On Jacobian evaluation for numerical integration of DAEs with partial symbolic information

Masoud Najafi and Ramine Nikoukhah

Abstract—The problem of Jacobian evaluation for numerical integration of DAEs is considered in the case where the DAE system is composed of subsystems some of which are defined in terms of analytical expressions and others are not. The application to dynamic system simulator Scicos is discussed.

KEYWORDS: Dynamic system simulation; Simulation software; Scicos; Numerical integration

I. INTRODUCTION

System and component level modeling often leads to DAEs (Differential-algebraic Equations) [1], [2]. The simulation of such systems requires specific numerical software such as DASSL [3], [4]. Jacobian evaluations are needed by these solvers. They can either be done numerically or symbolically. Analytical expression for the Jacobian obtained through symbolic computation is more accurate and leads to better simulation performance. So if the system description is available symbolically, Jacobian evaluation by symbolic manipulation should be performed, if possible. There are situations however where symbolic information is only partially available. This happens in particular when the system is defined in a modular way, for example by a block diagram, and only the dynamics of some of the blocks are available symbolically, the others are defined by computer programs.

In this paper we explain that this partial information can be exploited to improve the accuracy of the global Jacobian. The application of this result to simulation software Scicos is discussed.

II. MOTIVATION

Scicos (www.Scicos.org) is a free, open-source software for the simulation of hybrid dynamical systems. Scicos is a toolbox of Scilab (www.Scilab.org) [5], [6].

Dynamical systems in Scicos are defined as block diagrams. Until recently only explicit blocks (blocks with explicitly defined inputs and outputs) were allowed. The dynamics of these blocks are defined by C, Fortran, and Scilab routines and are seen by simulator as black boxes, i.e. no symbolic information is available about their dynamics [8], [9].

Recent addition allowed Scicos to use implicit blocks. They are in contrast with explicit blocks which have explicit inputs and outputs and their outputs are computed as a function of their input and internal states. Implicit blocks have ports which are not a-priori labeled as input or output. An electrical resistor is such an example, there is no input or output. In fact, depending on the circuit configuration, either current or voltage can be considered as input or output. In Fig. 3, the blocks in green are implicit blocks and the rest of blocks are explicit ones. Implicit blocks are very useful for modeling physical components (e.g. a resistor or transistor in an electrical circuit or a pump or hose in hydraulic circuits). The Modelica language has been used to describe the internal dynamics of these implicit blocks [13].

The introduction of these new blocks has had two consequences for the numerical solver:

1) The global system to be simulated is a DAE (with explicit blocks the resulting global system was an ODE).
2) The implicit part of the dynamics, defined in Modelica, contains symbolic expressions which can be used to obtain an analytical expressions for a part of the entire Jacobians matrix.

Fig. 1. A system consisting several implicit blocks (components)

Consider the Scicos diagram illustrated in Fig. 1. This model contains both explicit and implicit blocks. In this case all of the continuous-time dynamics is in the implicit parts so that when Scicos generates Modelica program for this part, generates a C code from Modelica program and replaces this part with an explicit block whose internal dynamics is described by this program, the resulting explicit Scicos model is given in Fig. 2.

In this case the continuous dynamics is in the block MBlock and the DAE equations are symbolically available. This means that the expression of the Jacobian can be computed analytically and made available to the numerical solver. Unfortunately, this is not always the case as it can be seen in the example illustrated in Fig. 3.

In this case, after the substitution of the implicit part by an explicit block we obtain the diagram in Fig. 4. Note that in this case the explicit part contains also continuous time
III. JACOBIAN EVALUATION OF MIXED-MODEL SYSTEMS

Consider the mixed-model DAE system illustrated in Fig 6. This system can be represented as follows:

\[ 0 = \phi(\dot{x}, x) \]  

where \( x \), the state vector, is given by

\[ x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \]

we assume the analytical expressions of \( F \) and \( G \) are available and thus can be differentiated symbolically. On the other hand \( H \) and \( L \) can only be evaluated numerically. If \( n_t \) is the total number of continuous-time states and the analytical part of model contains \( n_1 \) states, \( n_o \) explicit outputs and \( n_i \) explicit inputs, then the size of \( H, L, F, \) and \( G \) will be \( (n_t - n_1) \times (n_t - n_1), \) \( (n_i) \times (n_t - n_1), \) \( (n_1) \times (n_1) \), and \( (n_o) \times (n_1) \) correspondingly.

