

A Differentiation System Designed for Dynamic Optimization Algorithms in Process Engineering

C. Bischof, M. Bücker, M. Petera, A. Vehreschild, J. Wyes
 Institute for Scientific Computing
 RWTH Aachen University
 D-52056 Aachen
 Germany

1 Introduction

Dynamic optimization is crucial in many scientific and engineering areas including process engineering. The algorithms have to solve a system of differential algebraic equations

$$M\dot{x} = F(x, p, t), \quad t \in (t_0, t_f), \quad x \in R^n, p \in R^q$$

with initial conditions

$$x(t_0) = x_0.$$

Here, x and p denote the vectors of state variables and system parameters, respectively. The aim is to minimize some objective function $\Phi(x(t_f))$ where t_f is the final time step. An important ingredient to the optimization algorithms is the evaluation of the derivatives of the states with respect to the parameters, $S = \partial x / \partial p$ and $D = \partial S / \partial p$. The first- and second-order sensitivity equations are given by

$$M\dot{S} = A \cdot S + K_1$$

and

$$(M \otimes I_q)\dot{D} = (A \otimes I_q) \cdot D + K_2,$$

respectively. Here, I_q is the identity of dimension q and

$$A = \frac{\partial F}{\partial x}, \quad K_1 = \frac{\partial F}{\partial p}, \quad K_2 = \left(\frac{\partial^2 F}{\partial x^2} S + \frac{\partial^2 F}{\partial x \partial p} \right) (I_n \otimes S) + \frac{\partial^2 F}{\partial x \partial p} S + \frac{\partial^2 F}{\partial p^2}.$$

Algorithmically, a given optimization algorithm involving first-order derivatives uses iterates of the form

for $k = 0, \dots, j - 1$

$$\begin{aligned} x_{k+1} &= x_k - (L_j U_j)^{-1} \cdot F_k \\ S_{k+1} &= S_k - (L_j U_j)^{-1} \cdot (A_k \cdot S_k + K_1), \end{aligned}$$

where $L_j U_j = A - M/h_j$ is a decomposition in an outer loop on j using a step size of h_j .

The aim of this work is to additionally use second-order derivatives by observing that the sensitivity equations for first- and second-order derivatives are linear. Hence, an implementation of

$$D_{k+1} = D_k - (LU)^{-1} \cdot (A_k \cdot D_k + K_2)$$

is added to the innermost loop.

2 The intermediate format CapeML

At the Institute for Process Systems Engineering at RWTH Aachen University, a framework for the solution of dynamic optimization algorithms using first-order derivatives is developed [5]. The system is called DyOS and is designed to be a general purpose dynamic optimization software. DyOS interacts with an interpreter where the objective function of a dynamical optimization problem is represented in intermediate format called CapeML. The advantage of this additional level of abstraction is that DyOS easily works together with different modeling languages, for instance Modelica [1] [3], or gPROMS [4] [2].

For an equation system described in gPROMS the Cape Open Interface with the gPROMS Interpreter is used. For a description in Modelica input language the transformation to CapeML is performed by the Interpreter, which works on the CapeML platform.

Suppose that `V_accel` is a variable occurring in some problem description written in Modelica, say. Suppose further that we are interested in the expression `4*arctan(V_accel)`. The interpreter transforms this expression into an XML-like syntax as follows:

```
<Expression>
  <Term>
    <Factor>
      <Number value='4' />
    </Factor>
    <Factor mul.op='MUL'>
      <FunctionCall fcn.name='arctan'>
        <Expression>
          <Term>
            <Factor>
              <VariableOccurrence definition='V_accel'>
            </VariableOccurrence>
            </Factor>
          </Term>
        </Expression>
      </FunctionCall>
    </Factor>
  </Term>
</Expression>
```

There is a number of diverse CapeML tags. Besides expressions there are definition of variables, constants, vectors and equations which can be transformed to the corresponding CapeML elements. The equation system in Modelica may be defined as a complex model structure, which can be completely transformed to a adequate CapeML model form.

After this transformation the interpreter builds a variable list. The names on the list reflect the model structure of the modelica source. For the left and right hand sides of each equation an expression tree is build. Each node of this syntax tree contains a variable reference, a constant value, an operation or a function call. The Interpreter can evaluate the values of the expressions with the actual variable values.

3 Exploiting structure in the derivative computations

In realistic dynamic optimization problems, the computation of derivative information is the most time-consuming part. We demonstrate how derivative computations can be carried out efficiently. The key ingredient is the exploitation of problem structure as described in the first section. We also point out how code written in CapeML is transformed using the forward mode of automatic differentiation. To this end, we do not perform an activity analysis but treat all variables as active. It is planned to use the same XML-tags to express the derivatives of a given XML code, i.e. no new tags are to be introduced.

References

- [1] P. Fritzson. *Principles of Object-Oriented Modeling and Simulation with Modelica 2.1*. Wiley-IEEE Press, 2004.
- [2] gPROMS Homepage. <http://www.psenterprise.com>.
- [3] Modelica Homepage. <http://www.modelica.org>.
- [4] C. C. Pantelides. *gproms : an advanced tool for process modelling, simulation and optimisation*. October 1996.
- [5] M. Schlegel, W. Marquardt, R. Ehrig, and U. Nowak. Sensitivity analysis of linearly-implicit differential-algebraic systems by one-step extrapolation. *Applied Numerical Mathematics*, 48(1):83–102, 2004.