ARGESIM Report no. 7

EUROSIM Comparison C1 "Lithium Cluster Dynamics"

Solutions and Results

F. Breitenecker, I. Husinsky
editors

ISBN 3-901608-07-9

ISBN 3-901608-07-9

ARGESIM Report No. 7

ARGE Simulation News (ARGESIM)
c/o Technical University of Vienna
Wiedner Hauptstr. 8-10
A-1040 Vienna, Austria
Tel: +43-1-58801 5386, 5374, 5484
Fax: +43-1-5874211
Email: argesim@simserv.tuwien.ac.at
WWW: <URL:http://eurosim.tuwien.ac.at/>

Printed by *CA Druckerei*, Vienna, Austria

FOREWORD

EUROSIM, the Federation of European Simulation Societies, started in 1990 the publication of the journal **EUROSIM Simulation News Europe (SNE)**, a newsletter distributed to all members of the European simulation societies under **EUROSIM's** umbrella and to people and institutions interested in simulation. **SNE** is also part of **Simulation Practice and Theory (SIMPRA)**, the scientific journal of **EUROSIM**.

The idea of the journal **SNE** (circulation 2500; edited by F. Breitenecker and I. Husinsky, ARGE Simulation News (**ARGESIM**) , Technical University of Vienna, Austria; three issues per year) is to disseminate information related to all aspects of modeling and simulation.

The contents of **SNE** are news in simulation, simulation society information, industry news, calendar of events, essays on new developments, conference announcements, simulation in the European Community, introduction of simulation centers and comparison of simulation software, simulators and (parallel) simulation techniques.

The series on comparisons of simulation software has been very successful. Based on simple, easily comprehensible models the software comparisons compare special features of modeling and experimentation within simulation languages:

- modeling technique
- event handling
- submodel features
- numerical integration
- steady-state calculation
- frequency domain
- plot features
- parameter sweep
- postprocessing
- statistical features
- statistical processors
- control strategies
- optimization
- random numbers
- complex strategies
- animation, etc.

Seven Software Comparisons, four continuous ones and three discrete have been set up. Furthermore, a second type of comparisons, the Parallel Comparison has been initiated.

The continuous comparisons are:

- Comparison 1 (C1; Lithium-Cluster Dynamics under Electron Bombardment, November 1990) deals with a stiff system;
- Comparison 3 (C3; Analysis of a Generalized Class-E Amplifier, July 1991) focusses on simulation of electronic circuits and eigenvalue analysis;
- Comparison 5 (C5; Two State Model, March 1992) requires very high accuracy computation;
- Comparison 7 (C7; Constrained Pendulum, March 1993) deals with state events.

The discrete comparisons are:

- Comparison 2 (C2; Flexible Assembly System, March1991) gives insight into flexible structures of discrete simulators;
- Comparison 4 (C4; Dining Philosophers, November 1991) involves not only simulation but also different modeling techniques like Petri nets;
- Comparison 6 (C6; Emergency Department - Follow-up Treatment, November 1992) deals with complex control strategies;

SNE 10 introduced a new type of comparison dealing with the benefits of distributed and parallel computation for simulation tasks. Three test examples have been chosen to investigate the types of parallelization techniques best suited to particular types of simulation tasks.

Up to now, 100 solutions have been sent in. The table at the end of this ARGESIM report shows the number of solutions for the Software Comparisons as well as for the Parallel Comparison. The series will be continued.

This ARGESIM Report summarizes and discusses the solutions and results sent in for Comparison 1 (C1) ***„Lithium Cluster Dynamics"***.

The report starts with a summary, which is an extended version of a contribution to the congress **EUROSIM'95**.
The presentation of the solutions sent in starts with the definition of this EUROSIM comparison (Definition and Definition with remarks, resp), formulated by W. Husinsky in **SNE 0** and **SNE 1**, resp..

In the following the solutions sent in up to now are printed in chronological order. Each solution is represented by the page printed in **SNE** and, if available, by the originals sent in by the originators. It is evident, that early solutions are accompanied by more original paper work.

As conclusion a Table of the EUROSIM Comparisons and the number of solutions sent in is given.

F. Breitenecker, I. Husinsky, Editors

About ARGESIM

ARGE Simulation News (ARGESIM) is a non-profit working group providing the infra structure for the administration of **EUROSIM** activities and other activities in the area of modelling and simulation.

ARGESIM organizes and provides the infra structure for

- the production of the journal EUROSIM Simulation News Europe
- the comparison of simulation software (EUROSIM Comparisons)
- the organisation of seminars and courses on modelling and simulation
- COMETT Courses on Simulation
- "Seminare über Modellbildung und Simulation"
- development of simulation software, for instance: mosis - continuous parallel simulation, D_SIM - discrete simulation with Petri Nets, GOMA - optimization in ACSL
- running a WWW - server on EUROSIM activities and on activities of member societies of EUROSIM
- running a FTP-Server with software demos, for instance
    * demos of continuous simulation software
    * demos of discrete simulation software
    * demos of engineering software tools
    * full versions of tools developed within ARGESIM

At present ARGESIM consists mainly of staff members of the Dept. Simulation Technique and of the Computing Services of the Technical University Vienna.

In 1995 ARGESIM became also a publisher and started the series **ARGESIM Reports**. These reports will publish short monographs on new developments in modelling and simulation, course material for COMETT courses and other simulation courses, Proceedings for simulation conferences, summaries of the EUROSIM comparisons, etc.

Up to now the following reports have been published:

No.	Title	Authors / Editors	ISBN
# 1	Congress EUROSIM'95 - Late Paper Volume	F. Breitenecker, I. Husinsky	3-901608-01-X
# 2	Congress EUROSIM'95 - Session Software Products and Tools	F. Breitenecker, I. Husinsky	3-901608-02-8
# 3	EUROSIM'95 - Poster Book	F. Breitenecker, I. Husinsky	3-901608-03-6
# 4	Seminar Modellbildung und Simulation - Simulation in der Didaktik	F. Breitenecker, I. Husinsky, M. Salzmann	3-901608-04-4
# 5	Seminar Modellbildung und Simulation - COMETT - Course "Fuzzy Logic"	D. Murray-Smith, D.P.F. Möller, F. Breitenecker	3-901608-05-2
# 6	Seminar Modellbildung und Simulation -COMETT - Course "Object-Oriented Discrete Simulation"	N. Kraus, F. Breitenecker	3-901608-06-0
# 7	EUROSIM Comparison 1 - Solutions and Results	F. Breitenecker, I. Husinsky	3-901608-07-9
# 8	EUROSIM Comparison 2 - Solutions and Results	F. Breitenecker, I. Husinsky	3-901608-08-7

For information contact: ARGESIM, c/o Dept. Simulation Techniques,
attn. F. Breitenecker, Technical University Vienna
Wiedner Hauptstraße 8-10, A - 1040 Vienna
Tel. +43-1-58801-5374, -5386, -5484, Fax: +43-1-5874211
Email: argesim@simserv.tuwien.ac.at

TABLE OF CONTENTS

Results of the EUROSIM Comparison "Lithium Cluster Dynamics"

F. Breitenecker[a] and I. Husinsky[b]

[a] Dept. Simulation Techniques, fbreiten@email.tuwien.ac.at
[b] Computing Services, husinsky@edvz.tuwien.ac.at
Technical University Vienna, Wiedner Hauptstraße 8-10, A - 1040 Vienna

This contribution summarizes the solutions of the EUROSIM Comparison on Simulation Software "Lithium Cluster Dynamics". The EUROSIM Software Comparisons (up to now eight) and the solutions sent in are published in the journal **EUROSIM Simulation News Europe** (**SNE**). Based on the results some developments and trends in continuous simulation software and related problems are briefly sketched.

1. THE EUROSIM COMPARISONS

EUROSIM, the Federation of European Simulation Societies, started in 1990 the publication of the journal **EUROSIM Simulation News Europe (SNE),** a newsletter distributed to all members of the European simulation societies under **EUROSIM's** umbrella and to people and institutions interested in simulation. **SNE** is also part of **Simulation Practice and Theory (SIMPRA),** the scientific journal of **EUROSIM**.
The idea of the journal **SNE** (circulation 2500; edited by F. Breitenecker and I. Husinsky, ARGE Simulation News (**ARGESIM**) , Technical University of Vienna, Austria; three issues per year) is to to dissemination information related to all aspects of modeling and simulation. The contents of **SNE** are news in simulation, simulation society information, industry news, calendar of events, essays on new developments, conference announcements, simulation in the European Community, introduction of simulation centers and comparison of simulation software, simulators and (parallel) simulation techniques.
The series on comparisons of simulation software has been very successful. Based on simple, easily comprehensible models the software comparisons compare special features of modeling and experimentation within simulation languages:

- modeling technique
- event handling
- submodel features
- numerical integration
- steady-state calculation
- frequency domain
- plot features
- parameter sweep

- postprocessing
- statistical features
- statistical processors
- control strategies
- optimization
- random numbers
- complex strategies
- animation, etc.

Seven Software Comparisons, four continuous ones and three discrete ones (a fourth discrete comparison is in preparation) have been set up. Furthermore, a second type of comparisons, the Parallel Comparison has been initiated.
The continuous comparisons are: Comparison 1 (C1; Lithium-Cluster Dynamics under Electron Bombardment, November 1990) deals with a stiff system; Comparison 3 (C3; Analysis of a Generalized Class-E Amplifier, July 1991) focusses on simulation of electronic circuits and eigenvalue analysis; Comparison 5 (C5; Two State Model, March) requires very high accuracy computation; Comparison 7 (C7; Constrained Pendulum, March 1993) deals with state events.
The discrete comparisons are: Comparison 2 (C2; Flexible Assembly System, March) gives insight into flexible structures of discrete simulators; Comparison 4 (C4; Dining Philosophers, November 1991) involves not only simulation but also different modeling techniques like Petri nets; Comparison 6 (C6; Emergency Department - Follow-up Treatment, November 1992) deals with complex control strategies; Comparison 8 (C8, locks on channels) will deal with variance reduction methods.
SNE 10 introduced a new type of comparison dealing with the benefits of distributed and parallel computation for simulation tasks. Three test examples have been chosen to investigate the types of parallelization techniques best suited to particular types of simulation tasks.

Up to now, 100 solutions have been sent in. Table 1 shows the number of solutions for the Software Comparisons as well as for the Parallel Comparison. The series will be continued.

