European Symposium on Computer Aided Process Engineering – 15
L. Puigjaner and A. Espuña (Editors)

A Library for Equation System Processing based on the CAPE-OPEN ESO Interface

G. Schopfer[a], J. Wyes[a], W. Marquardt[a1], L. von Wedel[b]

[a] Lehrstuhl für Prozesstechnik, RWTH Aachen, D-52056 Aachen, Germany

[b] AixCAPE e.V., D-52072 Aachen, Germany

Abstract

In this contribution we present the LptNumerics library for processing of equation systems that are provided via the CAPE-OPEN ESO interfaces. The library provides functionality for the pre-processing of an equation system and its solution. Here the term pre-processing refers to the steps required prior to solving a model in an equation-oriented solution approach. The library is implemented in C++ and can be used within any equation-oriented application. This contribution discusses the design and implementation of the library and presents two application examples.

Keywords: CAPE-OPEN, process modelling, object-oriented design

1. Introduction

In the EU-funded project CAPE-OPEN and its successor project Global CAPE-OPEN standard interfaces for process modelling tools were defined. The goal of these projects and the follow up organization CO-LaN is to achieve interoperability of process modelling tools by means of standardized interfaces that make modules exchangeable across tools from different vendors (CO-LaN, 2004). These modules include unit operation models, thermodynamic calculation packages and solvers.

Within the "numerics" work package of CAPE-OPEN, interfaces were developed for simulation and optimisation of models that are defined in an equation-oriented approach. In this approach, models are typically defined in some modelling language in terms of *equations* describing the process behaviour and *variables* denoting process quantities (von Wedel et al., 2002). In case of a process consisting of several process units the equations of all process units are aggregated to one system and solved by a numerical equation solver. In case of recycles in the process model, the equation system has to be solved iteratively. For example for an algebraic equation system, in each iteration step the equation solver evaluates the differences of left- and right-hand sides of each equation for a given set of variable values. Based on these equation residuals and their derivatives with respect to the variables, called Jacobian matrix, the equation solver determines the variable values for the next iteration step. A solution of an equation system is obtained when all residuals are (close to) zero.

[1] Correspondence should be addressed to W. Marquardt, marquardt@lpt.rwth-aachen.de

In the CAPE-OPEN project, the representation of an equation system in the form suitable for evaluation by an equation solver is called "Equation Set Object" (ESO). The CAPE-OPEN standard distinguishes between ESO that represent only algebraic equations, defined by the interface *ICapeNumericAlgebraicESO*, and ESO that evaluate additionally differential equations, defined by the interface *ICapeNumericDifferential-AlgebraicESO*. In the remainder of this paper, we refer to both interfaces as the CAPE-OPEN ESO interfaces. These interfaces allow to set variable values and to evaluate the corresponding residuals and the jacobian matrix. In case of a differential-algebraic ESO, an independent quantity (e.g. time) can be set additionally and the derivatives of the differential variables with respect to it are considered as additional unknowns of the system. The work of the numerics work package resulted in a draft interface specification that is implemented by the commercial process modelling tool gPROMS (Process Systems Enterprise, 2002) and the final interface specification that was implemented prototypically by Belaud and co-workers (Belaud et al., 2001). However, CAPE-OPEN only deals with the definition of interface standards and does not provide any kind of functionality. Especially in the area of equation-oriented modelling, there is a large set of functionality which is required by many applications. Hence, reusable software components for existing process systems would be of great help to the modelling engineer.

In this contribution, we present the LptNumerics library for the processing of equation systems based on the CAPE-OPEN ESO interfaces with the goal of exploiting this opportunity of reuse. The LptNumerics library provides reusable functionality for pre-processing of equation systems and their solution in equation-oriented modelling tools. Here the term pre-processing refers to several steps required prior to solving a process model, e.g. the aggregation of the equations of subordinate process unit models and the specification of input values. The structure of this paper is as follows. Section 2 describes the conceptual architecture of the LptNumerics library and discusses its benefits. Section 3 describes the implementation of the library in terms of actual classes. Section 4 presents applications of the library. Section 5 concludes the paper with a summary and an outlook on future work.