Solving this system numerically requires the evaluation of the Jacobian (see Appendix)

\[ \alpha \frac{\partial \phi}{\partial \dot{x}} + \frac{\partial \phi}{\partial x} \]

for any parameter \( \alpha \).

**Theorem:**

Consider the DAE system illustrated in Fig 6 and assume that it contains no algebraic loop, i.e.

\[ \frac{\partial M}{\partial u_1} = 0 \]
where

\[
M = G(\dot{x}_2, x_2, L(\dot{x}_1, x_1, u_1))
\]

\[
N = L(\dot{x}_1, x_1, G(\dot{x}_2, x_2, u_2)).
\]

Let

\[
J = \frac{\partial \phi}{\partial x} + \frac{\partial \phi}{\partial y}.
\]

Then

\[
J = \begin{pmatrix}
J_{11} & J_{12} \\
J_{21} & J_{22}
\end{pmatrix}.
\]

where

\[
\begin{aligned}
J_{11} &= \frac{\partial H}{\partial x_1} + \alpha \frac{\partial H}{\partial x_1} + \frac{\partial H}{\partial u_1} \frac{\partial L}{\partial x_1} + \alpha \frac{\partial L}{\partial x_1} \\
J_{12} &= \frac{\partial H}{\partial u_1} + \frac{\partial G}{\partial u_1} + \frac{\partial G}{\partial x_1} \\
J_{21} &= \frac{\partial F}{\partial u_2} + \frac{\partial L}{\partial x_2} + \alpha \frac{\partial L}{\partial x_2} \\
J_{22} &= \frac{\partial F}{\partial x_2} + \alpha \frac{\partial F}{\partial x_2} + \frac{\partial F}{\partial u_2} + \frac{\partial G}{\partial u_2} + \alpha \frac{\partial G}{\partial u_2}.
\end{aligned}
\]

Proof:

Note that the system equations are the followings

\[
\begin{aligned}
0 &= H(\dot{x}_1, x_1, u_1) \\
0 &= F(\dot{x}_2, x_2, u_2) \\
y_1 &= L(\dot{x}_1, x_1, u_1) \\
y_2 &= G(\dot{x}_2, x_2, u_2) \\
u_1 &= y_2 \\
u_2 &= y_1
\end{aligned}
\]

from which we obtain the following expressions

\[
\begin{aligned}
J_{11} &= \frac{\partial H}{\partial x_1} + \alpha \frac{\partial H}{\partial x_1} + \frac{\partial H}{\partial u_1} \frac{\partial L}{\partial x_1} + \alpha \frac{\partial L}{\partial x_1} \\
J_{12} &= \frac{\partial H}{\partial u_1} + \frac{\partial G}{\partial u_1} + \frac{\partial G}{\partial x_1} \\
J_{21} &= \frac{\partial F}{\partial u_2} + \frac{\partial L}{\partial x_2} + \alpha \frac{\partial L}{\partial x_2} \\
J_{22} &= \frac{\partial F}{\partial x_2} + \alpha \frac{\partial F}{\partial x_2} + \frac{\partial F}{\partial u_2} + \frac{\partial G}{\partial u_2} + \alpha \frac{\partial G}{\partial u_2}.
\end{aligned}
\]

In (6), the term \( \frac{\partial u_1}{\partial x_1} \) can be rewritten as

\[
\frac{\partial u_1}{\partial x_1} = \frac{\partial y_2}{\partial x_1} = \frac{\partial y_2}{\partial u_2} \frac{\partial u_2}{\partial x_1} = \frac{\partial G}{\partial u_2} \left( \frac{\partial L}{\partial x_1} + \alpha \frac{\partial L}{\partial x_1} + \frac{\partial L}{\partial u_1} \right).\]

From (3) we have \( \frac{\partial L}{\partial u_2} \frac{\partial L}{\partial u_1} = 0 \), thus we obtain

\[
\frac{\partial u_1}{\partial x_1} = \frac{\partial L}{\partial u_2} \left( \frac{\partial L}{\partial x_1} + \alpha \frac{\partial L}{\partial x_1} \right).\]

Hence from (6) and (10) we obtain the \( J_{11} \) expression in (5).

For \( J_{12} \), (7) can be rewritten as

\[
\begin{aligned}
J_{12} &= \frac{\partial H}{\partial u_1} \frac{\partial u_1}{\partial x_2} \\
&= \frac{\partial H}{\partial y_2} \frac{\partial y_2}{\partial x_2} \\
&= \frac{\partial H}{\partial G} \frac{\partial G}{\partial x_2} + \frac{\partial G}{\partial u_2} \frac{\partial L}{\partial u_1} + \frac{\partial L}{\partial u_2} \frac{\partial L}{\partial u_1}.
\end{aligned}
\]

Since \( \frac{\partial L}{\partial u_2} \frac{\partial L}{\partial u_1} = 0 \), the desired expression for \( J_{12} \) in (5) can be obtained.

