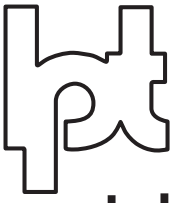


Lehrstuhl für Prozeßtechnik
Prof. Dr.-Ing. W. Marquardt
RWTH Aachen

Technical Report LPT–2000–25

Component–based Implementation of a Dynamic Optimization Algorithm Using Adaptive Parameterization

M. Schlegel, Th. Binder, A. Cruse, J. Oldenburg, W. Marquardt

November 2000

Contribution to:
ESCAPE–11, Kolding, Denmark, 27–30.5.2001.

Published in:
R. Gani, S.B. Jørgensen (Eds.): “European Symposium on Computer Aided Process Engineering – 11”, Elsevier, 2001, 1071–1076.

Enquiries should be addressed to:

Lehrstuhl für Prozeßtechnik
RWTH Aachen
Templergraben 55
D–52056 Aachen

Tel.: ++49 / 241 / 80 46 68
Fax: ++49 / 241 / 88 88 326
E–Mail: secretary@lfpt.rwth–aachen.de

RWTH

Component–based implementation of a dynamic optimization algorithm using adaptive parameterization *

M. Schlegel, T. Binder, A. Cruse, J. Oldenburg, W. Marquardt

Lehrstuhl für Prozesstechnik, RWTH Aachen, Germany.

In this work we present a component software technique applied to a dynamic optimization algorithm based on the sequential approach. The implementation of the algorithm allows the optimization of existing models formulated in recent modeling environments without the need of model transfer or recoding. The numerical algorithm is capable of generating problem–dependent, non–uniform discretization grids which might differ for each control variable. The software is used to solve an example problem of industrial relevant size. Based on the experience drawn from the example benefits and drawbacks of this technology are discussed.

Keywords: dynamic optimization, large–scale systems, direct method, adaptive mesh refinement, sequential approach, component technology, CAPE–OPEN

1. Introduction

Dynamic optimization nowadays is used for various applications in process design and operations. Examples include the design of trajectories for the optimal operation of batch and semi–batch reactors or for continuous processes during transient phases such as grade transitions, start–up or shut–down. In general, dynamic optimization algorithms require a mathematical model of the process considered. For industrial processes, the development, validation and maintenance of — usually large–scale — process models often require major financial expenses as well as a substantial amount of engineering experience. For these reasons, the reuse of existing models or parts of them is highly desirable. However, a large variety of proprietary software tools for computer–aided process engineering, especially modeling tools, are used in industry today, whereas model–based numerical algorithms usually require the model information in some specific format, which typically is not compliant to the modeling tools. A transfer of model information between such applications or the conversion to a special format can be cumbersome and error–prone. In certain cases it might be even impossible due to bottlenecks in interoperability and reusability. To overcome such problems, the EC–funded CAPE–OPEN project (Braunschweig *et al.*, 2000) aims at an open standard system of interfaces for information exchange between software tools in process systems engineering. Communication between heterogeneous applications is foreseen to be done by using interoperability standards, e.g. CORBA (Henning and Vinoski, 1999). Following these ideas, ideally there would be no need for any superfluous model transformation. In order to benefit from the concepts developed in this project,

*This work has been partially funded by the European Commission under grant G1RD–CT–1999–00146 in the "INCOOP" project.

obviously there is the need for application software, which implements those interface definitions for practical use.

In this contribution we consider the application of these ideas to the area of dynamic optimization. Applied to industrial processes, dynamic optimization commonly leads to large–scale and strongly nonlinear differential–algebraic problems containing path and endpoint constraints. The solution of such problems is still a computationally demanding task. Efficient solution algorithms are important for real–time applications, such as optimization–based monitoring and control on receding horizons. Many algorithms for dynamic optimization have been developed based on the direct methods, which convert the continuous problem into a finite–dimensional nonlinear programming problem (NLP) (e.g. Vassiliadis *et al.* (1994), Bock *et al.* (2000), Cervantes and Biegler (1998)). In particular within the sequential or single–shooting approach, this NLP is obtained by parameterization of the control variables only. Accuracy, efficiency, and robustness of the solution of dynamic optimization problems strongly depend on the chosen discretization grids (e.g. Binder *et al.* (2000), Betts and Huffmann (1998)).