2. Conceptual architecture of the LptNumerics library

The conceptual architecture of the LptNumerics library can be described by a layered model as shown in Figure 1. On the *Tool Layer* we consider equation-oriented process modelling tools such as gPROMS, Aspen Custom Modeler (AspenTech, 2004) or UnitGen (a research prototype from our lab to generate a CAPE-OPEN ESO from a Modelica specification) (Geffers et al., 2001). gPROMS and UnitGen provide their models in terms of the CAPE-OPEN ESO interfaces, whereas the current version of Aspen Custom Modeler does not provide the CAPE-OPEN ESO interfaces to external applications. However, since the equation-oriented approach of Aspen Custom Modeler allows the realisation of the CAPE-OPEN interfaces at least in principle and a future version might actually offer them, it is considered here as an additional example. On the *Tool Level* it has to be considered that different technologies can be used for the implementation of the CAPE-OPEN interfaces. For example gPROMS and UnitGen use the CORBA distributed object technology. On the other hand, since Aspen Custom

Modeler generally makes use of the COM middleware, we assume that it eventually will provide a CAPE-OPEN ESO via COM interfaces.

In the *Wrapper Layer* the technically different CAPE-OPEN ESO interfaces from the tools are wrapped to one C++ interface, called *LptESO*. For the integration of gPROMS and UnitGen, this layer contains a C*ORBA Wrapper* that establishes a bridge between the CORBA CAPE-OPEN ESO interfaces and the LptESO interface. For Aspen Custom Modeler a COM wrapper would be required for establishing a bridge between the COM CAPE-OPEN ESO interfaces and the LptESO interface. In the following we refer to classes that realise the LptESO interface as LptESO and classes that realise the CAPE-OPEN ESO interface as CAPE-OPEN ESO for short.

Based on these wrappers, the modules of the *Pre-processing Layer* are implemented. These modules realize functionality that might be required for pre-processing of process models before they can be solved. In case that not all derivatives of a CAPE-OPEN ESO are provided, the *Perturbation Module* provides methods that add missing derivatives through perturbation strategies. When the different process unit models of a process model are realized in different CAPE-OPEN ESOs, the *Aggregation Module* is used. It provides methods that aggregate several ESOs to a single overall one and allow to enforce identity constraints between variables of different ESOs, e.g. to represent connections between two process unit models. Finally, the *Specification Module* is concerned with imposing constraints on individual variables, such as the assignment of model inputs for a steady-state or a dynamic simulation.

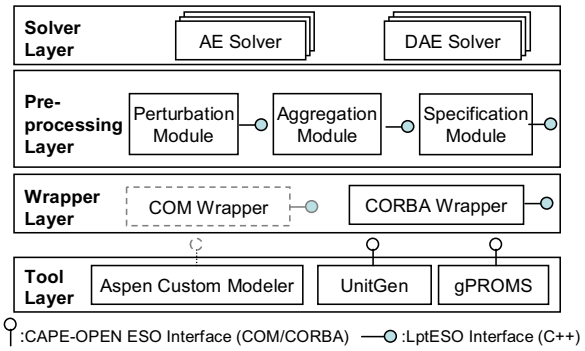

Figure 1: Conceptual Architecture of the LptNumerics Library

The modelling engineer controls and designates the pre-processing steps, which are executed from LptNumerics. After that, the fully specified ESO does not reveal any remaining degrees of freedom. It can hence be solved by one of the algebraic equation (AE) solvers or differential-algebraic equation (DAE) solvers of the *Solver Layer*.

The introduction of a wrapper layer between the CAPE-OPEN interfaces of the tools and the pre-processing layer has several advantages. Firstly, it decouples the technically different realizations of the CAPE-OPEN interfaces from the pre-processing layer. Furthermore, in reality different tools realize even semantically slightly different versions of the CAPE-OPEN ESO interface. For example, gPROMS in its current version implements a draft standard of the CAPE-OPEN ESO whereas (Belaud et al.,

2001) implement the final specification of the standard. By implementing the pre-processing layer based on the LptESO interface, adaptations of the tool interfaces and the integration of new tools only affects the corresponding class of the wrapper layer, whereas the pre-processing layer remains unaffected. Another advantage of mapping the COM and CORBA interfaces to a C++ interface is the reduction of the complexity of the pre-processing classes and the improvement of their runtime performance. This is due to the higher complexity of COM and CORBA implementations compared to pure C++ implementations and the overhead of communicating through a middleware compared to in-process communication.

Note that all elements of the pre-processing layer implement the LptESO interface and interact with each other only via this interface. Furthermore, this interface is used by the solvers of the solver layer to access models. For this reason, the elements of the pre-processing layer can be combined flexibly as required in a specific application.