(5)

To obtain \( J_{21} \), (8) can be rewritten

\[
\begin{aligned}
J_{21} &= \frac{\partial F}{\partial u_2} \frac{\partial u_2}{\partial x_1} \\
&= \frac{\partial F}{\partial y_2} \frac{\partial y_2}{\partial x_1} \\
&= \frac{\partial F}{\partial G} \frac{\partial G}{\partial x_1} + \frac{\partial G}{\partial u_2} \frac{\partial L}{\partial u_1} + \frac{\partial L}{\partial u_2} \frac{\partial L}{\partial u_1}.
\end{aligned}
\]

From (4) we have \( \frac{\partial L}{\partial u_1} \frac{\partial L}{\partial u_2} = 0 \), thus we obtain the desired expression for \( J_{21} \) in (5).

To obtain \( J_{22} \), the term \( \frac{\partial L}{\partial u_2} \frac{\partial L}{\partial u_1} \) in (9) can be rewritten as

\[
\frac{\partial u_2}{\partial x_1} = \frac{\partial y_1}{\partial x_1} \frac{\partial u_1}{\partial x_1} + \frac{\partial y_1}{\partial u_1} \frac{\partial u_1}{\partial x_1} = \frac{\partial L}{\partial u_1} \frac{\partial L}{\partial x_1} + \alpha \frac{\partial L}{\partial x_1} + \frac{\partial L}{\partial u_1} \frac{\partial L}{\partial u_1}.
\]

Since \( \frac{\partial L}{\partial u_2} \frac{\partial L}{\partial u_1} = 0 \) we have

\[
\frac{\partial u_2}{\partial x_1} = \frac{\partial L}{\partial u_1} \frac{\partial L}{\partial x_1} + \alpha \frac{\partial L}{\partial x_1} + \frac{\partial L}{\partial u_1} \frac{\partial L}{\partial u_1}.
\]

Combining (9) and (11) we get \( J_{22} \) expression in (5).

IV. SCICOS IMPLEMENTATION AND SIMULATION

The numerical solver in Scicos needs the Jacobian obtained in (5). In this expression, the following can be computed analytically because their expressions are symbolically defined in Modelica:

\[
\begin{aligned}
\frac{\partial F}{\partial x_2}, \frac{\partial F}{\partial x_2}, \frac{\partial F}{\partial x_2}, \frac{\partial G}{\partial x_2}, \frac{\partial G}{\partial x_2}, \frac{\partial G}{\partial u_2}, \frac{\partial G}{\partial u_2}, \frac{\partial G}{\partial u_2}.
\end{aligned}
\]
These expressions are indeed computed by the Modelica parser and compiler integrated in Scicos. On the other hand numerical differentiation is used to obtain the following expressions.

\[
\frac{\partial H}{\partial x_1} + \alpha \frac{\partial H}{\partial x_1} + \frac{\partial L}{\partial x_1} + \alpha \frac{\partial L}{\partial u_1} + \frac{\partial H}{\partial u_1} + \frac{\partial L}{\partial u_1} \tag{13}
\]

To compute \( \frac{\partial H}{\partial x_1} + \alpha \frac{\partial H}{\partial x_1} \) and \( \frac{\partial L}{\partial x_1} + \alpha \frac{\partial L}{\partial u_1} \) numerically, \( x_1 \) and \( H \) are perturbed and \( h \) and \( L \) are evaluated and compared with their previous values (when \( x_1 \) and \( \dot{x}_1 \) are not perturbed). Thus the approximate value of \( i^{th} \) element of \( \frac{\partial H}{\partial x_1} + \alpha \frac{\partial H}{\partial x_1} \) and \( \frac{\partial L}{\partial x_1} + \alpha \frac{\partial L}{\partial u_1} \) are

\[
\approx H_i(\dot{x}_1 + \alpha \sigma_j, x_1 + \sigma_j, u_1) - H_i(\dot{x}_1, x_1, u_1) \sigma_j
\]

\[
\approx L_i(\dot{x}_1 + \alpha \sigma_j, x_1 + \sigma_j, u_1) - L_i(\dot{x}_1, x_1, u_1) \sigma_j
\]

