and equations. The larger distillation examples allow to demonstrate the advantage of using directional sensitivities and parallelization.

It should be noted that the overall performance of our multiple shooting based strategy critically depends on the efficiency of the state and sensitivity integrations, particularly in case of large-scale applications. For the solution of all application problems presented in this paper, the advanced BDF code DAESOL (Bauer, 2000 [2]) has been used as DAE integrator within MUSCOD-II. This integrator is capable of calculating directional sensitivities in a very efficient manner.

2 MUSCOD-II: A Modular Dynamic Optimization Framework

2.1 General Software Structure

MUSCOD-II has been implemented as a modular package in ANSI C/C++ and Fortran 77. Portability of the mixed-language code is ensured by using a specifically designed C/Fortran interface (Leineweber, 1999 [10]). The software runs on several Unix and Windows platforms. Apart from optimal control and design optimization problems in DAEs and ODEs, multipoint BVPs (no optimization) as well as standard NLP problems (no dynamic model) can be formulated and solved in a unified fashion, i.e., the full range of possible special cases of the problem formulation is covered. Due to its modular structure, MUSCOD-II allows using various SQP-type strategies for the solution of the discretized problem, and it accomodates many different external integrators.

Figure 1 schematically depicts the architecture of the model solver and model stage interfaces of MUSCOD-II. The external integrators are "sandwiched" between the two generic interface layers IND and MSTAGE, decoupling them from both the RSQP optimizer and the model stage implementation. This allows using widely different DAE and ODE solvers in a "plug and play" manner. It is even possible to employ a different model solver on each model stage. Observe that a set of nontrivial stage transition conditions between two dynamic model stages can be interpreted as an additional *algebraic* model stage of zero duration. These algebraic stages are handled by the transition solver STRANS, a special and very simple model solver implementation.

A discussion of the internal modular structure of the RSQP optimizer MSSQP is beyond the scope of this paper, as our main focus here is on model-related issues. For further detail on the implementation of the RSQP variant presented in Part I, see Leineweber (1999 [10]).