An implementation, which combines a dynamic optimization algorithm using a sequential approach and adaptive discretization grid refinement with innovative software technology is presented in this paper. Experiences with the application to a large–scale example problem are shown and benefits and drawbacks are discussed.

2. Problem formulation

We focus on the use of dynamic optimization for optimal trajectory generation and consider an optimal control problem formulation of the following form:

$$\min_{\boldsymbol{u},\boldsymbol{p},t_f} \; \Phi := \Phi_0\left(\boldsymbol{x}\left(t_f\right)\right) + \int_{t_0}^{t_f} f_0\left(\boldsymbol{x},\boldsymbol{u},\boldsymbol{p}\right) d\tau \tag{P1}$$

$$\text{s.t.} \qquad \boldsymbol{M}\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x},\boldsymbol{u},\boldsymbol{p},t)\,, \quad t \in [t_0,t_f]\,, \tag{1}$$

$$0 = \boldsymbol{x}(t_0) - \boldsymbol{x}_0\,, \tag{2}$$

$$0 \geq \boldsymbol{g}(\boldsymbol{x},\boldsymbol{u},\boldsymbol{p},t), \quad t \in [t_0,t_f]\,, \tag{3}$$

$$0 \geq \boldsymbol{e}\left(\boldsymbol{x}\left(t_f\right)\right)\,. \tag{4}$$

In this formulation, $\boldsymbol{x}(t) \in \mathbb{R}^{n_x}$ denote the state variables, which can be either of differential or algebraic type, whereas $\boldsymbol{x}_0$ are initial conditions. The variables to be determined by the optimization procedure for minimization of the objective function Φ are the control vector $\boldsymbol{u}(t) \in \mathbb{R}^{n_u}$, the unknown time–independent parameters $\boldsymbol{p} \in \mathbb{R}^{n_p}$, as well as the final time t_f. The differential–algebraic (DAE) model is given by the equation system (1), (2). We only consider DAE systems with an index of less than or equal to one. Furthermore, path constraints (3) can be applied on the states, control variables and time–independent parameters. Finally, endpoint constraints (4) on the state variables can be employed.

3. Single–shooting solution approach

In the sequential approach (Vassiliadis *et al.*, 1994) the control profiles $u_i(t), i = 1, ..., n_u$ have to be approximated, and often piecewise polynomial expansions of the form

$$u_i(t) \approx \overline{u}_i(\boldsymbol{c}_i, t) = \sum_{k \in \Lambda_i} c_{i,k} \phi_{i,k}(t), \tag{5}$$

are used, where Λ_i denotes the index set of the chosen parameterization functions $\phi_{i,k}(t)$ and the vector $\boldsymbol{c}_i$ contains the corresponding parameter vector. For brevity, in this paper we only consider piecewise constant functions $\phi_{i,k}(t) := 1 \quad \forall\, t_k \leq t \leq t_{k+1}$, otherwise $\phi_{i,k}(t) = 0$, though an extension to higher–order polynomials is possible. The grid points for each u_i are contained in the mesh $\Delta_{\Lambda_i} := \{t_k | k \in \Lambda_i\}$.

By discretization of the control variables, the dynamic optimization problem (P1) can be reformulated into the following NLP:

$$\min_{\boldsymbol{c},\boldsymbol{p},t_f} \Phi = \Phi_0\left(\boldsymbol{x}\left(\boldsymbol{c},\boldsymbol{p},t_f\right)\right) + \int_{t_0}^{t_f} f_0\left(\boldsymbol{x}\left(\boldsymbol{c},\boldsymbol{p},\tau\right), \overline{\boldsymbol{u}}(\boldsymbol{c},\tau), \boldsymbol{p}\right) d\tau \tag{P2}$$

