17th European Symposium on Computer Aided Process Engineering – ESCAPE17
V. Plesu and P.S. Agachi (Editors)

Mixing computer algebra and numerical methods when solving CAPE models

Karim Alloula[a], Jean-Pierre Belaud[a], Jean-Marc Le Lann[a]

[a]Laboratoire de Génie Chimique (CNRS UMR 5503), INPT-ENSIACET, 118, route de Narbonne, 31077 Toulouse Cedex 04, France, Karim.Alloula@ensiacet.fr

Abstract

A "hybrid" approach, mixing computer algebra and numerical methods, is introduced for solving CAPE models. Mathematical expressions are handled using computer algebra techniques, and are evaluated to real numbers when the numerical methods require. A software realization, compliant with the CAPE-OPEN standard, provided accurate results on an implicit model.

Keywords: CAPE software tools, computer algebra, model driven engineering, MathML, component technology, CAPE-OPEN standard.

1. Introduction

Today, CAPE software community takes full advantage of advanced numerical methods and tools, and promotes the software component technology to solve the interoperability challenge. This paper presents a new approach for leveraging both the numerical simulation and the computer algebra techniques within the framework of the component technology:

1. describe the process model using *eXMSL on the Web*, a set of model building components, incorporating computer algebra capabilities;
2. solve and optimize this model using (CAPE-OPEN compliant) components.

2. Computer algebra versus numerical methods

Before introducing our proposal, we first study in detail the relations between the process model and the solving method in different cases: when simulating with a CAPE-OPEN compliant simulator, when simulating with a computer algebra system, or when using a computer algebra system as a pre-processor for a CAPE software tool.

2.1. Simulation in CAPE-OPEN compliant software tools

A CAPE-OPEN [1] compliant simulator will provide interoperability interfaces to other software tools. The main interfaces to set or get the process model belong to the *Unit Operations* package and to the *Physical Properties* package. The solving capabilities of the simulator may be compliant with several or all the interface specifications available in the *Numerics* package described in [2]. Any call to a method in the *Numerics* package sets or gets numerical values. Symbolic expression handling is not of concern with this standard .

2.2. Simulation in computer algebra systems

Part of a process may be simulated in today's computer algebra systems (CAS) such as Maple or Mathematica: model equations are entered using a syntax very close to usual mathematical notation. The solving process, involving both symbolic and numerical calculations is started by calling some specific routine. Nowadays high performance numerical mathematical libraries are seamlessly integrated in CAS [3], bringing numerical facilities to the computer algebra users. But, because general purpose CAS are not CAPE oriented, many facilities are missing. Simulating and optimizing a real plant with them remains unrealistic.

2.3. Computer algebra systems as pre-processors for CAPE software tools

Benefits of computer algebra techniques are well known in the CAPE community. One of the very first uses of computer algebra environments in our discipline was thermodynamic model derivative calculation as illustrated by [4]. [5] used the Mathematica system to simulate a reactive distillation column. In fact, simulation does not take place inside the computer algebra system. The CAS, used as a code generator, provides a very efficient simulation code, written in a compiled language. This pre-processing task can be viewed as a model transformation technique. Numerical mathematical libraries may solve the resulting model. This approach seems to be attractive because it conciliates accuracy and efficiency. However, the benefits of the computer algebra techniques are limited because the generated code handles numerical expressions only.

3. Numerical methods integrating computer algebra steps

The success of current computer algebra systems in all the engineering communities is owed (at least partly) to the fact that such systems seamlessly combine symbolic manipulation steps and numeric evaluation steps. Starting from a formal description of a model, involving symbols, functions and numbers, they deliver what most of us are interested in: numbers! The numeric evaluation steps may occur only at the end of the solving process but, most of the time, the models to be solved are a combination of analytical expressions and numerical expressions, such as thermodynamic correlations. Consequently, symbolic manipulation steps and numeric evaluation steps are very intricate in today's CAS.
We believe in such a paradigm, however, in order for contemporary CAPE software tools to take full advantage of these "hybrid" calculation techniques, computer algebra features have to be clearly identified, and delivered in a proper manner. The following paragraph lists the computer algebra features process simulation environments should involve when solving various model classes. We suggest providing those computer algebra services within a CAPE-OPEN process simulation environment. Finally, a case study involving this tool highlights the main benefits the CAPE software user may expect from mixing computer algebra and numerical methods when solving models.