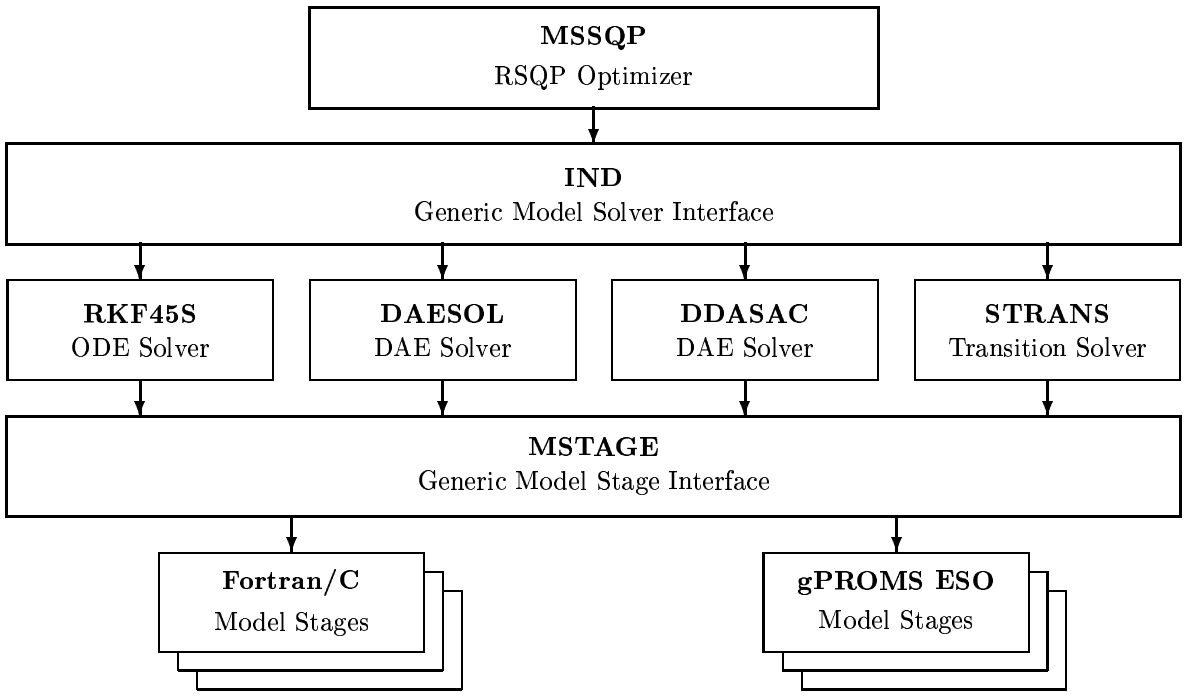

Figure 1: Architecture of MUSCOD-II (Model Solver and Model Stage Interfaces).

2.2 Coupling with gPROMS

The layered architecture of MUSCOD-II together with the standard ESO interface of gPROMS enables a straightforward coupling of both packages. At the heart of this coupling is the newly developed module GPMSTAGE which provides a specific implementation of the generic model stage interface MSTAGE, wrapping the gPROMS ESO interface and automatically generating an ASCII input file with problem specs required by MUSCOD-II. No changes had to be made to other parts of the optimization code.

As shown in Figure 2, the specification of a dynamic optimization problem just requires a standard gPROMS input file which describes the DAE process model, and an auxiliary ASCII input file which defines, e.g., the controls and parameters of the problem by listing the corresponding variable names in the model. An existing gOPT problem formulation (gPROMS Manual, 2000 [8]) can of course be adapted very easily and quickly.

During the initialization phase, the gPROMS input file is parsed by gPROMS, creating an ESO which contains a representation of the model residuals and the Jacobian as well as additional information on the model and its variables. The information required to set up the dynamic optimization problem for MUSCOD-II – e.g., the structure of the DAE system, the variable names, initial values, and bounds – is extracted from the ESO and the GPMSTAGE input file. A so-called start integration is performed in order to generate initial guesses for the values of the states at the multiple shooting nodes and to determine suitable scaling factors. All this information is written to the MUSCOD-II input file.

Then the actual solution phase is entered. On each SQP iteration, state and sensitivity integrations are performed on the multiple shooting intervals to calculate the continuity condition residuals and their derivatives. The model residual and Jacobian values required by the DAE integrator are directly obtained from gPROMS through calls of the corresponding ESO methods.

2.3 Treatment of General Multistage Problems

Batch processes in the chemical process industries typically comprise several different interconnected process stages, e.g., a sequence of reaction and separation steps. Additional couplings are often introduced by a cyclic process operation involving one or more internal "recycle loops", e.g., the recycling of slop cuts