$$\text{s.t.} \qquad 0 \geq \boldsymbol{g}(\boldsymbol{x}, \boldsymbol{c}, \boldsymbol{p}, t_i), \quad \forall t_i \in \Delta_\Lambda\,, \tag{6}$$

$$0 \geq \boldsymbol{e}\left(\boldsymbol{x}\left(t_f\right)\right)\,. \tag{7}$$

The path constraints (6) are now evaluated on the unified mesh of all control variables $\Delta_\Lambda := \bigcup_{i=1}^{n_u} \Delta_{\Lambda_i}$. The DAE model (1), (2) is not present directly in the NLP problem, rather it is solved by numerical integration in each function evaluation step of the NLP solver to determine $\boldsymbol{x}(\boldsymbol{c}, \boldsymbol{p}, t)$ for given $\boldsymbol{c}$ and $\boldsymbol{p}$ and therefore present implicitly. Algorithms for the solution of this NLP, typically SQP methods, require gradient information of the constraints and the objective function with respect to the decision variables. There are several possibilities to obtain these gradients. Here, we consider the explicit solution of the arising sensitivity equation systems, which is the method of choice in most optimization algorithms (e.g. Vassiliadis *et al.* (1994)).

The sensitivity systems can be solved by numerical integration together with the original DAE system. Although there are efficient algorithms available, which exploit the special properties of the sensitivity system (e.g. Feehery *et al.* (1997)), still the most significant computational effort is spent on the sensitivity analysis. Since the influence of a decision variable $c_{i,k}$ on the states $\boldsymbol{x}$ is limited to the time region $t \geq t_k$, it is sufficient to solve each sensitivity equation system for the determination of $\boldsymbol{s}_i := \frac{\partial \boldsymbol{x}}{\partial c_{i,k}}$ on the time interval $[t_k, t_f]$. Still, this leads to a computational effort increasing polynomially with the number of decision variables. Hence, it is clearly desirable to keep the number of decision variables as small as possible, without losing much accuracy. This raises the question of an optimal selection of the discretization grids. The formulation (5) offers the choice of separate, non–uniform parameterization grids for each control variable. This fact can be exploited by using adaptive refinement strategies in order to generate efficient, problem–adapted meshes Δ_{Λ_i}, as explained in the following section.

4. Adaptive refinement algorithm

Problem–adapted possibly non–uniform grids for an efficient approximation of u_i are generated by successive refinement of an initial coarse discretization mesh $\boldsymbol{u}^0(\Lambda^0)$. In each refinement step ℓ the previous solution $u^{\ell-1}$ is inspected by a wavelet–analysis (Binder *et al.*, 2000). Based on this analysis the discretization is refined locally in areas where $u^{\ell-1}$ reveals large variations. (P2) is then resolved on the improved discretization grid Λ^ℓ

where the interpolated old solution $u^{\ell-1}$ is used as an initial guess. Hence, (P2) is resolved repeatedly on different meshes with an increasing number of parameterization variables. Consequently all the quantities in (5), (P2), (6), (7) should have the refinement counter ℓ as superscript, but this has been omitted for ease of notation. The refinement is carried out for each control u_i such that individual discretization grids for each u_i are obtained.

5. Implementation

The numerical concepts presented above have been implemented into the prototype software tool *ADOPT*. Those parts of the program, which require access to the model information have been coded compliant to the so–called ESO interface definition, as defined in the CAPE–OPEN project (Keeping and Pantelides, 1998). This Equation Set Object (ESO) is an interface definition for communicating all information contained in the DAE model (1) which could be required by numerical algorithms, such as number and values of variables and residuals, structure and values of the model Jacobian matrix etc. Any modeling package, which allows access to numerical model information through the same interface could be used as a model server. To the authors' knowledge, this feature currently is only provided by gPROMS. The prototype is able to access gPROMS as the model server via a CORBA object bus (Henning and Vinoski, 1999), but also any legacy model wrapped with an ESO interface can be used, provided that there are no discontinuities present. Dynamic optimization following the methods presented above can be performed without the need to recode the model in a programming language.