3.1. Required computer algebra features

3.1.1. Linear equations

Linear systems involved when solving a process model come mainly from linearization of the non linear model required by the Newton-Raphson method. Those linear systems may have hundreds of thousands unknowns and are very sparse. Managing such a sparse structure for ensuring accuracy and for minimizing calculation time is crucial.
When a direct method is used for solving a large sparse linear system, equation reordering techniques may be valuable in order to limit the fill-in phenomenon. Those techniques are in fact very close to computer algebra techniques: reordering techniques work on lists of equations and variables, while computer algebra systems work on expressions viewed as lists of sub-expressions.
When an iterative method is used, and especially when a matrix-free method is employed, like the GMRES method [6], the numerical method can take full advantage of a computer algebra representation of the system $A \cdot x = b$ to be solved. Thus, each product of the incident matrix A times any real vector r is obtained efficiently by setting x to r and then evaluating numerically the expression $A \cdot x$.

3.1.2. Non linear equations

The model is entered the way it is edited in computer algebra systems, the residual function F being automatically calculated by subtracting the right hand sides from the left hand sides. The Newton operator $\mathrm{N}_F(x)$, which is the Jacobian matrix in the bijective case, is obtained by formal differentiation of the residual function. Matrix transpose and matrix products are required in the surjective and injective cases. At each Newton iteration, k, the formal mathematical expressions $F(x)$ and $\mathrm{N}_F(x)$, involving symbols x_1, x_2, ..., x_n, are evaluated numerically after setting x to the real vector x_k.

3.1.3. Differential algebraic equations

The model is entered the way it is edited in computer algebra systems, the residual function F being automatically calculated by subtracting the right hand sides from the left hand sides. Consistent initial conditions are computed by setting the independent variable to its initial numerical value and solving the resulting non linear system of equations. This step may require previous differentiations of the original system. The residual derivative $[\partial F/\partial x(t,x(t),\dot{x}(t)), \partial F/\partial \dot{x}(t,x(t),\dot{x}(t))]$, with respect to the dependent variables x and with respect to the dependent variable derivatives $\dot{x}$, is obtained by formal differentiation of the residual function F. According to the integrator needs, the formal mathematical expressions $F(t,x,dx)$, $\partial F/\partial x(t,x,dx)$ and $\partial F/\partial \dot{x}(t,x,dx)$, involving symbols t, x_1, x_2, ..., x_n, dx_1, dx_2, …, dx_n, may be evaluated numerically after setting t to some real value t_k, x to some real vector x_k and dx to some real vector dx_k.

3.1.4. Non linearly constrained optimization

The criterion f and the constraints are entered the way they are edited in computer algebra systems. Constraints are automatically converted to a canonical form by subtracting one side of the inequality from the other depending on the inequality operator. This canonical form defines a constraint function c. The criterion derivative $f'(x)$ and the constraint derivative $c'(x)$ are obtained by formal differentiation. According to the optimizer needs, the formal mathematical expressions $f(x)$, $c(x)$, $f'(x)$ and $c'(x)$, involving symbols x_1, x_2, ..., x_n, may be evaluated numerically after setting x to some real vector x_k.

3.2. eXMSL on the Web, a CAPE-OPEN problem solving environment

The previous guidelines were applied within the framework of the CAPE-OPEN software architecture. The result is a new software component, *eXMSL evaluation server*, in charge of building models at the equation level, and

evaluating them and their derivatives at any point. Model descriptions and model evaluations are both coded in an XML application: MathML 2.0 [7,8]. This textual and standardized format has been selected as the input for generating the computer algebra representation of our models. *eXMSL Evaluation Server* incorporates two CAPE-OPEN interfaces using .NET technologies: *Equation Set Object* and *Model*. It uses *Numerical Services Provider* which incorporates a CAPE-OPEN *Solver* component to provide a complete modelling and solving solution.
eXMSL Model Editor, is built on top of these business interfaces, allowing the end-users to edit, solve and optimize models through a graphical interface.