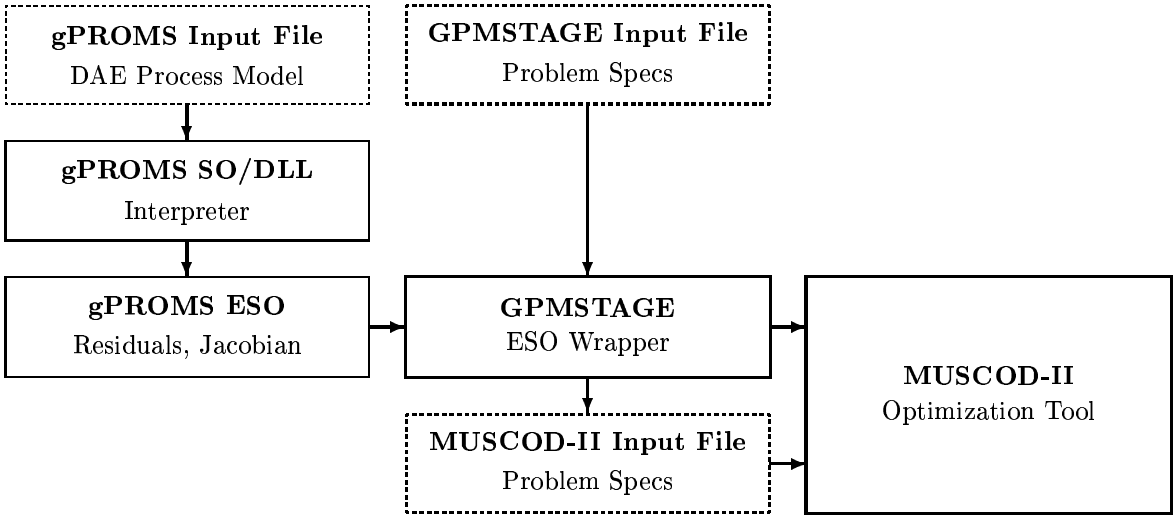

Figure 2: Coupling of gPROMS and MUSCOD-II (In-Process ESO).

and unused reactants. The rigorous optimization of such complex integrated batch processes is a major challenge – in general, it requires the simultaneous treatment of all process stages and their interconnections within one large dynamic optimization problem. Most currently available process optimization packages do not even allow to *formulate* this kind of multistage problem.

As can be seen in Figure 3, gPROMS/MUSCOD-II provides a flexible and powerful optimization platform for general integrated batch processes. Each process stage is separately modeled in gPROMS, and the multistage dynamic optimization problem is assembled and solved with MUSCOD-II. Since gPROMS itself does not support general multistage formulations, the stage transition conditions and the definition of the overall process structure must be provided in a MUSCOD-II-specific format.

2.4 Parallelization Concept

In Part I of this paper, it has already been shown that the DAE integrations (including the generation of directional sensitivities) on different multiple shooting intervals are completely decoupled and hence can be performed in parallel. The parallelization of our algorithm at this level is quite straightforward: basically, just one loop must be replaced by an equivalent parallel construct. Since the DAE integrations are usually by far the most expensive part of the method, a significant speed-up can be obtained through little coding effort. It should be noted that this parallelization concept is particularly well suited for workstation clusters and similar architectures, because the parallelism exploited is relatively coarse.

A simple but effective heuristic load balancing strategy is used in order to enhance the efficiency of the parallel computation, as the CPU times for the DAE integrations tend to differ greatly on the multiple shooting intervals. Our strategy involves estimating the relative computational costs of all integration tasks to be performed for the current evaluation and then assigning the tasks to free processors in the order of decreasing cost. Usually, the task CPU times measured on the *previous* evaluation provide good estimates of the relative computational costs on the current evaluation. If these estimates are correct, i.e., the tasks are actually performed in the order of decreasing cost, then the resulting total processing time can be shown to be at most 33% larger than the time required for the optimal task sequence (Graham, 1969 [9]). Observe that finding this optimal task sequence is in general an NP-complete problem and therefore does not provide a practical alternative to the above heuristic task ordering. Of course, the number of multiple shooting intervals should be chosen significantly higher than the number of processors to enable effective load balancing.

Figure 4 shows the master-slave processor configuration employed by MUSCOD-II: the master processor performs the overall RSQP algorithm and the load balancing, while the DAE integration (IND) tasks are assigned to the available n slave processors. Each slave processor runs its own DAE integrator and