	C1	C2	C3	C4	C5	C6	C7	CP
SNE 0	Def							
SNE 1	5	Def						
SNE 2	4	4	Def					
SNE 3	4	3	3	Def				
SNE 4	1	5	5	3	Def			
SNE 5	4	-	1	1	2			
SNE 6	-	2	-	2	1	Def		
SNE 7	1	2	1	2	-	1	Def	
SNE 8	-	1	-	-	-	1	3	
SNE 9	-	-	-	-	-	2	3	
SNE 10	1	2	-	-	-	2	2	Def / 1
SNE 11	2	2	1	-	1	-	-	2
SNE 12	1	-	1	-	-	-	2	3
SNE 13	-	-	-	-	-	-	3	1
SNE 14	3	-	1	-	-	-	2	-
Total	**26**	**21**	**13**	**8**	**4**	**6**	**15**	**7**

Table 1: EUROSIM Comparisons, publication of solutions

2. THE EUROSIM COMPARISON C1 "LITHIUM CLUSTER DYNAMICS"

EUROSIM comparison 1 (Lithium-Cluster Dynamics under Electron Bombardment) has been performed by 26 simulation languages or simulators. This comparison is based on a stiff third order system of ODE's describing the concentrations $f(t)$, $m(t)$, and $r(t)$ of molecule agglomerates (F- , M - and R- centers) of alkali halides under electron bombardment:

$$dr/dt = -d_r r + k_r m f$$
$$dm/dt = d_r r - d_m m + k_f f^2 - k_r m f$$
$$df/dt = d_r r + 2 d_m m - k_r m f - 2 k_f f^2 - l_f f + p$$
$$k_r = d_m = 1,\ k_f = d_r = 0.1,\ l_f = 1000$$
$$r(0) = 9.975,\ m(0) = 1.674,\ r(0) = 84.99$$

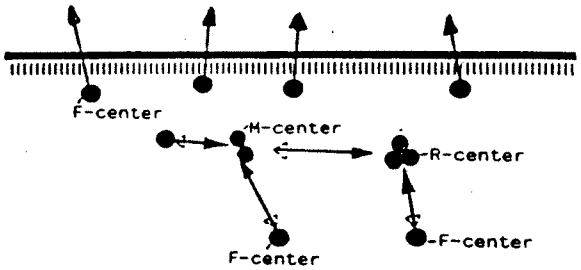


Fig. 1: Comparison1, physical background

The following three tasks had to be performed:

i) test and comparison of integration algorithms ($t \in [0, 10]$),
ii) parameter sweep of l_f (100, ... 10000) with log plots, and
iii) steady state calculation for $p = 0$ and $p = 10000$.

First it has to be noted that all simulation languages fulfilled the tasks with sufficient accuracy. Table 1 gives an overview about simulation languages and simulators, where solutions were sent in (column 1). The simulators can be divided roughly into three groups: equation oriented languages, (graphical) block-oriented languages, application-oriented languages. The table indicates these different modeling techniques (column 2). As some languages offer different modeling approaches, the one used in the solution sent in is marked with an asterisk. Special features and essential properties are remarked in column 3.

LANGUAGE	MODEL DESCRIPTION	REMARKS
ACSL	equations (ODEs)	CSSL-language with rich structure; 2 solutions
DESIRE	equations (ODEs)	combination with neural network simulation; interfaces to C and Turbo Pascal
DYNAST	equations (DAEs) (*) graphical blocks (sub models) port diagrams (graphical)	semi-symbolic analysis for linear systems; documentation environment based on AutoCAD or TeX for PC version
ESACAP	equations (DAEs) (*), nodes / branches, arbitrary expressions	"European Space Agency Circuit Analysis Program"; based on numerical algorithm for circuit analysis
ESL	equations (ODEs) (*) graphical blocks (sub models)	interpretative and compile mode; graphic postprocessor
EXTEND	graphical blocks	continuous and next event modeling; mainly Macintosh,
FSIMUL	graphical blocks (sub models)	"Control Engineering" - optimisation features
HYBSYS	blocks (elementary) (*) equations	"Hybrid Simulation System" (1980 TU - Wien) interpretative simulator; direct data base compilation;

Table 1, part 1: General features of simulation languages

LANGUAGE	MODEL DESCRIPTION	REMARKS
IDAS / SIMPLORER	graphical(ORCAD, ...) equations (Description Language) by dialog (Windows) (*)	specialized for electronic circuits and control problems; based on Windows
I Think	graphical blocks	modeling based on system dynamics; no slot to other modeling or programming languages
MATLAB	equations (MATLAB functions)	tool for mathematical and engineering calculations
MATRIXx	graphical blocks (*) matrix manipulation	interactive matrix-manipulation; using LINPACK and EISPACK
mosis	equations	"modular simulation system"; CSSL-type on C basis; features for parallelization on MIMD-systems;
NAP2	blocks (electronic circuits)	specialized for circuit simulation
POWERSIM	graphical blocks description	based on System Dynamics formulation
PROSIGN	equations (ODEs) graphical blocks (sub models) application-oriented components	"Process Design"; combination of modeling techniques; interfaces to C, Fortran, Modula2; variable number of input and output parameters
SABER	equations (ODEs)	specialized for analogue circuit simulation
SIL	equations (ODEs, DAEs)	simulation of discrete and continuous systems; free format
SIMNON	equations (ODEs) (*) macro function, sub models	simulation of discrete and continuous systems; real-time features; connecting systems; direct data base compilation
SIMULINK	graphical blocks (sub models) (*) equations (MATLAB functions)	based on MATLAB; special integration-algorithm Linsim; no limits for number of states and variables; 2 solutions
SIMUL_R	equations (ODEs) (*) bond graphs (graphical preprocessor) blocks (graphical preprocessor)	simulation of discrete and continuous systems; open system, based on C; runtime interpreter; combined simulation
STEM	equations (ODEs)	"Sim. Tool for Easy Modeling"; basis on Turbo Pascal
TUTSIM	graphical blocks, bond graphs equations (ODEs) (*)	"Twente University of Technology" (NL); simulation of discrete and continuous systems
XANALOG	graphical blocks (sub models)	sophisticated linearization, real-time features

Table 1, part 2: General features of simulation languages

3. RESULTS AND EVALUATION OF THE COMPARISON

Simulations show, that in the very beginning (in the interval [0, 5E-3]) fast transient dynamic occurs, while later on (in the interval [5E-3, 10]) the system is relatively smooth (fig.2, logarithmic axes).

Eigenvalue analysis of the linearized model results in three eigenvalues being negative real numbers.

At t = 0 the eigenvalues are -0.00898, -11.06, -1005.66, at t = 10 the values are -0.0978, -1.018, -1003.4 .

Dividing the absolute value of the biggest eigenvalue by the absolute value of the smallest eigenvalue results in a stiffness factor. At t = 0 this factor is approximately 120000, *at* t = 10 the factor is about 10000.

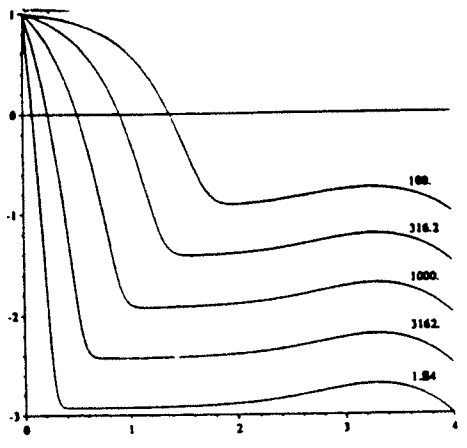


Fig.2: Results f(t), variation of l_f

Figure 3 shows this stiffness changing over the time (logarithmic scales): fast transients happen at the very beginning of the simulation, afterwards the system is relatively smooth.

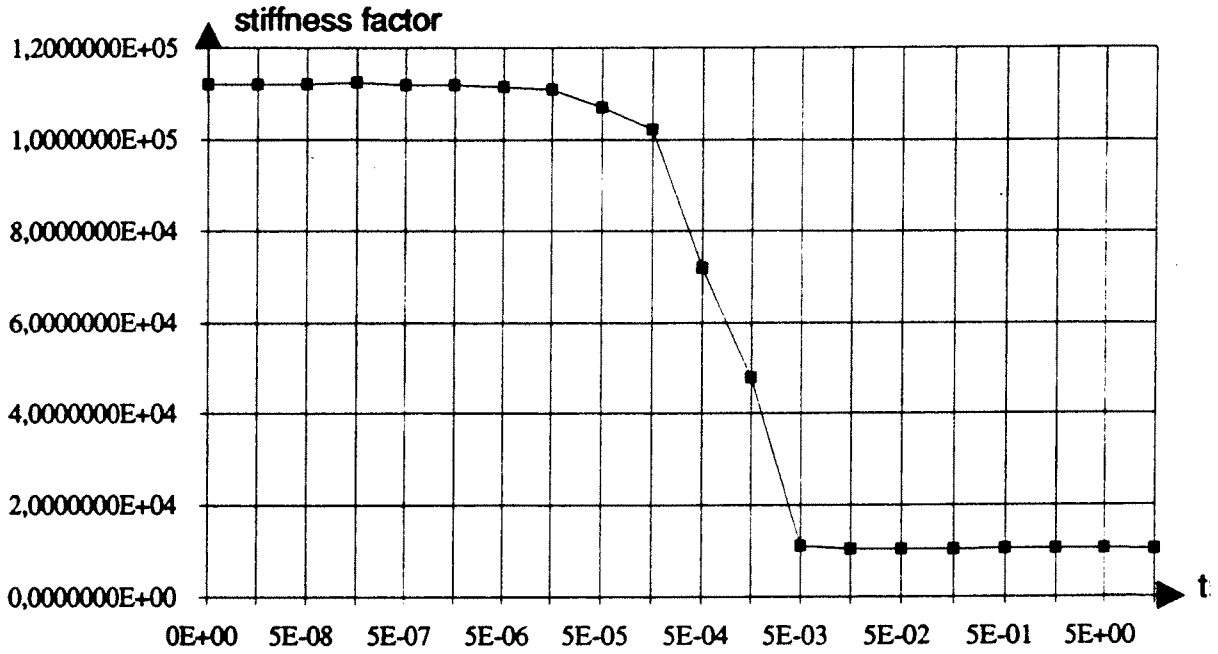


Fig.3: Stiffness of the system over time (logarithmic axes)

3.1 Task i): Test and comparison of integration algorithms

It is relatively difficult to compare the results of this task. Although most languages offer exact CPU-times for the different algorithms, these results suffer from side effects like I/O-time, straight-forward or tricky modeling, well tuned algorithm parameters (model-dependent!) or standard values, etc. Therefore, for the comparison of the algorithms the relation between the different algorithms is more significant than absolute CPU-times (normalized to Euler algorithm). Table 2, summarizing these results, is mostly restricted to three algorithms: Gear stiff algorithm (variable stepsize, variable order), Euler algorithm (fixed stepsize) and Runge- Kutta algorithm (RK4, mainly fixed stepsize), because these algorithms all work "well" (in case one or more of these algorithms are missing, preferably results of Runge-Kutta-Fehlberg and Adams-Moulton algorithm are given).