3. Implementation of the LptNumerics library

Figure 2 gives an overview on the implementation of the LptNumerics library that consists of three parts: the *interface classes*, the *wrapper classes*, the *pre-processing classes* and the *solver classes*. The interface classes comprise the model interface class *LptESO* and the solver interface classes *LptAESolver* and *LptDAESolver* for AE and DAE solvers, respectively. The interface classes are realized as pure virtual C++ classes. By defining inheritance relations between the interface classes and the processing, wrapper and solver classes it is enforced that these classes realize the corresponding interface.

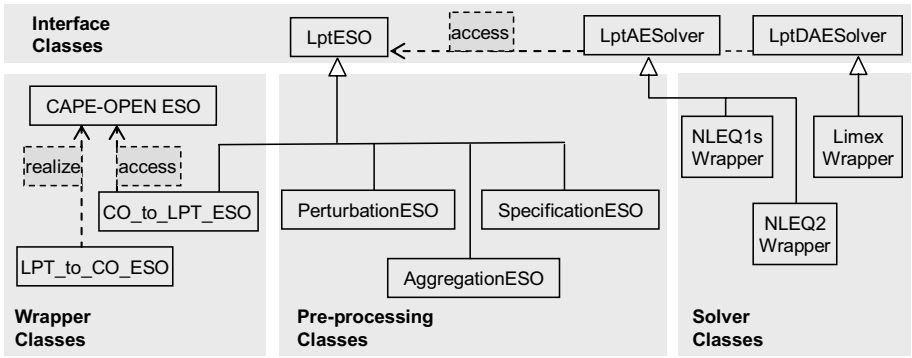

Figure 2: Overview of the LptNumerics implementation (UML)

The wrapper classes comprise the classes *CO_to_Lpt_ESO* and *Lpt_to_CO_ESO*. The class *CO_to_Lpt_ESO* accesses a CAPE-OPEN ESO and provides it to the processing classes via the LptESO interface. The class *Lpt_to_CO_ESO* realizes the CAPE-OPEN ESO interface and allows providing a LptESO as a CAPE-OPEN ESO. It is used for example by UnitGen, which first generates an LptESO and then provides it linked with the wrapper class as a CAPE-OPEN ESO. Both wrapper classes use the CORBA framework, i.e. the class *CO_to_Lpt_ESO* is implemented as a CORBA client and the class *Lpt_to_CO_ESO* is implemented as a CORBA server.

The pre-processing classes comprise the classes *PerturbationESO*, *AggregationESO*, and *SpecificationESO* with the functionality of the corresponding modules discussed in Section 2. The class *PerturbationESO* evaluates missing elements of the Jacobian matrix by means of numerical perturbation and provides an LptESO with a complete Jacobian matrix. The class *AggregationESO* aggreagtes several LptESO to a single overall LptESO and adds further equations for the identity relations between variables of different LptESOs. Finally, the class SpecificationESO provides a mechanism for assigning variables either to constant values (for AEs) or time dependent functions (for DAEs) and provides an LptESO with additional equations for the assignments.

The solver classes serve as wrappers for integrating different numerical solvers into the library. In order to make all solvers accessible through unified interfaces, each solver is integrated by one wrapper class that inherits from the corresponding interface class (i.e. *LptAESolver* for AE solvers and *LptDAESolver* for DAE solvers). Furthermore, the wrapper classes access the equation system through the LptESO interface and pass the corresponding numerical information to the solver in the format it requires. Currently, the algebraic equation solvers NLEQ1s and NLEQ2 (Nowak and Weimann, 1991) and the DAE solver Limex (Deuflhard et al., 1987) are integrated into the library. For this purpose the solver classes *Nleq1sWrapper*, *Nleq2Wrapper* and *LimexWrapper* were implemented.

The LptNumerics library is implemented in C++ using the Microsoft Visual C++ compiler. For the implementation of CORBA interfaces the CORBA framework omniORB (omniORB, 2004) is used.

4. Applications

Two applications have been realized using *LptNumerics*. The first one is the program *UnitGen*, which generates solvable models from a model specification written in the Modelica language. From this model specification, *UnitGen* generates C++ code that provides the model as an LptESO. In a second step a *SpecificationESO* is created based on a configuration file that specifies which variables serve as model inputs and assigns their values. Then a solver of the *LptNumerics* library can be applied to solve the model. Alternatively, the LptESO can be combined with the wrapper class *Lpt_to_CO_ESO* and be exported as a CAPE-OPEN ESO.