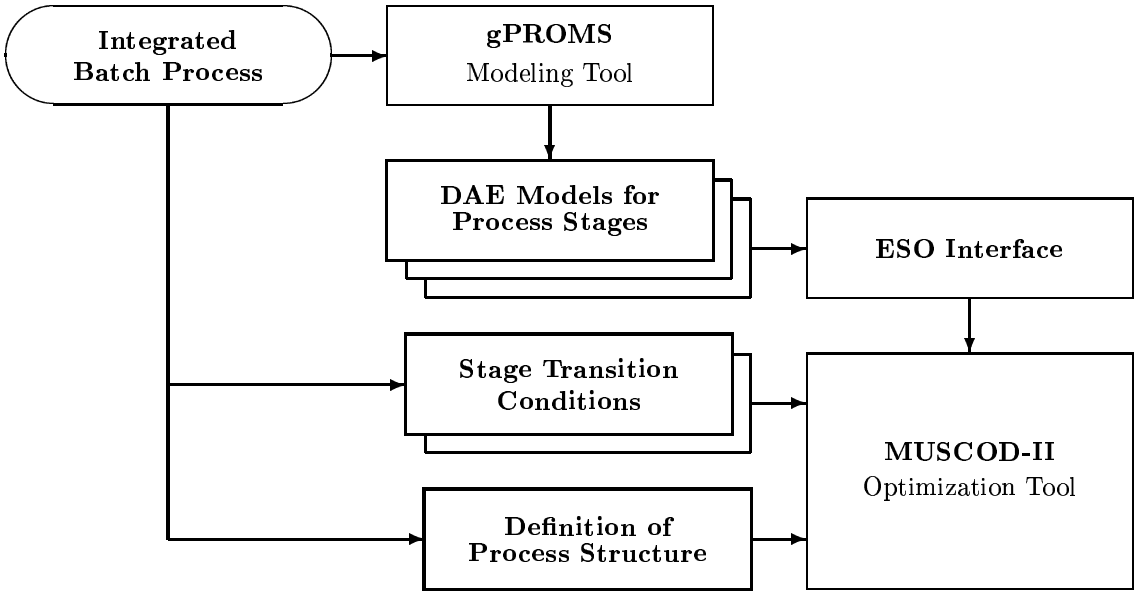

Figure 3: Treatment of General Multistage Problems (gPROMS/MUSCOD-II).

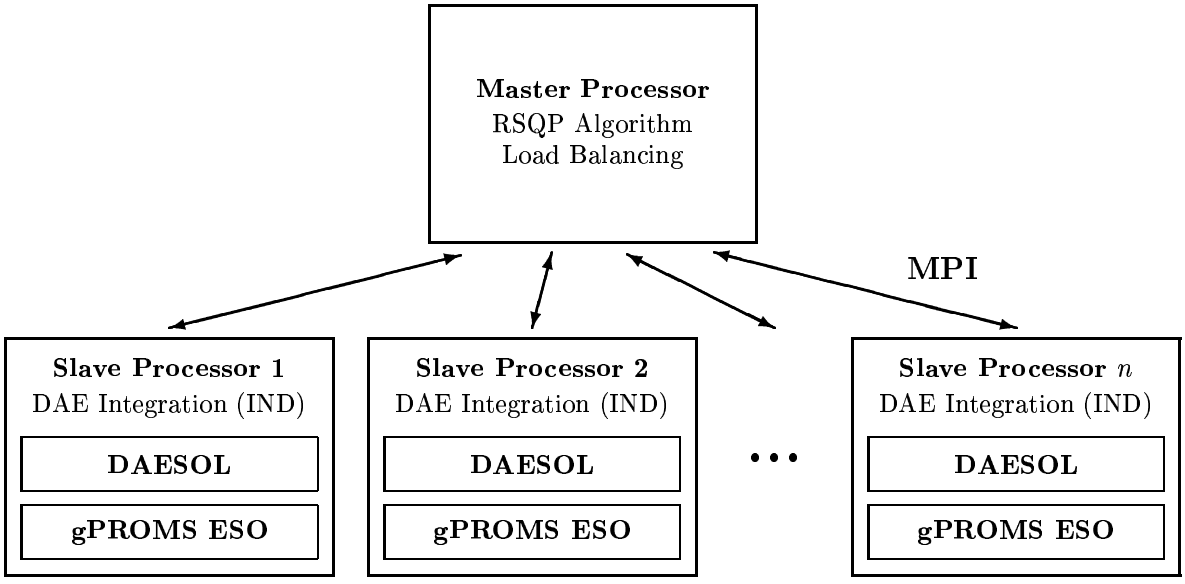

Figure 4: Parallelization Concept (gPROMS/MUSCOD-II).