Table 2 generally shows that the Gear algorithm is the best one for this model because of the stiffness of the system. Unfortunately some reports do not indicate which order the Gear algorithm had to choose in order to fulfill the constraints on the relative or absolute errors, resp. Insight into these questions offers for instance ESACAP, which compares different BDF-algorithms (Backward Differential Formulas, the predecessors of the Gear algorithms) on the basis of number of steps, function evaluations, calculations of the Jacobian matrix, etc. Furthermore, the most efficient Gear algorithms or BDFs are offered by languages (DYNAST, ESACAP, SIL) using model description on basis of DAEs (Differential Algebraic Equations) - by reformulating the model in implicit form.

LANGUAGE	SNE-NR C1-NR	COMPUTER	ALGORITHM	STEPSIZE ACCURACY	COMPUTATION TIME OTHERS
ACSL	SNE-1 C1-3	PC 80287/12	Adams-Moulton	5.E-3 iss	1 (155.055 sec)
			Gear	5.E-3 iss	0.022
			RKF 4/5, vs	5.E-3 iss	0.355
ACSL	SNE-5 C1-17	Micro VAX/ Sun4	Euler	1.E-5 / 2.E-1 ss	1 (8.43 sec) / 0.056
			RK 4	1.E-5 / 2.E-1 ss	1.981 / 0.101
			Gear	1.E-8 ae, 1.E-5 iss	0.236 / 0.018
DESIRE	SNE-4 C1-	PC 80387/16	Gear	1.E-5 ae, 1.E-6 logiss	10 sec
		Sun 4c	Gear	1.E-5 ae, 1.E-6 logiss	1.7 sec
DYNAST	SNE-3 C1-12	PC 80387	Gear-Newton-Raphson	1.E-3 re, 1.E-5 iss	0.506
				1.E-6 ae, 1.E-5 iss	1 (4.45 sec)
ESACAP	SNE-1 C1-1	PC 80387	BDF 1o, vs	1.E-3 re/ 1.E-7 re	118ns,237f/10271ns,20547f
			BDF 2o, vs	1.E-3 re/ 1.E-7 re	53 ns,105f/ 316 ns, 632f
			BDF 3o, vs	1.E-3 re/ 1.E-7 re	51 ns,102f/185 ns,370 f
ESL	SNE-2 C1-8	PC 80387 SX/16	RK 4	1.E-3 ss	0.571
			Adams Bashforth	1.E-1 iss	1 (21 sec)
			Gear	1.E-1 iss	0.01
EXTEND	SNE-5 C1-15	Macintosh IIfx	Euler impr.	12000 ns / 10000 ns	1 (1 sec) / unstable
			Trapezoidal rule	30000 ns/ 20000 ns	2.3 / unstable
FSIMUL	SNE-1 C1-4	PC 80387 /25	AB 2o, vs	5.E-4 iss/ 1.E-3 iss	0.556 / unstable
			implicit Heun	5.E-4 ss/ 1.E-3 ss	0.973 / unstable
			RK4	5.E-4 iss/ 1.E-3 iss	1 (187 sec) / unstable
HYBSYS	SNE-2 C1-7	DECStation 3100/16	ABM	1.E-5 iss	1.983
			Euler	1.E-4 ss	1 (8.47 sec)
			RK 4	2.E-4 iss	1.099
IDAS	SNE-C1-25	Pentium 60mHz	Euler	minss=0.002	1 (8 sec)
			Trapezoidal	mss=0.01	1
I Think	SNE-5 C1-16	Macintosh IIfx	Euler	1.E-4 ss/ 1.E-3 ss	1 (420 sec) / unstable
			RK 2	1.E-4 ss/ 1.E-3 ss	1.286 / unstable
			RK 4	1.E-4 ss/ 1.E-3 ss	1.714 / unstable
MATLAB	SNE-3 C1-10	PC 80387 (PS/S80)	RKF 2/3	1.E-5 re	739 sec
			RKF 4/5	1.E-6 / 1.E-7 re	563 sec / 752 sec
MATRIXx	SNE-10 C1-19	PC 80486/33	Euler	1.E4 equ. time points	1 (90.3 sec)
			RK2 / RK4	1.E4 equ. time points	1.468 / 2.411
		Sun 4 /40	Euler	1.E4 equ. time points	1 (8.19 sec)
			RK2 / RK4	1.E4 equ. time points	1.442 / 2.322
mosis	SNE-12 C1-22	PC 486/33	Euler	1.0E-3 ss	1 (2.3 sec)
			RK4	1.0E-3ss /1.0E-4 ss	1.783 / 17.957
			Adams Moulton	1.0E-4 ss,1.0E-8 mae	1.122
			Stiff Alg.	1.0E-4 ss,1.0E-8 mae	0.039
NAP2	SNE-1 C1-2	PC 80387 Norton CI 25,6	mod. Gear, vs,vo	1.E-5 iss	4.56 sec
POWERSIM	SNE 14 C1-25	PC 80486/66	Euler	1.0E-3 ss	1 (32 s)
			RK4	vs, 1.0E-3 iss	1.2

Legend: ss ... stepsize; iss ... initial ss; log (i)ss ... logarithmic (i)ss; mss ... max. ss; re ... relative error; ae ... absolute error; ns ... number of steps; f ... function evaluations, vs ... variable ss; vo ... variable order; 4o ... 4th order; etc.; RK4 ... classical Runge-Kutta; RKF ... Runge-Kutta-Fehlberg; AB(M) ... Adams-Bashforth(-Moulton); BDF ... Backwards Differential Formulas

Table 2, part 1: Results of task i): test and comparison of integration algorithms

LANGUAGE	SNE-NR C1-NR	COMPUTER	ALGORITHM	STEPSIZE ACCURACY	COMPUTATION TIME OTHERS
PROSIGN	SNE-3 C1-13	not given	Simpson 2o, vs	1.E-3 mss	1 (470 sec)
			AB 4o, vs	2.5.E-3 mss	0.434
SABER	SNE-11 C1-20	Sun	Gear 1o/Gear 2o	vs	1 (0.75 sec)/ 0.44
		SPARC10/402	Gear 2o/Gear 2o	5.E-4 ss/1.E-3 ss	1 (47.3 sec)/ 0.448
			Trapezoidal rule	vs	0.016
SIL	SNE-2 C1-9	PC 80387	Stiff alg., vs, vo	1.E-2 re/ 1.E-4 re	0.231 / 0.351
				1.E-6 re/ 1.E-10 re	0.49/ 1 (11.43 sec)
SIMNON	SNE-12 C1-23	PC 80386/25	Euler	1.0E-3	1 (23 sec)
			RKF23	vs, 1.E-6 re	0.913
			RKF45	vs, 1.E-6 re	0.652
SIMNON	SNE-11 C1-21	PC 80386/40	Euler	1.0E-3	1 (31 sec)
			RKF23	vs, 1.E-6 re	0.39
			RKF45	vs, 1.E-6 re	0.264
		PC 80486/66	Euler	1.0E-3	1 (9.8 sec)
			RKF23	vs, 1.E-6 re	0.398
			RKF45	vs, 1.E-6 re	0.276
SIMULINK	SNE-3 C1-11	Sun 4	RK 5, vs	1.E-2 re,1.E0-.E-4 ss	1 (10.4 sec)
			Gear	1.E-2 re,1.E0-.E-4 ss	0.034
			Linsim	.E-2 re,1.E0-1E-4 ss	0.018
SIMUL_R	SNE-1 C1-5	not given	Euler	1.E-3 ss, 1.E-5 re	1 (not given)
			RK 4	2.E-3 ss, 1.E-5 re	1.9
			Euler implicit	1.E-1 ss, 1.E-3 re	0.22
STEM	SNE-5 C1-18	PC 80287/20	RKF 1/2o, vs	1.E-6 re, 1.E-3 ae	1 (18.84 sec)
			RKF 4/5o, vs	1.E-6 re, 1.E-3 ae	0.574
			Gear, vs	1.E-6 re, 1.E-3 ae	0.027
TUTSIM	SNE- C1-24	PC 80387/16	Euler	5.E-4 mss	1 (44 sec)
			AB	5.E-4 mss	1.114
XANALOG	SNE-2 C1-6	PC 80287 /16	RK 4	1.E-3 ss / 2.5.E-3 ss	2.744 / 88 sec
			Euler	1.E-3 ss / 2..E-3 ss	1 (82 sec) / unstable
			mod. Euler	1.E-3 ss / 2..E-3 ss	1.439 / unstable

Legend: ss ... stepsize; iss ... initial ss; log (i)ss ... logarithmic (i)ss; mss ... max. ss; re ... relative error; ae ... absolute error; ns ... number of steps; f ... function evaluations, vs ... variable ss; vo ... variable order; 4o ... 4th order; etc.; RK4 ... classical Runge-Kutta; RKF ... Runge-Kutta-Fehlberg; AB(M) ... Adams-Bashforth(-Moulton); BDF ... Backwards Differential Formulas

Table 2, part 2: Results of task i): test and comparison of integration algorithms

The classical RK4 algorithm works well, if an appropriate stepsize and an appropriate relative error is chosen, being approximately 10 times slower than the Gear algorithm. RKF algorithms (Runge-Kutta-Fehlberg) speed up the integration time using stepsize control.
It is known from theory that the Adams-Moulton and /or Adams-Bashforth-algorithms are not suitable for this kind of systems; but it is astonishing that they are really very slow.
Another astonishing phenomenon is the result of the Linsim algorithm of SIMULINK, which is twice faster than the classical Gear algorithm. This algorithm extracts the linear parts of the models and calculates the linear dynamics via power series, the nonlinear parts are integrated in the usual manner.

Three solutions sent in showed that it is worth thinking over a model before simulating it. The authors made use of the fact that fast transients happen only at the very beginning.
Consequently, the second ACSL solution choose exponentially spread sampling points, resulting also in related stepsize (also better suited for log plots).
The DESIRE solution and the first SIMNON solution performed this exponential time shift directly in the model equations (logarithmic time transformation). As a consequence, the integration algorithms became (much) faster, the system became nearly non-stiff.