The second application is *CHEOPS* (Schopfer et al., 2004), an integration platform for chemical process modelling with models that are provided by different modelling tools. Cheops creates an image from a flowsheet using nearly all LptNumerics classes of Figure 2. Here, the LptNumerics library is used for integrating process unit models provided as CAPE-OPEN ESO as LptESO (using the class *CO_to_Lpt_ESO*). For providing the model as a module that calculates the outputs given the inputs (as well referred to as closed-form representation), it is combined with the class *SpecificationESO*, which assigns the process units inputs, and a numerical solver. In case of missing derivatives of the jacobian matrix, the *PerturbationESO* is added during runtime. Furthermore, on the process model level, the LptESOs from different unit models can be aggregated (by using the class *AggregationESO*) to a single overall LptESO and solved subsequently.

5. Conclusions

In this contribution, we have presented the *LptNumerics* library that provides functionality which is applicable in different equation-oriented modelling and simulation tools. The functionality supports the integration of CAPE-OPEN ESO, their aggregation to one overall equation system, the specification of variable assignments and the solution of the resulting, square equation system. Wrappers are used to decouple the functionality of the library from the technically (and sometimes semantically) different realisations of the CAPE-OPEN interfaces. These wrappers map the CAPE-OPEN interfaces to a C++ interface, the LptESO interface. Based on the LptESO interface, a purely C++ based implementation realizes the pre-processing functionality of the library. The approach of implementing the pre-processing functionality based on a C++ interface instead of using directly the CAPE-OPEN COM or CORBA interfaces considerably reduces the complexity of the implementation and improves its run-time performance. Furthermore, it makes the library implementation independent of the different technologies used to implement the CAPE-OPEN ESO interfaces and makes it more robust towards changes in the interface implementations. In the architecture chosen, such changes only affect the wrapper implementations and new communication technologies (e.g. COM) can be realized simply by adding a new wrapper.

The architecture offers the possibility to extend the library with additional software modules at little effort. Currently work focuses on extending the library interfaces and implementation towards optimisation, including providing second order derivatives, as it is required by Newton-type optimisation algorithms.

References

AspenTech (2004): Product information available online at http://www.aspentech.com/

B. L. Braunschweig, H. Britt, C. C. Pantelides, and S. Sama (2000): Process modelling, the promise of open software architectures. Chemical Engineering Progress, pages 65–76.

CO-LaN (2004): CAPE-OPEN Laboratories Network webpage: www.colan.org

P. Deuflhard, E. Hairer, and J. Zugck (1987): One-step and extrapolation methods for differential-algebraic systems. Numerische Mathematik, (51):501–516.

W. Geffers, L. von Wedel, J. Wyes, W. Marquardt (2001): Integration of declarative and executable models in open simulation environments. In K. Panreck and F. Dörrscheidt, editors, 15. Symposium Simulationstechnik, pages 579–586, Paderborn, Germany, 11.9.-14.9.2001, SCS International.

Process Systems Enterprise (2002): gPROMS introductory user guide. release 2.1.1, Process Systems Enterprise, London, UK

U. Nowak and L. Weimann. A familiy of newton codes for systems of highly nonlinear equations. Technical Report TR-91-10, Konrad-Zuse-Zentrum für Informationstechnik, Berlin, 1991.

omniORB (2004): homepage: http://omniorb.sourceforge.net/

J.-P. Belaud, K. Alloula, J.-M. Le Lann, X. Joulia (2001): Open software architecture for numerical solvers: design, implementation and validation, ESCAPE 11 Conference, Kolding, Denmark. Available in Computer-Aided Chemical Engineering, 9, Elsevier, Editors R. Gani & S.B. Jorgensen, pp 967-972.

G. Schopfer, A. Yang, L. von Wedel, W. Marquardt (2004): CHEOPS: A tool-integration Platform for Chemical Process Modelling and Simulation, International Journal on Software Tools for Technology Transfer Vol. 6, No. 3, 2004, 186-202.

L. von Wedel, W. Marquardt, R. Gani (2002): Modelling Frameworks. In: B. Braunschweig, R. Gani (Eds.): Software Architecture and Tools for Computer Aided Process Engineering, Elsevier Science, 89-126.