Table 1: Problem Sizes and Solution Characteristics for Application Examples.

	problem sizes			
	`twobat`	`reactor_opt`	`bdtop10`	`bdtop50`
diff./alg. states	3/0, 4/0	6/14	10/112	50/560
controls	1, 0	1	1	5
parameters	3	0	4	4
discretization points	8	21	13	13
NLP variables	39	442	1619	8079
equality constraints	30	421	1599	7945
inequality constraints	80	905	3242	16178

	solution characteristics			
	`twobat`	`reactor_opt`	`bdtop10`	`bdtop50`
KKT tolerance	10^{-5}	10^{-5}	10^{-5}	10^{-5}
rel. integ. tolerance	10^{-7}	10^{-7}	10^{-7}	10^{-7}
SQP iterations	31	7	16	33
residual evaluations	53400	21700	58900	120000
Jacobian evaluations	11000	5470	12700	25900
objective value	25.591	7647.2	3.9524	3.9524
norm of Lag. gradient	$7.15 \cdot 10^{-3}$	$1.90 \cdot 10^{-2}$	$1.80 \cdot 10^{-2}$	$1.52 \cdot 10^{-2}$
norm of infeasibilities	$6.49 \cdot 10^{-5}$	$1.12 \cdot 10^{-6}$	$3.03 \cdot 10^{-6}$	$6.91 \cdot 10^{-6}$

gPROMS ESO model instance. The communication between master and slaves is based on the message passing interface (MPI) standard (Message Passing Interface Forum, 1997 [12]), thus it is portable to virtually any existing parallel computing system.

3 Numerical Results

3.1 Overview of Application Examples

Numerical results are presented for the following set of reaction and distillation problems (the detailed model equations and problem specifications can be found in the references cited):

- two-stage batch reactor problem `twobat` (Vassiliadis *et al.*, 1994 [16]; Leineweber, 1999 [10])
- batch reactor problem `reactor_opt` (gPROMS Manual, 2000 [8])
- batch distillation problem `bdtop10` (Farhat *et al.*, 1990 [7]; Leineweber, 1999 [10])
- enlarged batch distillation problem `bdtop50` (this paper)

As shown in the upper part of Table 1, the state dimensions of the DAE models range from less than 10 to more than 600, and the problems involve between one and five control functions and up to four time-invariant parameters. Control parametrization and multiple shooting discretization lead to NLP problems with up to several thousand variables, but relatively few degrees of freedom (compare the numbers of NLP variables and equality constraints). The large numbers of inequality constraints are due to the specification of lower and upper bounds for all NLP variables; additional inequalities may come from path constraints or end point constraints.

Problem `twobat` comprises two coupled batch reaction stages which are described by separate DAE models of different state and control dimensions, thus allowing to demonstrate the unique capability of MUSCOD-II to treat general multistage formulations. In contrast to Vassiliadis *et al.* (1994 [16]),

Table 2: Directional Sensitivities vs. Complete Linearization (Fortran/ADIFOR).

type of strategy	CPU times (sec)		
	`reactor_opt`	`bdtop10`	`bdtop50`
directional sensitivities	24.1 (100%)	140 (100%)	1894 (100%)
complete linearization	25.3 (105%)	262 (187%)	9150 (483%)

Table 3: Detailed Breakdown of Computational Cost for Problem `bdtop10` (Fortran/ADIFOR).

type of strategy	CPU times (sec)			
	IND	SI	OPT	total
directional sensitivities	112.3 (100%)	24.7 (100%)	2.7 (100%)	139.7 (100%)
complete linearization	234.8 (209%)	24.7 (100%)	2.9 (107%)	262.4 (188%)

no dummy variables and equations have to be introduced, and the two DAE model stages can be kept completely separate. The one-stage batch reactor example `reactor_opt` with fixed end time has been included as a well-known reference problem. Problems `bdtop10` and `bdtop50` involve time-optimal control of a batch distillation process for separating a ten-component mixture into two product cuts and two slop cuts. The enlarged problem `bdtop50` was generated from `bdtop10` by considering the operation of *five* identical batch distillation columns in parallel, thereby multiplying the state and control dimensions by a factor of five.