3.1 Task ii): Parameter sweep and log plots

The second task should test whether a simulation language offers features for parameter sweeps. Table 3 summarises the results in column 2, where it is tried to distinguish between parameter loops in the model description and at run-time level. In case of graphical model description model frame and experimental frame are mixed, so that this distinction becomes difficult.
Furthermore, it turned out that the additional requirement of a logarithmic parameter sweep and logarithmic plot was no further challenge: if parameter loops are available, different increments can be used; if the parameter sweep has to be formulated in a "manual" way, the logarithmic sweep is also simple. The third column in table 3 therefore indicates only, whether logarithmic representations are supported directly ("standard") or not ("manual" transformation).

3.3 Task iii): Steady state calculation:

The third task should check which languages offer features for steady state calculation. The model is simple enough to calculate the steady states analytically, so all results could be compared with the exact values:

$$p = 10000: f_s = 10,\ m_s = 10,\ r_s = 1000$$

$$p = 0: \quad f_s = m_s = r_s = 0).$$

Languages with steady state finder (column 3 of table 3 ,"trim command, iteration") calculated the results for both cases with sufficient accuracy. Usually the iterative solution of the steady state equations started with the initial values for f, m and r.

Languages without a steady state finder ("longterm simulation") simulated over a long period stopping when derivatives are nearly zero (approx. at $t = 100$), getting as accurate results as the steady state finders.

LANGUAGE	PARAMETER VARIATION	LOG.	STEADY STATE CALC.
ACSL	manual variation at runtime	standard	trim command, iteration
DESIRE	parameter loop in model description	manual	not given
DYNAST	manual variation in model description	standard	long term simulation
ESACAP	parameter loop in model description	standard	long term simulation
ESL	parameter loop in model description	standard	trim command, iteration
EXTEND	manual variation in graphic model description	standard	long term simulation
FSIMUL	parameter loop in graphic model description	standard	long term simulation
HYBSYS	parameter loop at runtime	standard	trim command, iteration
IDAS	manual variation in model description	standard	long term simulation
I Think	manual variation in graphic model description	standard	long term simulation
MATLAB	parameter loop in model description	standard	trim command, iteration
MATRIXx	manual variation in model description	standard	trim command, iteration
mosis	parameter loop at runtime	standard	trim command, iteration
NAP 2	manual variation in model description	standard	long term simulation
POWERSIM	parameter loop in model desrc.(co-models)	manual	not given
PROSIGN	parameter loop in graphic model description	standard	trim command, iteration
SABER	parameter loop in model description	standard	trim command, iteration
SIL	parameter loop at runtime	manual	trim command, iteration
SIMNON	parameter loop at runtime	manual	long term simulation
SIMULINK	manual variation in graphic model description	standard	trim command, iteration
SIMUL_R	parameter loop at runtime	standard	trim command, iteration
STEM	manual variation in model description	manual	trim command, iteration
TUTSIM	parameter loop at runtime	standard	long term simulation
XANALOG	parameter loop in graphic model description	standard	trim command, iteration

Table 3: Results of tasks ii) and iii): Parameter sweep and steady state calculation

4. TRENDS AND DEVELOPMENTS

The results of this comparison also allows a view on developments and trends of simulation languages and simulators. In the following some trends are listed, but also the problems which may arise:

Developments:

- Implicit model descriptions
- Submodel features
- Graphical model descriptions
- Graphical preprocessors
- Sophisticated integration algorithms
- State event handling
- New methods (formula manipul.)
- Separation of model and experiment
- More powerful runtime interpreters
- Windows Implementations

Problems:

- Loss of input-output relations
- Conflicts with macro features
- Loss of segment structure
- Overhead in generated equations
- Overhead for about 80% of problems
- Dependent on modeling technique
- CSSL structure too weak
- Interpreters not powerful enough
- Documentation with model
- Loss of speed, esp. on PC

In general, it is interesting, that

- Big enterprises tend to develop their own language, which are marketed, too
- Universities and institutions develop also new languages, which partially are successfully marketed
- In continuous simulation on the one side CSSL standard - languages become a common denominator for modeling, on the other hand a block-oriented graphical description based on control technique is frequently used.

Comparison of Simulation Software

In the early 70's only a few simulation languages existed. But soon, together with the use of PCs, the number of languages increased rapidly. Looking at the catalogue of simulation software over the years the increase started exponentially, but now a limited growth can be observed.

Even for a specialist in simulation it is now difficult to overview all languages and their features. A lot of benchmarks have been developed, but they are quite complicated.

EUROSIM - Simulation News Europe now starts a series using another approach for comparison of simulation software. Based on simple, easily comprehensible models special features of modelling and experimentation within simulation languages, also with respect to an application area, shall be compared.

We invite all institutes and companies developing or distributing simulation software to participate in this comparison:

Please, simulate the model described and send a report to the editors in the following form:

- short description of the language
- model description (source code, diagram, ...)
- results of the tasks with experimentation comments
- approx. 1/2 page A4

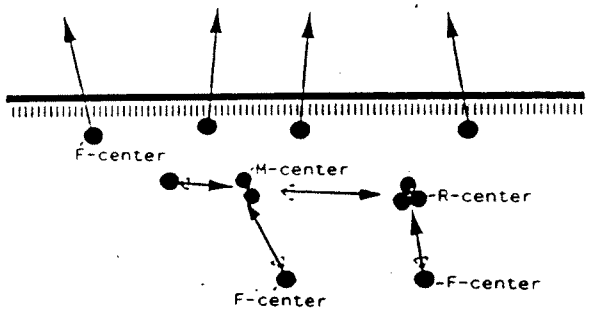
Reports will be published in EUROSIM - Simulation News Europe.

New comparisons will be prepared for the next issues. As it is difficult to find suitable "simple" models and relevant tasks we would like to ask you to contact the editors if you have an idea for a model to be compared in different simulation languages.

Comparison 1: Lithium-Cluster Dynamics under Electron Bombardment

The first model to be compared is taken from solid state physics. The special features to be compared are rate equations (application area), stiff systems (numerical integration), parameter sweep and steady-state calculation (experimentation).

The model describes formation and decay of defect ("F-centers") aggregates in alcali halides. The defects are produced by electron bombardment near the surface of the crystal and can either form aggregates or will evaporate if they reach the surface.



The variable f(t) denotes the concentration of F-centers, m(t) and r(t) respectively denote the concentration of aggregates consisting of two (M-center) or three F-centers (R-center). In principle the system can be easily extended taking into account formation of larger aggregates (n F-centers). The variable p(t) is the production term of F-centers due to electron bombardement (irridiation):

$$dr/dt = -d_r r + k_r mf$$
$$dm/dt = d_r r - d_m m + k_f f^2 - k_r mf$$
$$df/dt = d_r r + 2d_m m - k_r mf - 2k_f f^2 - l_f f + p$$

The parameter l_f measures the loss of F-centers at the surface. k_r and k_f are rate constants describing the formation of an M-center out of two F-centers, or the formation of an R-center out of an M-center and an F-center. The decay of an R-center into an M-center and an F-center is described by the rate constant d_r and the decay of an M-center into two F-centers by the rate constant d_m. Investigations are started after constant electron bombardment $p(t) = p_c = 10^4$ of approximately 10 s; the production term has to be set to zero ($p(t) = 0$), the initial values are:

$f(0) = 9.975$
$m(0) = 1.674$
$r(0) = 84{,}99$

The parameter values are:

$k_r = 1$
$k_f = 0.1$
$l_f = 1000$
$d_r = 0.1$
$d_m = 1$

The following tasks should be performed

a) simulation of the stiff system over [0,10] with indication of computing time depending on different integration algorithms

b) parameter variation of l_f from 1.0E2 to 1.0E4 and a plot of all $f(t;l_f)$, logarithmic steps preferred.

c) calculation of steady states during constant bombardment $p(t) = p_c = 1.0E4$) and without bombardment ($p(t) = 0$).

Comparison of Simulation Software

In the last issue (November 1990) EUROSIM-Simulation News Europe started a series on comparisons of simulation software.

This idea has become a great success: Based on simple, easily comprehensible models special features of modelling and experimentation within simulation languages, also with respect to an application area, are being compared.

In this issue the first results for "**Comparison 1: Lithium-Cluster Dynamics under Electron Bombardment**" are published. Here we would like to thank all the authors who solved the problem and sent in their contributions. Some of the reports contained complete descriptions of various experiments and different modelling approaches. Therefore we have excerpted abstracts from the reports received. Those who are interested in the full descriptions of the comparisons may write to the editors. If many people are interested we will consider to edit a special issue containing the full contributions. Reports on Comparison 1 will be continued to be published in the next issue, so please send in your contribution for simulation languages that have not yet been introduced.

Comparison 1 - Physical background

The "Lithium-Cluster Dynamics Model" describes the behaviour of defects under electron (and photon) bombardment of alkali halides. Among many others, one of the important consequences of these electronic defects is the desorption of surface atoms. The understanding and the control of such electronic desorption processes is essential for these materials when used in an environment of intensive radiation such as lasers.

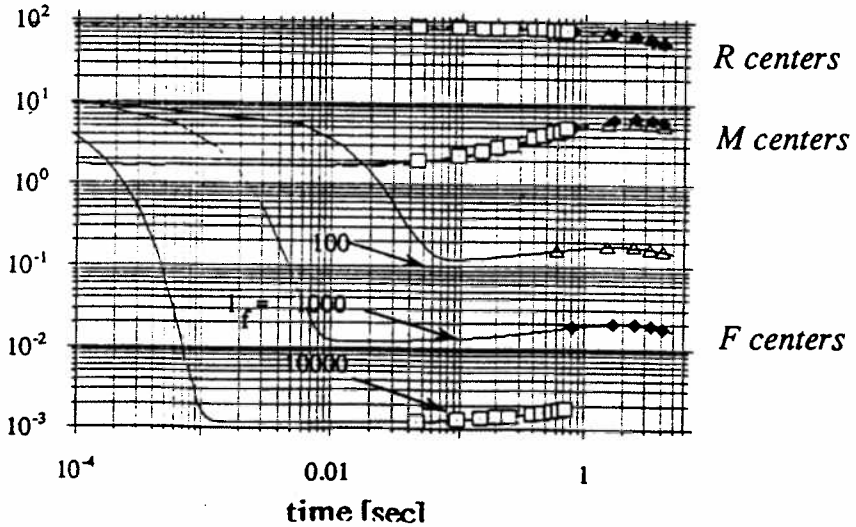
During exposure to radiation F centers are created in the surface and near surface bulk region of the crystal. The diffusion time of these F centers to the surface at elevated temperatures is very fast (msec timescale). It is a good assumption that every F center reaching the surface creates an neutral alkali atom which can desorb if the temperature is sufficiently high. In the experiments which are simulated by the model system discussed here the desorbing alkali atoms (Li) have been monitored with a quadrupol mass analyzer or via Laser Induced Fluorescence. (The temperature of the LiF crystal was 400°C to assure fast F center diffusion and evaporation of every Li atom created at the surface by a F center). Hence the amount of detected desorbed Li atoms is identical with number of F centers.