The basic structure of the tool is depicted in Figure 1. The refinement loop can be recognized in the left–hand part of the picture. On the right, the way how the model information is transferred between the dynamic optimizer and the model server is shown. Two of the methods defined in the ESO standard, *SetVariables* and *GetResiduals*, exemplarily shown in Figure 1, indicate how the residual values for the current variable set can be obtained from the model server. The CORBA bus enables platform independence and interoperability between operating systems. A drawback of this technology is an overhead in time consumption caused by the communication. In the following section, this will be discussed in more detail.

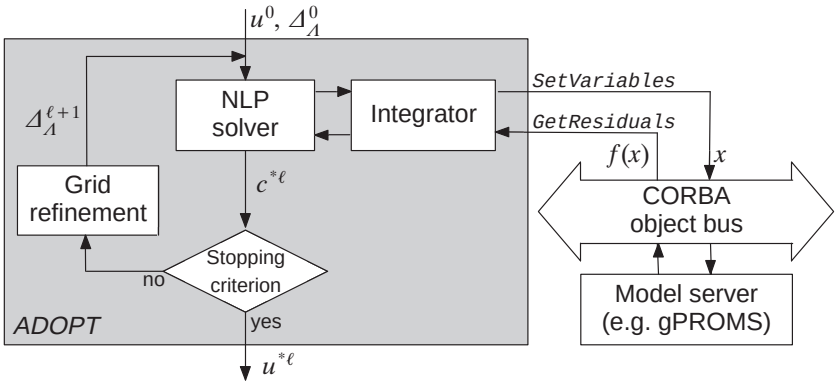
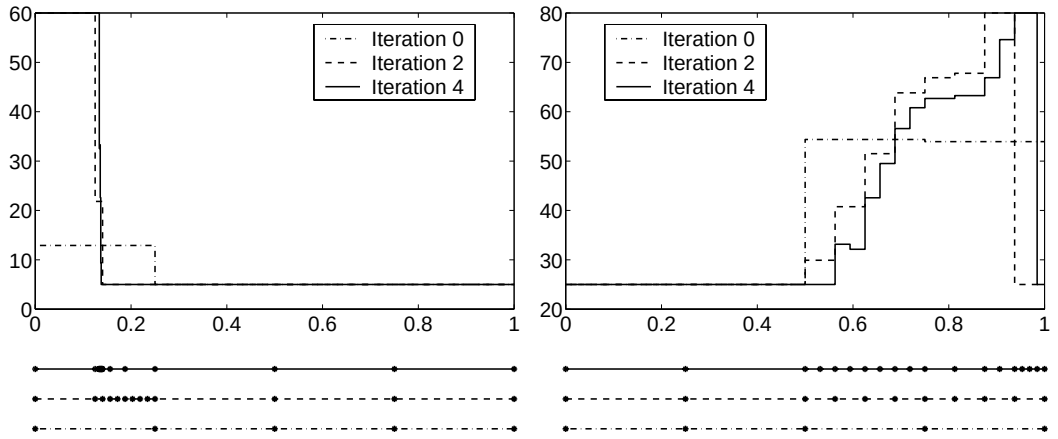


Figure 1: Basic structure of *ADOPT* and communication via ESO interface.



(a) Reflux ratio (b) Vapor stream

Figure 2: Control variable profiles and corresponding problem–adapted grids.

6. Case study

In this section we present some results obtained by applying this software to an example model. The model describes a batch reactive distillation column for the production of ethyl acetate (Cervantes and Biegler, 1998). We use a modified version with the simplifying assumption of constant molar overflow. The model is available in gPROMS and comprises reboiler, condenser and 10 trays and contains in this form 418 equations and variables, of which 63 are differential. The reflux ratio and the vapor stream leaving the reboiler have been chosen as control variables. The objective is to minimize the energy demand in the reboiler for a process running one hour, under the constraints, that the amount of final distillate should be at least 6.0 kmol, with a purity of the ester of at least 0.46. The optimization was initialized with constant profiles. Figure 2 shows optimal profiles and corresponding grids for the two control variables. For clarity only results from every second iteration in the adaptation loop are shown. The adaptation of the grids to the problem becomes apparent. Table 1 compares computational results from the different refinement iterations with those obtained by using a uniform mesh of comparable accuracy as the one in iteration 4. The objective value decreases, while the CPU time per iteration increases due to the growing number of decision variables caused by the refined grids. With the adaptive approach, results with comparable accuracy (e.g. after iteration 2) can be obtained within significantly smaller computation time as compared to a solution on a uniform mesh. Since each iteration produces a continuously refined intermediate feasible solution, this approach is particularly useful for real–time applications.

