The module partDEQ - solving partial differential equations using SIMUL_R

R. Ruzicka
SIMUTECH
Hadikgasse 150, A-1140 Vienna

Abstract

This contribution will show all features of the simulation language SIMUL_R, which make it possible to solve systems of partial differential equations (PDEs) and display their solutions. The modelling features are contained in a module called *partDEQ*. Some examples will show how to use partDEQ to solve PDE problems.

Introduction

Most modern simulation languages for continuous systems' simulation contain features for solving ordinary differential equations (ODEs). But in many cases ODEs are not as accurate as is necessary to describe all dynamic features of a real system:

think of electronical circuits with long wires or e.g. the bus of a computer; these cannot be simply modelled by using for example a capacitor and a resistor to show the effects of voltage reflections;

or the absorbtion of pollutants by a filter: you need to model the process of absorbtion to be able to predict, when the filter will be filled;

or the closing of valves in waterworks: the waves of water pressure and velocity have to be modelled using PDEs to simulate extreme and dangerous states.

There are a lot of tools, each spezialized in one specific field of PDE simulation: finite element methods for mechanical purposes or special numerical libraries for special types of PDEs. Nevertheless those are difficult to use for general purposes and mostly have no user friendly interface for modelling and presentation of results.

How to add PDE features to a simulation system for ODEs?

A modelling utility for PDEs in an environment for ODEs, like SIMUL_R, should be

- fully integrated into the ODE system (PDEs and ODEs combinable)
- easy to use
- the result functions must be easy accessible
- the results should be drawn over the locality dimensions without user "chin ups"

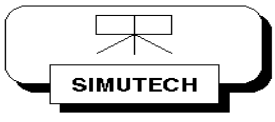
SIMUL_R features for PDEs

SIMUL_R a priori offers some features, which help modelling PDEs:

- a very powerful macro and meta language
- sorting of equations
- easy modelling of implicite problems
- so-called *cross plots*

Meta and macro commands

SIMUL_R contains all macro features known from the programming langugage C, but extended by very important points:



+ recursive macros
+ varying number of parameters for the same macro
+ meta loops (for, while)

Equation Sorting

The equation sorting algorithm of SIMUL_R - sorting occurs within the DERIVATIVE section of a model for finding dependencies of variables -

+ automatically detects algebraic loops,
+ searchs for a minimal set of variables, necessary to be iterated for solving an implicite problem,
+ transforms the equations into a zero search problem (yet available algorithms, choosable by menu: damped Newton, DASSL),
+ delivers information messages, e.g. about the structure of the equation matrix

  banded matrices can be easily detected, necessary variable resorting can be performed.

+ SIMUL_R offers special banded matrices methods for the Newton and DASSL algorithms.

Implicite problems

can be modelled in SIMUL_R as

$0=g(x)$ zero point notation

or

$x=f(x)$ fix point notation

(as well for vectors and matrices). Therefore it is much easier to use special algorithms for PDEs, like Crank-Nicolson's method.

Cross plots

mean, that the time variations of a set of variables (e.g. variables representing the solution of a PDE over the locality dimension) can be drawn in the vertical y direction of a plot (with constant horizontal x position) and are automatically connected.

These features have been used to build up the partDEQ module with a library of macros, which translate a PDE notation into a set of equations and algorithms, which can be treated by usual SIMUL_R.

Modelling PDEs

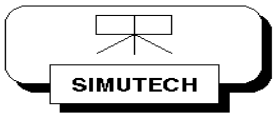
PDEs of the form

$$\frac{d^k y}{dt^k} = f\left(t, y, \frac{dy}{dt}, \frac{d^2y}{dt^2}, .., \frac{d^{k-1}y}{dt^{k-1}}, x, \frac{dy}{dx}, \frac{d^2y}{dx^2}, .., \frac{d^m y}{dx^m}\right)$$

where

t	time (first independend variable)	*f*	an arbitrary expression
x	locality (second independend variable)	*k*	maximum derivative order over t
y	state variable	*m*	maximum derivative order over x

are solved and can be inserted as parameters of a partDEQ macro; additionally the initial conditions and the length of locality is specified.



The macros are also available for up to 3 locality variables (3 dimensions) with mixed local derivatives.

Special macros are used to specify boundary conditions (fixed conditions, or switchable conditions).

The results are sampled in user specified array variables, which can be simply plotted using cross plots.

Currently the two methods

- method of lines
- Crank Nicolson (implicite method)

are available, which only differ - at the user interface - in the name prefix of the macro.

Examples

Introductional example

The first example shows a simple model of sucking water into wood:

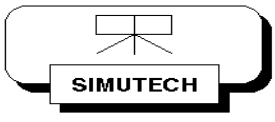
$$\frac{dy}{dt} = D * \frac{d^2y}{dx^2} \qquad \text{with} \qquad y(t0,x) = 1$$

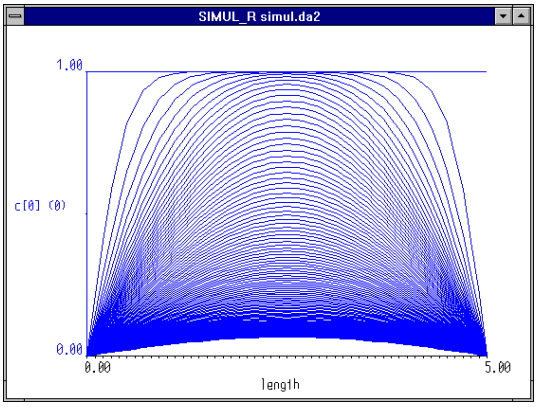
At time *T* a step from 1 to 0 is performed on the left and right boundaries. Here the first and second order locality derivatives are named *dyx* and *dyxx*, respectively.