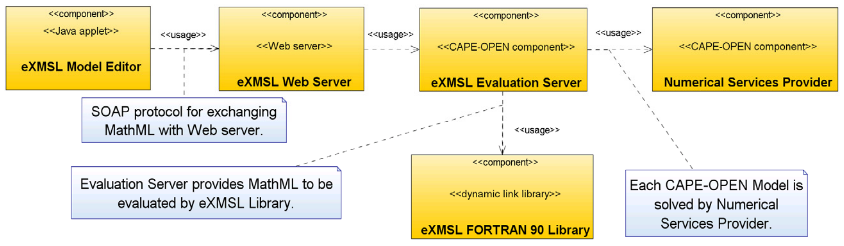

Figure 1 - eXMSL on the Web - UML component diagram

3.3. Case study: wine pH calculation

The case study model is detailed in [9]. A wine pH is calculated from the electro neutrality of the solution and from the ionic force definition. Those equations are formulated using all the species molalities, expressed as functions of the unknowns: the H^+ molality and the ionic force.
Acid molalities can be calculated from dissociation equilibria. These equilibria can be solved by hand or by using any computer algebra tool. This way, acid molalities are explicit functions of the unknowns. Such an approach has been adopted for calculations presented in .
On the other hand, we adopted a fully implicit function formulation, where dissociation equilibria are not solved in a pre-processing phase. All the model equations were given as they appeared in the initial formulation, the *eXMSL* software component being in charge of providing the acid molality numerical values corresponding to some unknown values, whenever needed during the solving process.

3.4. Results

The model was edited and solved under *eXMSL on the Web*, an application which can be accessed from any Java enabled browser. Any model is saved as a content MathML file, which can be viewed or partially evaluated in various

tools. Implicit function representation remains an original feature of the *eXMSL* component and cannot be directly evaluated in any other CAS.
Numerical results obtained for the referenced model were very accurate, although calculation time was slower than the calculation time associated to the numerical software. Fortunately however, because of its implicit formulation capabilities [10], *eXMSL* deals very easily with a model evolution (such as taking into account the tartaric acid complexification with calcium and potassium). Adding such evolutions to the initial model was much more time-costly in the context of the numerical software.

4. Conclusion

The CAPE community seems to be convinced of the benefits associated to open standards in software design, one goal being the production of interoperable software components. The CAPE community mainly regards computer algebra as a set of self-contained tools or as a pre-processing technique.
This work tries to introduce a combined approach, where computer algebra techniques are exploited in software components, in charge of model definition and evaluation. *S*eamless integration in a purely numerical solving environment is achieved using the CAPE-OPEN standard.
Mixing computer algebra and numerical methods on case studies, already brought reliable results: numerical calculations are very accurate, and models remain consistent during all their life cycles. This “hybrid” approach is related to process model driven engineering.

References

1. CAPE-OPEN standards and supporting documents, http://www.co-lan.org
2. J.-P. Belaud, K. Alloula, and J.-M. Le Lann, ESCAPE 11, Computer-Aided Chemical Engineering, 9, 2001, 967.
3. A. E. Trefethen and B. Ford, Mathematics and Computers in Simulation, No.54 (15-12-2000) 259.
4. R. Taylor, Fluid Phase Equilibria, No.129 (15-3-1997) 37.
5. M. F. Alfradique and M. Castier, Computers & Chemical Engineering, No.29 (15-8-2005) 1875.
6. Y. Saad and M. H. Schultz, SIAM Journal on Scientific and Statistical Computing, No.7 (1986) 856.
7. A. E. Trefethen and B. Ford, Mathematics and Computers in Simulation, No.54 (15-12-2000) 259.
8. Mathematical Markup Language (MathML) Version 2.0 (Second Edition), http://www.w3.org/TR/MathML2/
9. H. Akin, C. Brandam, X.-M. Meyer, and P. Strehaiano, SIMO 2006, France (Toulouse), 2006
10. K. Alloula, J.-P. Belaud, C. Leibovici, and J.-M. Le Lann, SIMO 2006, France (Toulouse), 2006