The additional time consumption, which is required by the CORBA bus, is proportional to the size of the vectors to be transfered and lies in the order of milliseconds. Obviously, the number of calls to the model server plays a crucial role in this context, as well. A typical optimization requires in the order of several hundred thousand calls via the CORBA bus. Therefore, the communication between the optimization software and the model server causes a significant overhead with the current implementation. For this reasons, the applicability to real–time problems is still limited. However, this problem can be overcome by modifications of the software architecture. One option is to rely

Table 1: Solutions on different adapted grids compared with uniform mesh.

Iteration ℓ	0	1	2	3	4	Uniform mesh
No. of dec. vars.	8	16	22	28	34	64
Objective value	62.322	61.655	61.575	61.553	61.542	61.564
CPU–sec per iter.	34.8	145.2	130.9	276.1	493.3	2474.3
CPU–sec accum.	34.8	180.0	310.9	587.0	1080.3	2474.3

on future developments and improvements in the CORBA technology. Alternatively, it is conceivable to connect the optimization software and the model server directly "in process" rather than using middleware components. The ESO interface still should be used for consistency. This approach might be the way to proceed for applications, where computation time is the major issue, especially in the real–time area.

7. Conclusions

The implementation of a dynamic optimization algorithm, which adaptively generates problem–dependent discretization grids for different control variables in order to increase the efficiency and robustness of the solution has been presented. As a new feature, the access of model information via a CAPE–OPEN interface has been introduced. The functionality of this approach has been proven by applying it to a large–scale problem. The heterogeneous implementation using CORBA as communication middleware appeared to be a practical approach, though there still exists a significant overhead in computation time solely caused by software–related reasons. Future improvements in this area will enable the use of such frameworks in industrial applications.

REFERENCES

Betts, J.T. and W.P. Huffmann (1998). Mesh refinement in direct transcription methods for optimal control. *Optim. Control Appl. Meth.* **19**, 1–21.

Binder, T., A. Cruse, C. Villas and W. Marquardt (2000). Dynamic optimization using a wavelet based adaptive control vector parameterization strategy. *Comp. Chem. Eng.* **24**, 1201–1207.

Bock, H.G., M.M. Diehl, D.B. Leineweber and J.P. Schlöder (2000). A direct multiple shooting method for real–time optimization of nonlinear DAE processes. In: *Nonlinear Model Predictive Control* (F. Allgöwer and A. Zheng, Eds.). pp. 246–267. Birkhäuser Verlag. Basel.

Braunschweig, B., C. Pantelides, H. Britt and S. Sama (2000). Process modeling: The promise of open software architectures. *Chem. Eng. Prog.* **96**(9), 65–76.

Cervantes, A. and L.T. Biegler (1998). Large–scale DAE optimization using a simultaneous NLP formulation. *AIChE Journal* **44**(5), 1038–1050.

Feehery, W., J. Tolsma and P.I. Barton (1997). Efficient sensitivity analysis of large–scale differential–algebraic systems. *Appl. Numer. Math.* **25**, 41–54.

Henning, M. and S. Vinoski (1999). *Advanced CORBA Programming with C++*. Addison–Wesley. Reading, MA.

Keeping, B. and C. Pantelides (1998). WP4.1 Numerical Solvers Open Interface Specification Draft. Technical Report Internal Draft 8. CPSE, Imperial College. London.

Vassiliadis, V.S., R.W.H. Sargent and C.C. Pantelides (1994). Solution of a class of multistage dynamic optimization problems: 2. Problems with path constraints. *Ind. Eng. Chem. Res.* **33**(9), 2123–2133.