All DAE models have been implemented in gPROMS as well as in Fortran for direct comparison of computational performance. In case of the Fortran models, the sparse Jacobians were generated with the automatic differentiation software ADIFOR 2.0/SparsLinC (Bischof *et al.*, 1995 [3]). It is important to note that the gPROMS and the Fortran/ADIFOR implementations were numerically equivalent, i.e., they produced identical solutions with the same numbers of residual and Jacobian evaluations. Therefore, the solution characteristics given in the lower part of Table 1 apply to both model implementations. Only moderate numbers of SQP iterations were required to solve the problems to high accuracy.

The sequential calculations were performed on an SGI Octane workstation (MIPS R12000 CPU with 300 MHz speed, MIPS R12010 FPU) running IRIX64 Release 6.5; the parallel results were obtained on an SGI Origin 2000 16-processor machine (MIPS R10000 CPUs with 250 MHz speed, MIPS R10010 FPUs) running the same operating system.

3.2 Directional Sensitivities

Two alternative solution strategies have been compared within the modular framwork of MUSCOD-II: our new PRSQP strategy with directional sensitivities, and a "classical" PRSQP variant with complete constraint linearization. Recall from Part I that in the former case only $n^x + n^{\hat{q}} + 1$ directional sensitivities are generated, while in the latter case all $n^x + n^z + n^{\hat{q}}$ differential state sensitivities must be computed (here, n^x and n^z denote the numbers of differential and algebraic states, respectively, and $n^{\hat{q}}$ denotes the number of control parameters and localized global parameters).

The corresponding CPU times for problems `reactor_opt`, `bdtop10`, and `bdtop50` (Fortran/ADIFOR) are shown in Table 2. While there is not much difference between the two strategies for the small problem `reactor_opt` ($n^z = 14$), the PRSQP strategy with directional derivatives is faster than the complete linearization variant by factors of almost *two* and *five* for the larger problems `bdtop10` ($n^z = 112$) and `bdtop50` ($n^z = 560$), respectively. Hence for large problems with many algebraic variables, significant savings of computational cost are achieved.

Table 3 provides a detailed breakdown of computational cost for problem `bdtop10`: the CPU times are reported separately for the combined state and sensitivity integrations (IND), the additional state

Table 4: Comparison of Fortran and ESO Performance.

model variant	CPU times (sec)			
	`twobat`	`reactor_opt`	`bdtop10`	`bdtop50`
Fortran/ADIFOR	24.3 (100%)	24.1 (100%)	140 (100%)	1894 (100%)
in-process ESO	33.9 (140%)	28.9 (120%)	234 (167%)	3692 (195%)
out-of-process ESO	122 (502%)	113 (469%)	474 (339%)	4351 (230%)

integrations (SI), and the optimization calculations (OPT). It should be noted that OPT comprises all the work done outside the integrator, i.e., the steps of basis construction and projection, condensing, QP solution, and Hessian approximation. Hence it is remarkable that these computations account for only a very small fraction of the total computing time (typically, less than 1–2%) although large NLP problems are involved. By far most of the computing time is spent calculating the sensitivities which are required for setting up the reduced QP problem. As can be seen in Table 3, this computational burden is significantly reduced by using directional sensitivities. The additional state integrations which are needed for the line search also account for a considerable fraction of the total CPU time (the corresponding computing times of the two PRSQP strategies are, of course, identical). Finally, it can be observed that the CPU time for the optimization calculations is slightly increased in case of the complete linearization variant due to the additional projection step which has to be performed for the differential state sensitivities.