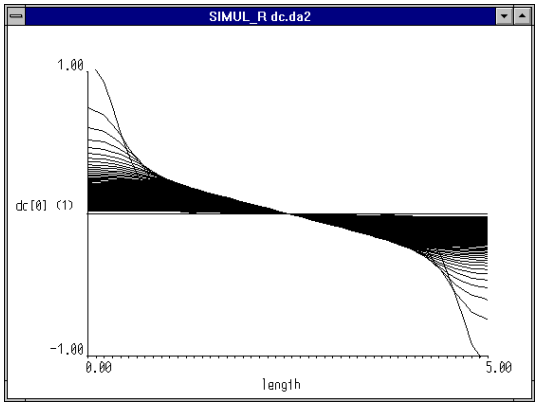
The model:

```
#set n=50#             " number of discretization points "
#include 'simcomac.def'
#include 'partdeq.def'       " include module partDEQ "

wood {

       CONSTANT tend=1000, len=1, D = 1e-5, T=0.2;
       float yl, yr, y[#n+1#], dyx[#n+1#], dyxx[#n+1#];
       int i;

       DYNAMIC {
             DERIVATIVE {
                  yl=1-#step(T);               " left bound "
                  yr=1-#step(T);               " right bound "
                  #partDEQlrc(1,D*dyxx[i],i,len,yl,yr,y,1,dyx,dyxx)
             }
             TERMINATE t>=tend;                " termination condition "
       }
}
```

It is easy to draw the state variable results and the locality derivatives over *x*:







Net of water pipes

The second example shows pressure waves in a pipe net of waterworks: each pipe is described by the two coupled PDEs (for pressure *p* and velocity *v*)

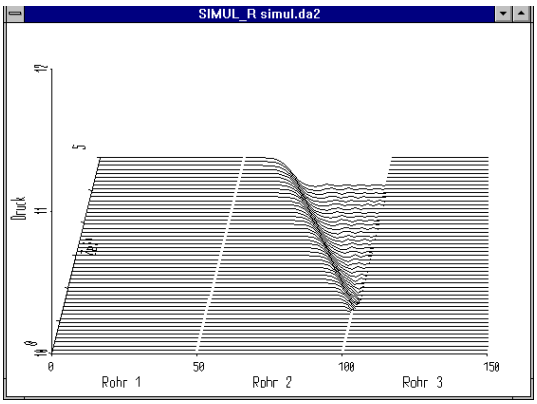
$$\frac{\partial p}{\partial t} = -\frac{c*c}{g} * \frac{\partial v}{\partial x} \quad \text{and} \quad \frac{\partial v}{\partial t} = -\frac{\partial p}{\partial x} * g - \frac{l}{2*D} * v * |v| \quad ,$$

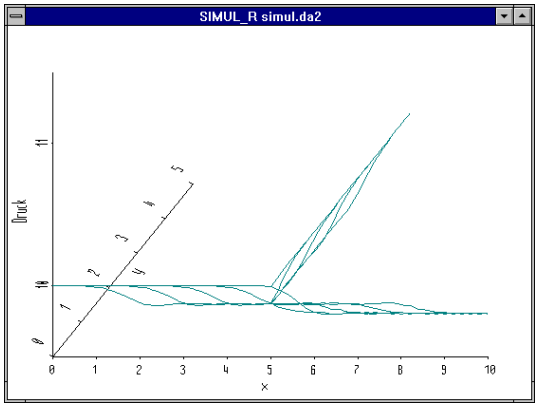
the net is described using bond graphs.

On the left hand side, there is a water container with constant pressure, on the right hand side and at the top there are two valves. A water consumer opens a valve: this leads to water reflections through the entire net.

A pipe is modelled using partDEQ macros for the PDEs and is formulated as a macro. The net is modelled by the bond graph tool BAPS using this macro. This allows for an easy changing of the net's topology.

The pictures show the pressure in the three pipes: the first wave after closing one valve and five states (at five different points of time over the real topology).





Future developments

In the near future there will be available methods for

- automatic adaption of the discretization points and
- new methods for computing the differential quotients.

SIMUTECH

Literature

- R.Ruzicka: *SIMUL_R - eine Simulationssprache mit speziellen Befehlern zur Modelldarstellung und -analyse*, Informatik Fachberichte 179, Proceedings of the 5th Symposium Simulationstechnik, Springer, Aachen, 1988
- R.Ruzicka: *Environments For SIMUL_R*, 3rd European Simulation Congress, Proceedings, Edinburgh, 1989
- J.Niwinski, R.Ruzicka: *Online Simulation With SIMUL_R*, SCSC 91, Baltimore Maryland, 1991
- R.Ruzicka: *SIMUL_R PARALLEL - Hardware-in-the-loop Simulation mit Transputern unter Windows*, Fortschritte in der Simulationstechnik, Band 6, Proceedings of the 8th Symposium Simulationstechnik, Vieweg, Berlin, 1993
- R.Ruzicka: *Optimierung technischer Systeme mittels Evolutionsstrategien - ein Standardverfahren in SIMUL_R*, Fortschritte in der Simulationstechnik, Band 9, Proceedings of the 9th Symposium Simulationstechnik, Vieweg, Stuttgart, 1994