The essential experimental observation is that after irradiation (production term set to 0 in the equations) the amount of desorbed Li drops by one to two orders of magnitude but lasts for several tens of seconds beyond irradiation. Furthermore, provided the experimental para- meters are set accordingly, a maximum in the desorption yield has been observed several seconds after beam turn off. This result must be imaged by the F center behaviour. Because the F center diffusion is so fast, the experimental data imply that F centers must be "stored" in so called agglomerates which are formed from - and can then disintegrate into - F centers. In reality agglomerates with many constituents can form. For simplicity only those with two and three atoms (M and R centers) are included here (We have shown that a good quantitative description can be obtained considering at least F_9 centers, the qualitative behaviour can be already seen with R centers).

The experimental parameters in the present simulation represented by the k_r, l_f..... values and initial conditions have been chosen in such a way that the characteristic (experimentally observed) maximum in F center concentration is qualitatively simulated. In order to "see" the maximum, however, a logarithmic plot of the concentration axis is needed, because otherwise the prompt decay by more than one order of magnitude would mask the maximum.

The model has been simulated with Mathematica using the standard Runge-Kutta package on a Macintosh II Si with floating point accelerator.

Wolfgang Husinsky, Institut für Allgemeine Physik, Technische Universität Wien, Wiedner Hauptstraße 8-10, A - 1040 Wien, Austria



Comparison 1 - ESACAP

Simulation carried out by means of the simulation program ESACAP at ElektronikCentralen, Denmark:

ESACAP is a general purpose program for simulation of non-linear dynamic systems. The first version of ESACAP (ESA Circuit Analysis Program) was developed in 1979-80 for the European Space Agency (ESA) by ElektronikCentralen, Denmark.

Problems are formulated in terms of a structure (nodes/branches) and/or arbitrary expressions. Besides node potentials and branch-flow, a so-called auxiliary variable can be specified.

Differential equations may be introduced by means of the auxiliary variable. If one of the variables can be isolated on one side, the procedure is straightforward. Otherwise, a pseudo-explicit expression is formed.

For example:

$F(x, y, dx/dt, dy/dt) = 0, G(x, y, dx/dt, dy/dt) = 0$

becomes:

$x = x + F(x, y, dx/dt, dy/dt), y = y + G(x, y, dx/dt, dy/dt)$

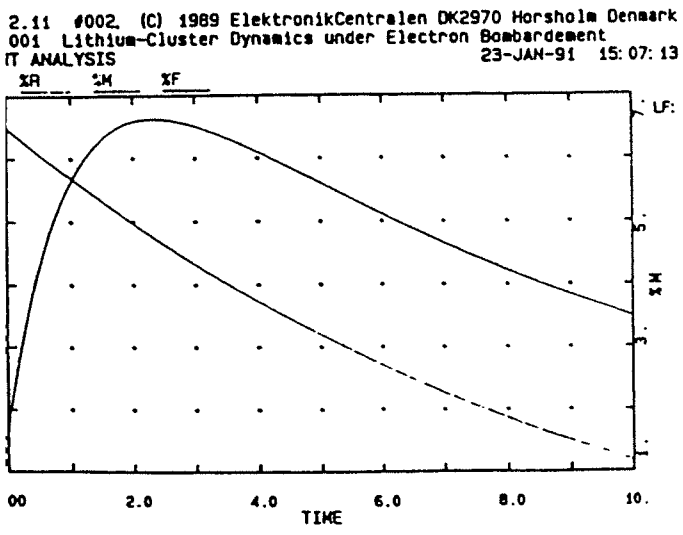
ESACAP employs numerical integration implemented as backward differential formulas of max order 6. Order and steplength are controlled by the relative truncation error. Non-linear systems are solved by a combined gradient/Newton method.

The ESACAP formulation of the actual problem is as follows:

```
KR=1; KF=.1; LF=1000; DR=.1; DM=1; P=1E4;
%R=%R-%R'-DR*%R+KR*%M*%F;
%M=%M-%M'+DR*%R-DM*%M+KF*%F*%F-
    KR*%M*%F;
%F=%F-%F'+DR*%R+2*DM*%M-KR*%M*%F-
    2*KF*%F*%F-LF*%F+P;
```

The prefix % indicates a system variable and '(apostrophe) stands for time-derivative.

The graphics presentation of the results from task a) is shown in the figure.

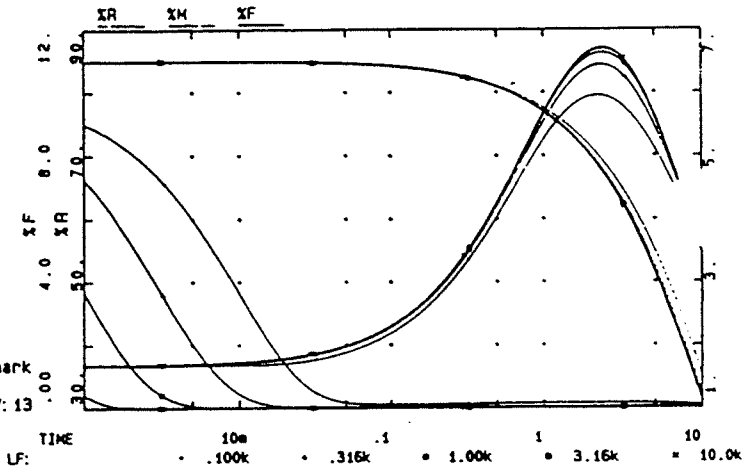


The task has been run on a PC under DOS with a 80387 math. co-processor. CPU time for the numerical calculations is masked by the time needed for I/O operations. An impression of the numerical effort may be gained from the following table in which the four numbers in each entry indicate: entry 1: number of integration steps, entry 2: number of equation factorizations, entry 3: number of substitutions (new right hand sides), entry 4: total operation count (number of double precision multiplications)

Order / Error	1	2	3	4	5	6
1e-3	118	59	53	51	51	51
	122	63	57	55	55	55
	237	118	105	102	102	102
	1321	669	600	581	581	581
1e-5	1043	204	124	105	106	106
	1051	212	132	113	114	114
	2091	410	250	214	216	216
	11528	2290	1410	1207	1218	1218
1e-7	10271	843	316	208	185	185
	10279	851	324	216	193	193
	20547	1689	632	416	370	370
	113036	9322	3516	2328	2075	2075

The next figure shows the results from task b). The graphic shows logarithmic time and parameter steps. The experimentation commands for the parameter sweep are:

```
$INIT: %F=9.975; %M=1.674; %R=84.99; END;

$PARAMETERS:
TIME=0,10; HFIRST=5E-5; ERROR=1E-7; MAXORD=6;
SWEEP(LF=1E2,1E4,LOG:2); END

$PLOT:
X(.001,10,LOG:50)=TIME;
Y(AUT)=%R!;  Y(AUT)=%M!;  Y(AUT)=%F!;  END;
```



The steady state solution during constant bombardment for different values of p is computed by the following experimentation commands (in the time domain):

```
$PARAMETERS: ERROR=1E-7; SWEEP(P=0,1E4,1E2); END;

$PLOT:
X=P;     Y(AUT)=%R!;  Y(AUT)=%M!;  Y(AUT)=%F!;  END;
```

Paul Stangerup, ElektronikCentralen, Venlighedsvej 4, DK-2970 Horsholm, Denmark. Tel: +45 42 86 77 22. Fax: +45 42 86 58 98

C O M P A R I S O N O F S I M U L A T I O N S O F T W A R E

Comparison 1: Lithium-Cluster Dynamics under Electron Bombardement

Simulation carried out by means of the simulation program ESACAP at ElektronikCentralen, Denmark:

ESACAP is a general purpose program for simulation of non-linear dynamic systems. The first version of ESACAP (ESA Circuit Analysis Program) was developed in 1979-80 for the European Space Agency (ESA) by Elektronik-Centralen, Denmark.

Problems are formulated in terms of a structure (nodes/branches) and/or arbitrary arithmetic expressions. Besides node potentials and branch-flow, a so-called auxiliary variable can be specified.

Differential equations may be introduced by means of the auxiliary variable. If one of the variables can be isolated on one side, the procedure is straight-forward. Otherwise, a pseudo-explicit expression is formed.

For example:
$$F(x, y, dx/dt, dy/dt) = 0$$
$$G(x, y, dx/dt, dy/dt) = 0$$

becomes:
$$x = x + F(x, y, dx/dy, dy/dt)$$
$$y = y + G(x, y, dx/dt, dy/dt)$$

ESACAP employs numerical integration implemented as backward differentiation formulas of max order 6. Order and steplength are controlled by the relative truncation error. Non-linear systems are solved by a combined gradient/Newton method.

The ESACAP formulation of the actual problem is as follows

```
KR=1; KF=.1; LF=1000; DR=.1; DM=1; P=1E4;

%R=%R-%R'-DR*%R+KR*%M*%F;
%M=%M-%M'+DR*%R-DM*%M+KF*%F*%F-KR*%M*%F;
%F=%F-%F'+DR*%R+2*DM*%M-KR*%M*%F-2*KF*%F*%F-LF*%F+P;
```

The prefix % indicates a system variable and ' (apostrophe) stands for time-derivative.

The grapics presentation of the results from task a) is shown in fig.1.

The task has been run on a PC under DOS with a 80387 math. co-processor. CPU time for the numerical calculations is masked by the time needed for I/O operations. An impression of the numerical effort may be gained from table I in which the four numbers in each entry indicate:

Entry 1. Number of integration steps
Entry 2. Number of equation factorizations
Entry 3. Number of substitutions (new right hand sides)
Entry 4. Total operation count (number of double precision multiplications)

TABLE I

Order / Error	1	2	3	4	5	6
1e-3	118	59	53	51	51	51
	122	63	57	55	55	55
	237	118	105	102	102	102
	1321	669	600	581	581	581
1e-5	1043	204	124	105	106	106
	1051	212	132	113	114	114
	2091	410	250	214	216	216
	11528	2290	1410	1207	1218	1218
1e-7	10271	843	316	208	185	185
	10279	851	324	216	193	193
	20547	1689	632	416	370	370
	113036	9322	3516	2328	2075	2075

In ESACAP, the user can specify various degrees of non-linearities thereby controlling how often the Jacobian is updated. Table II shows the influence of specifying the system as nearly linear and as strongly non-linear. When compared with the default specification, it is seen that the number of factorizations can be dramatically reduced. However, the gain is nearly lost by the greater number of integration steps.