3.3 ESO Performance

In Table 4, we compare classical Fortran/ADIFOR models, gPROMS in-process ESOs, and gPROMS out-of-process ESOs in terms of computational performance. Although Fortran models still provide the best performance throughout, only a moderate loss of performance is found for in-process ESOs. Their relative performance somewhat degrades with increasing problem size – for smaller problems, they are actually competitive with Fortran models. However, even a doubling of computing times compared to Fortran as in case of the largest problem `bdtop50` seems acceptable from a practical point of view.

Out-of-process ESOs, on the other hand, are significantly slower than Fortran models and in-process ESOs, in particular for smaller problems. Since the relative performance of out-of-process ESOs *improves* with increasing problem size, the difference between the two ESO variants becomes much less pronounced for larger problems like `bdtop10`. This effect is due to the relatively large fixed communication overhead which is involved in each call of an out-of-process ESO method.

Problem `twobat` represents an interesting special case for the in-process ESO, because here the two gPROMS model stages have to be *reloaded* alternately on each SQP iteration. (Unfortunately, the current version of gPROMS cannot handle multiple ESO instances.) The related additional overhead leads to a relatively poor performance on this very small problem. For the out-of-process ESO, the model reloading is not required.

3.4 Parallelization

Computational statistics for the parallel solution of problem `bdtop50` (gPROMS in-process ESO) with MUSCOD-II are given in Table 5. One master processor and up to eight slave processors were employed. For each processor configuration, the average elapsed time (wallclock time) was determined based on 30 measurements to give an impression of the speed-up which can be obtained under practical conditions. (Note that unlike the workstation used before, the parallel machine was not free of other load.) The speed-up and efficiency values shown in Table 5 are defined as follows,

$$\text{speed-up for } p \text{ slaves} = \frac{\text{average elapsed time with 1 slave}}{\text{average elapsed time with } p \text{ slaves}}$$

$$\text{efficiency for } p \text{ slaves} = \frac{\text{speed-up for } p \text{ slaves}}{p} \cdot 100\%.$$

Table 5: Parallel Solution of Problem `bdtop50` (in-process ESO).

# of slaves	average elapsed time	speed-up	efficiency
1	5760 sec	1.00	100%
2	3130 sec	1.84	92%
4	2010 sec	2.87	72%
8	1520 sec	3.79	47%

It can be seen that the computing time was considerably reduced by using more processors. For instance, with four slaves, the problem was solved in only about one third of the time required with a single slave. Going from four to eight slaves, however, did not lead to much further improvement because the problem had only 13 multiple shooting intervals – not enough to keep all slaves busy, i.e., the efficiency was drastically reduced. In order to enable adequate load balancing, the number of intervals should be at least three times higher than the number of slaves.

4 Summary and Conclusions

The multiple shooting based dynamic optimization package MUSCOD-II has been coupled with the process modeling software gPROMS, creating a flexible and powerful optimization platform for general integrated batch processes. Each process stage (e.g., reactor, column) is separately modeled in gPROMS, and the multistage dynamic optimization problem is assembled and solved with MUSCOD-II. The code has also been parallelized based on the portable MPI standard. Much potential is expected from the optimization of complex integrated batch operations in the chemical process industries, since a rigorous optimization of this kind of processes has not been practical before. Therefore, it is anticipated that future generations of process modeling software will directly support such multistage formulations.

We have shown that the use of directional sensitivities becomes very important for larger problems with many algebraic variables, leading to drastically reduced computing times compared to a classical PRSQP strategy with complete constraint linearization. For the largest batch distillation problem presented, the new directional PRSQP strategy was about five times faster than the classical PRSQP variant. Furthermore, we have compared the performance of two types of gPROMS ESOs with that of Fortran/ADIFOR model implementations. Only a moderate loss of performance was found for in-process ESOs, especially in case of smaller problems. From a practical point of view, the advantage of using a powerful modeling environment like gPROMS – i.e., the increased productivity of the modeler – by far outweighs the disadvantage of the larger computing times incurred. Finally, we have demonstrated that parallelization of the state and sensitivity integrations provides an option to significantly speed up the numerical solution process if computing time *does* become an issue for large and complex problems.