To compute the remaining terms of 13, the inputs of explicit blocks (outputs of generated explicit block) are disconnected and perturbed and \( H \) and \( L \) are evaluated and compared with their previous values (when \( u_1 \) is not perturbed). Thus the approximate value of \( i^{th} \) element of \( \frac{\partial H}{\partial u_1} \) and \( \frac{\partial L}{\partial u_1} \) are

\[
\frac{\partial H_i}{\partial u_{1j}} \approx \frac{H_i(\dot{x}_1, x_1, u_1 + \sigma_j) - H_i(\dot{x}_1, x_1, u_1)}{\sigma_j}
\]

\[
\frac{\partial L_i}{\partial u_{1j}} \approx \frac{L_i(\dot{x}_1, x_1, u_1 + \sigma_j) - L_i(\dot{x}_1, x_1, u_1)}{\sigma_j}
\]

Since in most cases nonlinearities and stiffness are found in implicit part, the use of exact expressions improves a great deal the precision of the evaluated Jacobian furnished to the solver. This results in more accurate simulation and can even avoid simulation failures in some cases.

A. An example

Consider the example shown in Fig. 1. The automatically generated DAE is given in (14).

\[
\begin{align*}
n_0 &= 220 \\
r_1 &= 50 \\
r_2 &= 0.0001 \\
r_3 &= 500 \\
r_4 &= 20 \\
v_0 &= r_0 \sin(\pi r_1 t) \\
v_1 &= r_3 (x_3 - f(-x_2) - f(x_1 - x_2)) \\
v_2 &= r_4 (f(x_2 + v_1) - f(-x_2)) \\
v_3 &= x_2 + v_0 + v_1 + v_2 \\
0 &= x_1 + v_0 + v_2 \\
0 &= x_4 + v_1 \\
0 &= f(x_1 - x_2) + f(-x_2) - f(x_2 + v_1) - f(v_3) \\
0 &= r_2 x_4 - x_3
\end{align*}
\tag{14}
\]

where the function \( f(x) \), which is the characteristic curve of an electrical diode, is defined as:

\[
\begin{align*}
R &= 10^8 \\
L_d &= 10^{-6} \\
V_t &= 0.04 \\
N &= 15 \\
f(x) &= \begin{cases} \\
\frac{x}{R} - L_d & \text{if } x < NV_t \\
\frac{x}{R} - L_d \left(\frac{V_t - NV_t}{V_t}\right) e^N & \text{if } x \geq NV_t
\end{cases}
\end{align*}
\]

This model is simulated in Scicos for two cases. First we use numerical Jacobian calculated by DASKR. In this case, as illustrated in Fig. 7, solver stops integration at 0.0012 with this error message “IDID = -7 -- The nonlinear system solver in the time integration could not converge.”

![Fig. 7. Simulation result of model of Fig. 1 with numerical Jacobian provided by DASKR](image)

For the same example but using the analytical Jacobian, the integration finishes successfully at 0.1, as shown in Fig. 8.

![Fig. 8. Simulation result of model of Fig. 1 with analytical Jacobian](image)
V. CONCLUSION

Although a little complicated to implement, with an analytic Jacobian the accuracy increases and as a result the convergence and the converge time is reduced drastically. Current version of Scicos provides the analytical Jacobian for implicit part of the model, while the Jacobian of explicit part is computed numerically. In this paper obtaining the global Jacobian of model using these two parts has been explained.

REFERENCES


APPENDIX

Scicos uses DASKR as numerical solver. It is the new version of DASRT (DASSL with root finder). DASKR uses BDF (backward differentiation formula) methods to solve a system of DAEs or ODEs. The methods are variable step-size variable order. The system of equations in DASKR is written in an implicit ODE form like

$$
\phi(\alpha_n x_n + \beta_n, x_n) = 0
$$
on each time step. Here, $\alpha_n$ and $\beta_n$ are scalars which depend on the method and step-size. In DASKR, this system is solved by a modified Newton iteration. Each iteration of the Newton method requires the solution of a linear system

$$
J_n x_n^{k+1} = B_n
$$

where the so-called Jacobian matrix $J$, is given by

$$
J = \beta_n \frac{\partial \phi}{\partial x} + \alpha_n \frac{\partial \phi}{\partial \dot{x}}
$$

The Jacobian matrix can be either provided by user or computed numerically by DASKR. If DASKR is called by Info(5)=0, then the Jacobian is computed automatically by DASKR. If Info(5)=1 DASKR uses the Jacobian matrix provided instead.