TABLE II

Error: 1e-5
Order: 3

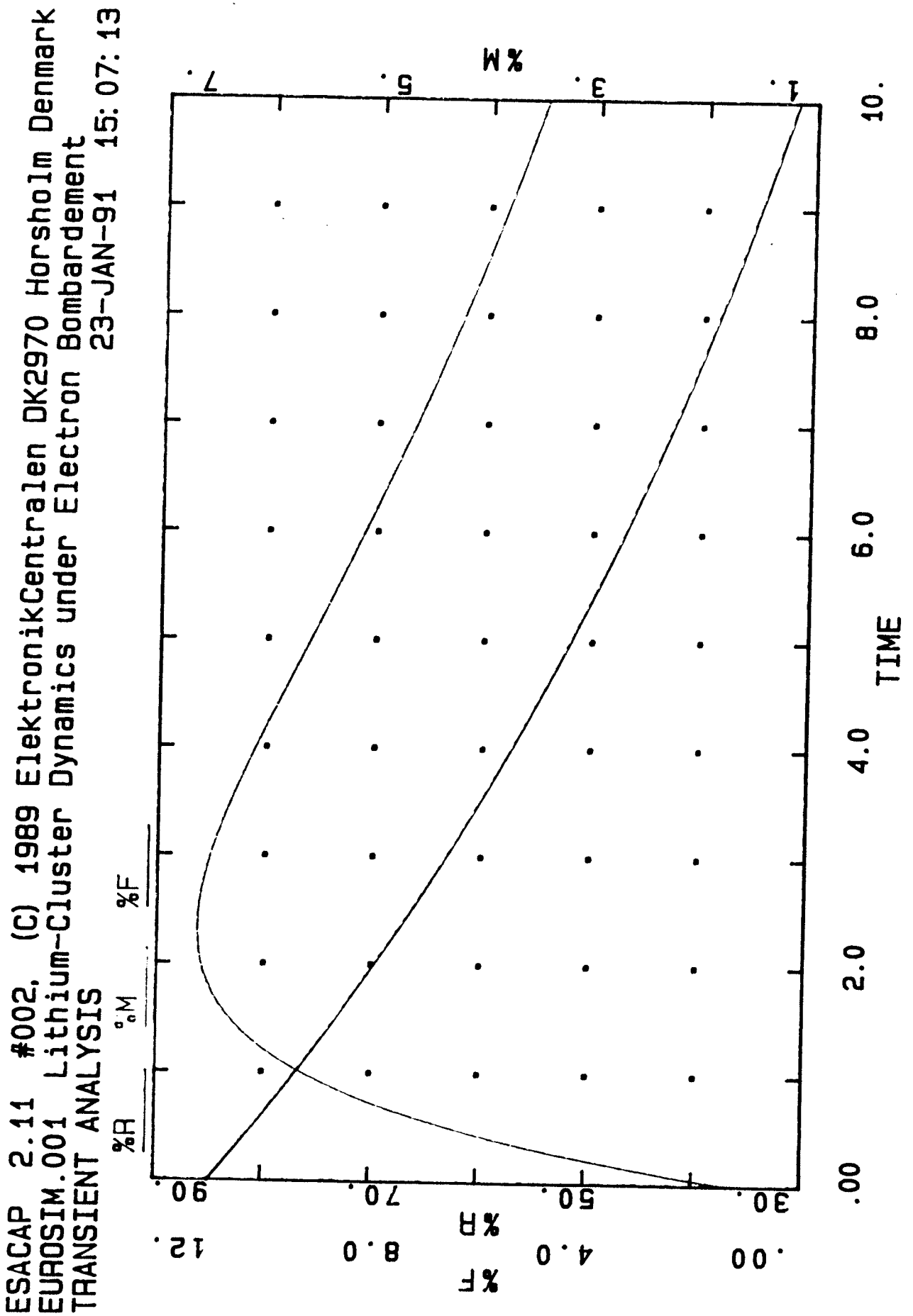
Nearly linear	Strongly linear
211	124
23	250
427	250
1396	2000

Fig.2 shows the results from task b). The graphics shows logarithmic time and parameter steps

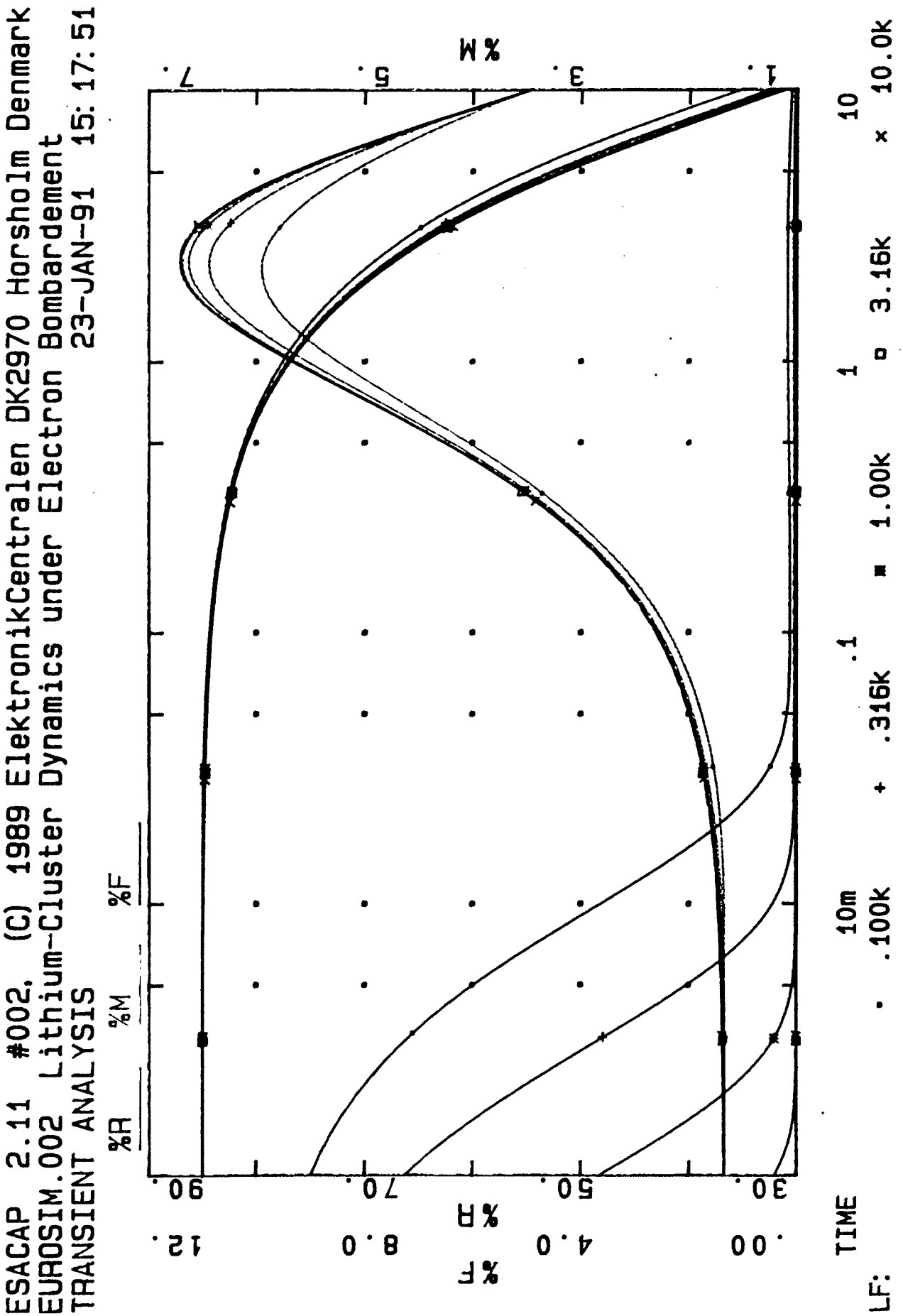
Fig.3 shows the results from task c). The effect of constant electron bombardment is shown for various values of p.

Fig.4 shows a simulation over 20 secs. The bombardment is stopped after 10 sec.