Acknowledgements. The authors wish to thank Christian Schulz (Process Systems Enterprise Ltd., London) for his assistance with the gPROMS ESO interface and Amit Sharma (Interdisciplinary Center for Scientific Computing, University of Heidelberg) for creating a portable parallel code version of MUSCOD-II based on the MPI standard.

References

[1] Barton, P. I., and C. C. Pantelides, Modelling of combined discrete/continuous processes. *AIChE J.* **40**, 966–979 (1994).

[2] Bauer, I., Numerische Verfahren zur Lösung von Anfangswertaufgaben und zur Generierung von ersten und zweiten Ableitungen mit Anwendungen bei Optimierungsverfahren in Chemie und Verfahrenstechnik. *Ph.D. thesis*, University of Heidelberg (2000).

[3] Bischof, C., A. Carle, P. Khademi, A. Mauer, and P. Hovland, ADIFOR 2.0 user's guide (revision C). *Technical Memorandum No. 192*, Mathematics and Computer Science Division, Argonne National Laboratory (1995).

[4] Braunschweig, B. L., C. C. Pantelides, H. I. Britt, S. Sama, Process modeling: the promise of open software architectures. *Chemical Engineering Progress* **96(9)**, 65–76 (2000).

[5] Diehl, M., Real-Time Optimization for Large Scale Nonlinear Processes. *Ph.D. thesis*, University of Heidelberg [Download: http://www.ub.uni-heidelberg.de/archiv/1659/] (2001).

[6] Diehl, M., H. G. Bock, J. P. Schlöder, R. Findeisen, Z. Nagy, and F. Allgöwer, Real-time optimization and nonlinear model predictive control of processes governed by large-scale DAE. *J. Proc. Contr.* **(in print)** (2002).

[7] Farhat, S., M. Czernicki, L. Pibouleau, and S. Domenech, Optimization of multiple-fraction batch distillation by nonlinear programming. *AIChE J.* **36**, 1349–1360 (1990).

[8] *gPROMS Advanced User's Manual.* Process Systems Enterprise Ltd., London (2000).

[9] Graham, R. L., Bounds on multiprocessing timing anomalies. *SIAM J. Appl. Math.* **17**, 416–429 (1969).

[10] Leineweber, D. B., Efficient Reduced SQP Methods for the Optimization of Chemical Processes Described by Large Sparse DAE Models. *Fortschritt-Berichte VDI*, Reihe 3, Nr. 613 (ISBN 3-18-361303-4). VDI Verlag GmbH, Düsseldorf (1999).

[11] Leineweber, D. B., I. Bauer, H. G. Bock, and J. P. Schlöder, An efficient multiple shooting based reduced SQP strategy for large-scale dynamic process optimization – part I: theoretical aspects. (2002).

[12] Message Passing Interface Forum, *MPI-2: Extensions to the Message-Passing Interface*. University of Tennessee, Knoxville (1997).

[13] Nagy, Z., R. Findeisen, M. Diehl, F. Allgöwer, H. G. Bock, S. Agachi, J. P. Schlöder, and D. B. Leineweber, Real-time feasibility of nonlinear model predictive control for large scale processes – a case study. *Proceedings of the ACC 2000*, Chicago (2000).

[14] Object Management Group, *The Common Object Request Broker: Architecture and Specification.* Needham, MA (1997).

[15] Pantelides, C. C., SPEEDUP – recent advances in process simulation. *Comput. Chem. Engng* **12**, 745–755 (1988).

[16] Vassiliadis, V. S., R. W. H. Sargent, and C. C. Pantelides, Solution of a class of multistage dynamic optimization problems. 1. Problems without path constraints. *Ind. Eng. Chem. Res.* **33**, 2111–2122 (1994).