C1 Lithium-Cluster Dynamics under Electron Bombardment – SDX

Numerical approach

Simulator: SDX™ (System Dynamics) is a programmable Windows environment for technical computing, modelling and simulation. It runs applications written and compiled as **`dlls`** in the Fortran compiler **IDE**. **SDX** is available for PCs running under Win 9x and Win NT 4.0 or later.

Model: The FORmula TRANslated model is a straightforward one-to-one transcription of the mathematical model aided by the built-in SDX modelling functions. The **`include`** file, not shown, specifies and exports the model variables for interactive runtime access. It is produced by the SDX Code Generator, a separate *Win utility* program. The **`model dll`** is loaded into SDX where simulation experiments are conducted as a native windows application.

```
subroutine model
*       Lithium cluster dynamics
        include 'sdx_gui.inc'
        external rate
        parameter (n = 3, init = -1)
        real x(n),mo
        data ro/84.99/, mo/1.674/, fo/9.975/,
       &     pc/1.e4/, tend/10/, inix/1/, inie/1/

        if(mode() .eq. init) then
           x(1) = ro
           x(2) = mo
           x(3) = fo
        endif
.
        t = time()
        p = (1 - sgn(t))*pc
        call integ (rate,x,p,n,inix)
.
        call esched (inie,tend)
        end
*-----------------------------------------------
        subroutine rate (x,p,t,dx)
*       eom:  dx/dt = f(x,u,t)
        include 'sdx_gui.inc'
        real x(*),dx(*),m,
   &       dr/.1/, kr/1/, dm/1/, kf/.1/, lf/1.e3/
.
     r = x(1);  m = x(2);  f = x(3)
.
     dx(1) = -dr*r + kr*m*f
     dx(2) =  dr*r - dm*m + (kf*f - kr*m)*f
     dx(3) =  dr*r + 2*dm*m-(kr*m + 2*kf*f + lf)*f+p
     end
```

Task a Simulation of the System. Set **`inie`** -- from *Edit Variable* dialog -- timer option in the event scheduling function. The timing results, extracted via a log *file view* facility, reflect the compiled speeds (GUI updates turned off). On AMD K6-II, 333MHz system:

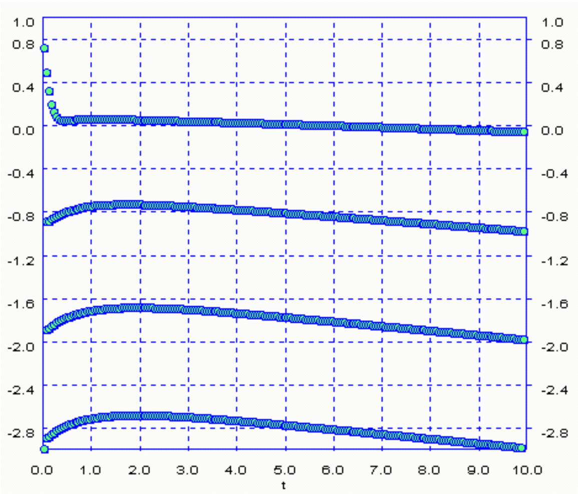
Algorithm	timing (ms)
adaptive step predictor/corrector	80
recursive state space solver	10

The difference in performance is due to the recursive algorithm, which requires a single derivative evaluation per step; efficiency may thus be measured against the ideal lower bound for numerical integration:

$$\frac{dx}{dt} = f(x,u) = A \cdot x + g(x,u)$$

$$x_{k+1} = x_k + T(A,dt) \cdot f(x_k,u_k)$$

Task b. Parameter variation Set **`lf`** -- from *Edit Variable* dialog -- parameter for **`log(lf)`** stepped 1:4, select variables for graphics display, and make the run(s). Overlaid run-time graphics, **`log(f)`** vs. **`t`**, was exercised in real-time computational mode. It shows the dominant dynamics and indicates a rapid initial transient (~**`1/lf`** sec) – see figure below.



Task c. Calculation of Steady States: Set **`inix`** - from *Edit Variable* dialog - **`trim`** option in the integrate function, and likewise for the **`pc`** parameter. The system trimmed states, shown in the table, may be viewed via the *numeric display* facility.

pc	r	m	f
0	0	0	0
10000	1000	10	10

Dr. B. Voh, Eclipse Software,
PO Box 25, Campbell, CA 95009, USA
www.sdynamix.com
info@sdynamix.com