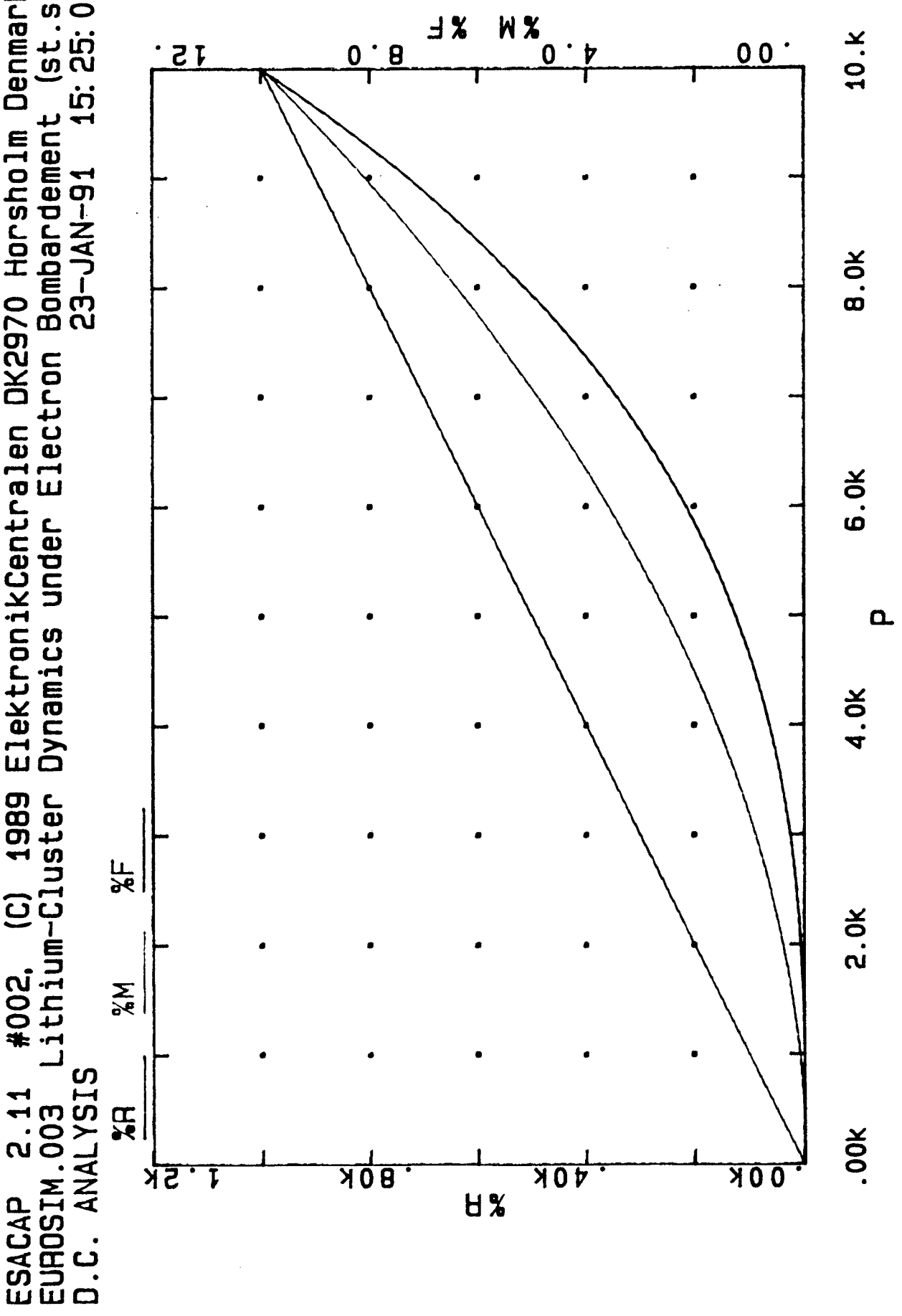
```
EUROSIM.001  Lithium-Cluster Dynamics under Electron Bombardement

# This ESACAP example shows the formulation and simulation of a dynamic
# system representing the concentration vs. time of various aggregates
# in alcali halides. For details, please refer to:

# Ref: Comparison of software. Comparison 1: Lithium-Cluster Dynamics
#      under Electron Bombardement.
#      EUROSIM Simulation News Europe. Nov.1990. Page 25


$$DES

$NET:

KR=1; KF=.1; LF=1000; DR=.1; DM=1;                          # Specification of
                                                            # parameters
P=0;

%R=%R-%R'-DR*%R+KR*%M*%F;                                   # Differential
%M=%M-%M'+DR*%R-DM*%M+KF*%F*%F-KR*%M*%F;                    # equations trans-
%F=%F-%F'+DR*%R+2*DM*%M-KR*%M*%F-2*KF*%F*%F-LF*%F+P;        # formed to pseudo
                                                            # explicit expres-
END;                                                        # sions


$$TRANSIENT

$INIT: %F=9.975; %M=1.674; %R=84.99; END;                   # Start vector

$PARAMETERS:                                                # Analysis para-
TIME=0,10; HFIRST=5E-5; ERROR=1E-7; MAXORD=6; END;          # meters

$PLOT:                                                      # Desired outputs
X=TIME;  Y(AUT)=%R!;  Y(AUT)=%M!;  Y(AUT)=%F!;  END;        # for graphics


$STOP
```


ESACAP 2.11 #002, (C) 1989 ElektronikCentralen DK2970 Horsholm Denmark
EUROSIM.001 Lithium-Cluster Dynamics under Electron Bombardement
TRANSIENT ANALYSIS
23-JAN-91 15:07:13
%M
%F
%R
%F
%R
%M
TIME
.00
2.0
4.0
6.0
8.0
10.
.00
4.0
8.0
12.
30.
50.
70.
90.
1.
3.
5.
7.

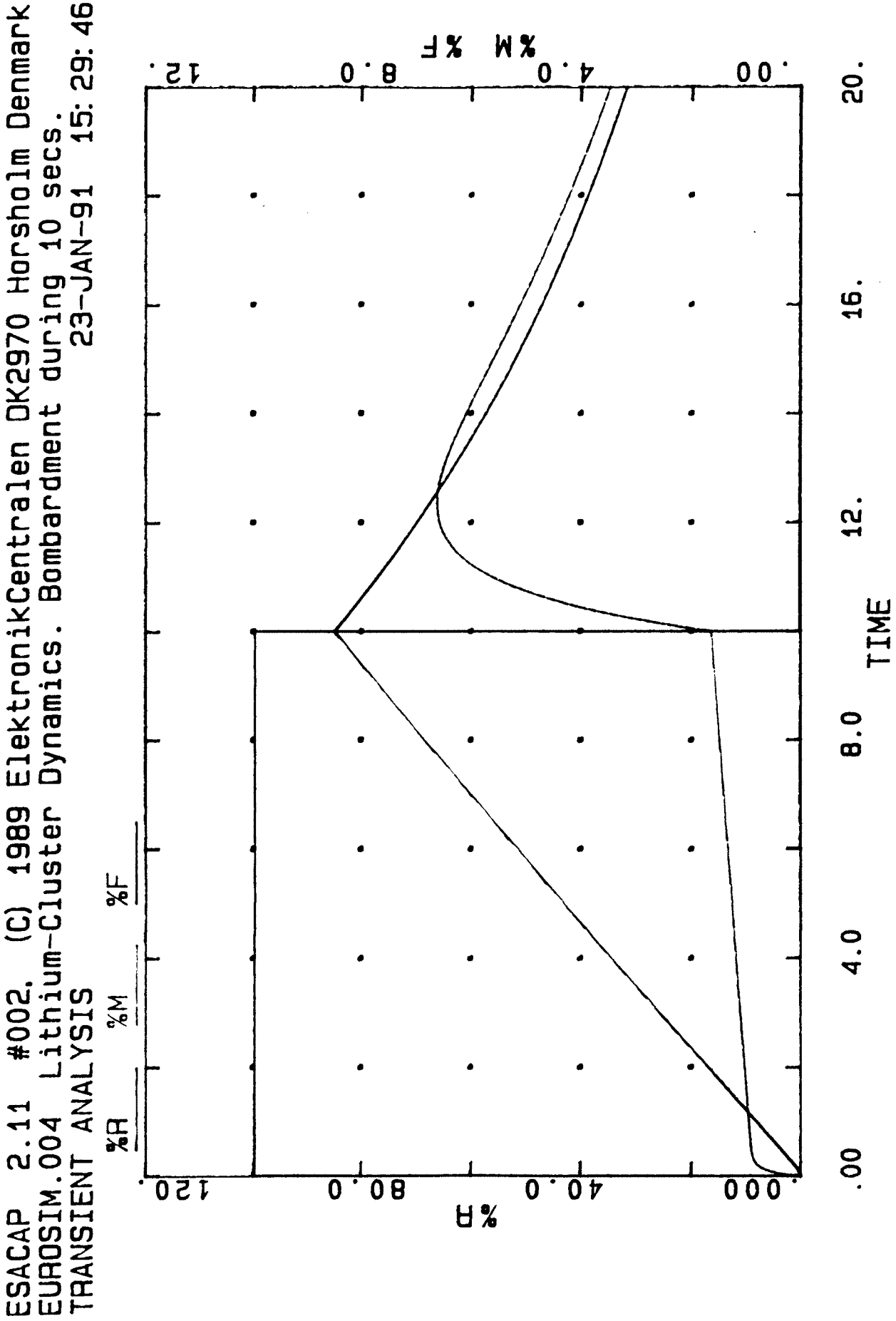
```
EUROSIM.002  Lithium-Cluster Dynamics under Electron Bombardement

# This ESACAP example shows the formulation and simulation of a dynamic
# system representing the concentration vs. time of various aggregates
# in alcali halides. For details, please refer to:

# Ref: Comparison of software. Comparison 1: Lithium-Cluster Dynamics
#      under Electron Bombardement.
#      EUROSIM Simulation News Europe. Nov.1990. Page 25


$$DES

$NET:

KR=1; KF=.1; LF=1000; DR=.1; DM=1;                          # Specification of
                                                            # parameters
P=0;

%R=%R-%R'-DR*%R+KR*%M*%F;                                   # Differential
%M=%M-%M'+DR*%R-DM*%M+KF*%F*%F-KR*%M*%F;                    # equations trans-
%F=%F-%F'+DR*%R+2*DM*%M-KR*%M*%F-2*KF*%F*%F-LF*%F+P;        # formed to pseudo
                                                            # explicit expres-
END;                                                        # sions


# In this example, the parameter LF is stepped betwen 1e2 and 1e4 in 5
# logarithmic steps.
# Graphics outputs have been changed to logarithmic scale as well

$$TRANSIENT

$INIT: %F=9.975; %M=1.674; %R=84.99; END;                   # Start vector

$PARAMETERS:                                                # Analysis para-
TIME=0,10; HFIRST=5E-5; ERROR=1E-7; MAXORD=6;               # meters
SWEEP(LF=1E2,1E4,LOG:2); END                                # Stepped LF value

$PLOT:                                                      # Desired outputs,
X(.001,10,LOG:50)=TIME;                                     # log scale
Y(AUT)=%R!;  Y(AUT)=%M!;  Y(AUT)=%F!;  END;                 # for graphics


$$STOP
```


ESACAP 2.11 #002. (C) 1989 ElektronikCentralen DK2970 Horsholm Denmark
EUROSIM.002 Lithium-Cluster Dynamics under Electron Bombardement
TRANSIENT ANALYSIS 23-JAN-91 15:17:51
%R %M %F
%M
%F
%R
TIME
LF:
10m .1 1 10
. .100k + .316k ▪ 1.00k ▫ 3.16k × 10.0k

```
EUROSIM.003  Lithium-Cluster Dynamics under Electron Bombardement (st.s

# This ESACAP example shows the formulation and simulation of a dynamic
# system representing the concentration vs. time of various aggregates
# in alcali halides. For details, please refer to:

# Ref: Comparison of software. Comparison 1: Lithium-Cluster Dynamics
#      under Electron Bombardement.
#      EUROSIM Simulation News Europe. Nov.1990. Page 25


$$DES

$NET:

KR=1; KF=.1; LF=1000; DR=.1; DM=1;                          # Specification o
                                                            # parameters
P=0;

%R=%R-%R'-DR*%R+KR*%M*%F;                                   # Differential
%M=%M-%M'+DR*%R-DM*%M+KF*%F*%F-KR*%M*%F;                    # equations trans
%F=%F-%F'+DR*%R+2*DM*%M-KR*%M*%F-2*KF*%F*%F-LF*%F+P;        # formed to pseud
                                                            # explicit expres
END;                                                        # sions


# In this example, steady state solution during constant bombardment is
# computed for varying values of P

$$D.C


$PARAMETERS: ERROR=1E-7; SWEEP(P=0,1E4,1E2); END;           # Analysis parame
                                                            # meters. The val
                                                            # of P is swepped

$PLOT:                                                      # Desired outputs
X=P;     Y(AUT)=%R!;  Y(AUT)=%M!;  Y(AUT)=%F!;  END;        # for graphics


$$STOP
```


ESACAP 2.11 #002, (C) 1989 ElektronikCentralen DK2970 Horsholm Denmark
EUROSIM.003 Lithium-Cluster Dynamics under Electron Bombardement (st.state)
D.C. ANALYSIS
23-JAN-91 15:25:03
%R
%M
%F
%R
1.2k
800
400
000
%M %F
12.
8.0
4.0
00.
10.k
8.0k
6.0k
4.0k
2.0k
.00k
P

```
EUROSIM.004  Lithium-Cluster Dynamics. Bombardment during 10 secs.

# This ESACAP example shows the formulation and simulation of a dynamic
# system representing the concentration vs. time of various aggregates
# in alcali halides. For details, please refer to:

# Ref: Comparison of software. Comparison 1: Lithium-Cluster Dynamics
#      under Electron Bombardement.
#      EUROSIM Simulation News Europe. Nov.1990. Page 25


# In this example, the simulation is carried out over 20 secs. The cons-
# tant bombardment is stopped after 10 secs. Initialization is the zero-
# vector.


$$DES

$NET:

KR=1; KF=.1; LF=1000; DR=.1; DM=1;                          # Specification of
                                                            # parameters

IF(TIME.LT.10) THEN                                         # Stop bombardment
 P=1E4;                                                     # after 10 secs
ELSE
 P=0;
ENDIF;

%R=%R-%R'-DR*%R+KR*%M*%F;                                   # Differential
%M=%M-%M'+DR*%R-DM*%M+KF*%F*%F-KR*%M*%F;                    # equations trans-
%F=%F-%F'+DR*%R+2*DM*%M-KR*%M*%F-2*KF*%F*%F-LF*%F+P;        # formed to pseudo
                                                            # explicit expres-
END;                                                        # sions


$$TRANSIENT

$PARAMETERS:                                                # Analysis para-
TIME=0,20; HFIRST=5E-5; ERROR=1E-7; MAXORD=6; END;          # meters

$PLOT:                                                      # Desired outputs
X=TIME;  Y(AUT)=%R!;  Y(AUT)=%M!;  Y(AUT)=%F!;  END;        # for graphics


$$STOP
```


ESACAP 2.11 #002. (C) 1989 ElektronikCentralen DK2970 Horsholm Denmark
EUROSIM.004 Lithium-Cluster Dynamics. Bombardment during 10 secs.
TRANSIENT ANALYSIS 23-JAN-91 15:29:46
%R
%M
%F
%R
%M %F
TIME
.00
4.0
8.0
12.
16.
20.
0.00
40.0
80.0
120.
0.00
4.0
8.0
12.

Comparison 1 - NAP2

Simulation Language

ANP3 & NAP2 - A package for circuits and system simulation.

An old idea: If you set up your differential equations and algebraic equations on an ideal analog computer then you may use an electronic circuit analysis program for the simulation.

The ideal integrator is modelled as a capacitor loaded current source. The voltage of the capacitor is the time integral of the current.

For "Comparison 1: Lithium-Cluster Dynamics under Electron Bombardment" the following input file for a general purpose electrical and electronic circuit analysis program is produced (model and experiment description):

```
*circuit; *list 2, 9;                    : file EUROSIM1.NAP;
:
*: Comparison 1: Lithium-Cluster Dynamics under Electron Bombardement >
: ref. EUROSIM - Simulation News Europe, pg.25, Number 0, November 1990
:
:integrating capacitors;
:
          cr 1 0 1 : vcr = v1 = r(t), R-center conc.;
          cm 2 0 1 : vcm = v2 = m(t), M-center conc.;
          cf 3 0 1 : vcf = v3 = f(t), F-center conc.;
:
: dr/dt = -dr*r + kr*m*f;
:
          irr  0 1 -0.1     vcr : dr = +0.1;
          irmf 0 1 +1.0*vcm vcf : kr = +1.0;
:
: dm/dt = +dr*r - dm*m + kf*(f**2) - kr*m*f;
:
          imr  0 2 +0.1     vcr : dr = +0.1;
          imm  0 2 -1.0     vcm : dm = +1.0;
          imf  0 2 +0.1*vcf vcf : kf = +0.1;
          immf 0 2 -1.0*vcm vcf : kr = +1.0;
:
: df/dt = dr*r + 2*dm*m - kr*m*f - 2*kf*(f**2) - lf*f + p
:
          ifr  0 3 +0.1     vcr : dr = +0.1;
          ifm  0 3 +2.0     vcm : dm = +1.0;
          ifmf 0 3 -1.0*vcm vcf : kr = +1.0;
          ifff 0 3 -0.2*vcf vcf : kf = +0.1;
lf=1.0e3;
.lf=1.0e2 : redefine lf;
          iff  0 3 -1.0*lf  vcf : lf = 1000;
:
: ebomb /tab2/ 0 1,  10  1,   10  0,   20  0;
:
:          gp   0 3  0    j=1e+4*ebomb(time);
:
  *modify  v1=84.99,   v2=1.674,  v3=9.975  : initial condition;
:            r(0)        m(0)       f(0)
  *time  0  10                 : variable order variable step integration;
  *tr vnall *plot(+50) v1 v2 v3  >     : linear time scale
            *plot(-50) v1 v2 v3  >     : logarithmic time scale
            *plot(+50) control.st          control.or 0 10  >
            *plot(-50) control.st          control.or 0 10     *probe ;
                     integration step   intgr. method order

  *run  hold cycle=500  minstep=1e-20  step=1n
: *end
:
: variation of parameter lf
: ------------------------
  *modify vnall=0, iall=0                  : reset solution   :
  *modify  v1=84.99,   v2=1.674,  v3=9.975  : initial condition;
  .lf=2.0e2
  *run  hold cycle=500  minstep=1e-20  step=1n
: ------------------------
```

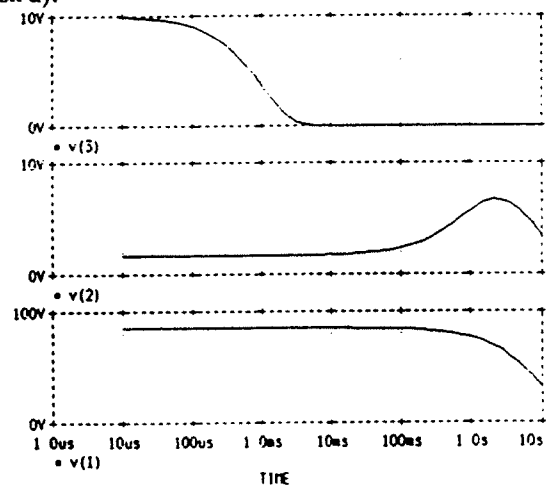
Please observe that it is not necessary to draw the equivalent circuit scheme. The integrating capacitors are given values 1 and placed between the reference node 0 and the nodes 1, 2 and 3. The coefficients of the differential equations are modelled as controlled current sources: ix <from-node> <to-node> <value> <control>.

The actual electronic circuit analysis program used (NAP2) is based on the extended node equations formulation. The integration method used is a modified Gear method with variable order variable step integration. The size of the program is: 256.143 kbytes. The computer used is IBM AT compatible. Operating system: DOS 3.30. Main processor: Intel 80386, Co-processor: Intel 80287. Norton computing index relative to IBM/XT: 25.6. Disk index: 3.4.

The following table summarizes the integration effort of the stiff system over [0,10].

```
initial integration-step = minstep = 10*1e-6 = 10usec
-----------------------------------------------------

final time ..................................    10.00 s
nr. of integrationsteps .....................    54
nr. of iterations ...........................   125
nr. of rejected steps .......................     1
max. nr. of iterations pr. integration step ..    25
nr. of NO CONVERGENCE .......................     0
total cpu-time consumption ..................     4.56 s
```

The figure shows a simple simulation of the system in the interval [0, 10] sec with the given initial conditions (task a).



The parameter sweep (task b) is formulated within the model description, the Gear integration method works with sufficient accuracy for all values of l_f.

Steady state calculation is performed by time domain computation over [0,1000] with following experimentation description and results (l_f = 1000, p = 10000:)

```
*MODIFY  V1=0,    V2=0,    V3=0 : INITIAL CONDITION;
         r(0)     m(0)     f(0)

initial integration-step = 1nsec
minimum integration-step = 1e-20
--------------------------------

final time ..................................... 1000.00 sec
max. nr. of iterations pr. integration step ..    25
total cpu-time consumption ....................    88.10 sec

nr. of integrationsteps ....................... 2896
nr. of iterations ............................. 1.83e+4
nr. of rejected steps ..........................  985
nr. of NO CONVERGENCE ..........................    3
solution at final time
                        1    9.998667D+02  r(1000)
                        2    9.998793D+00  m(1000)
                        3    9.999996D+00  f(1000)
```

Erik Lindberg, Institute of Circuit Theory and Telecommunication, 343 Technical University of Denmark, DK - 2800 Lyngby. Tel: + 45 45 93 12 22 3650. Fax: + 45 45 93 03 55

A contribution to the Comparison of Simulation Software

by

Erik Lindberg
Institute of Circuit Theory and Telecommunication
343 Technical University of Denmark
DK-2800 Lyngby, Denmark.

"Comparison 1: Lithium-Cluster Dynamics under Electron Bombardment"

Simulation language: Electrical circuit analogy.

An old idea:

If you set up your differential equations and algebraic equations on an ideal analog computer then you may use an electronic circuit analysis program for the simulation.

The ideal integrator is modelled as a capacitor loaded current source. The voltage of the capacitor is the time integral of the current.

For "Comparison 1: Lithium-Cluster Dynamics under Electron Bombardment" the following input file for a general purpose electrical and electronic circuit analysis program is produced. Please observe that it is not necesessary to draw the equivalent circuit scheme. The integrating capacitors are given values 1 and placed between the reference node 0 and the nodes 1, 2 and 3. The coefficients of the differential equations are modelled as controlled current sources: ix <from-node> <to-node> <value> <control>.

```
circuit; *list 2, 9;                    : file EUROSIM1.NAP;

: Comparison 1: Lithium-Cluster Dynamics under Electron Bombardement >
 ref. EUROSIM - Simulation News Europe, pg.25, Number 0, November 1990

integrating capacitors;

        cr 1 0 1 : vcr = v1 = r(t), R-center conc.;
        cm 2 0 1 : vcm = v2 = m(t), M-center conc.;
        cf 3 0 1 : vcf = v3 = f(t), F-center conc.;

 dr/dt = -dr*r + kr*m*f;

        irr  0 1 -0.1     vcr : dr = +0.1;
        irmf 0 1 +1.0*vcm vcf : kr = +1.0;
```

```
:
: dm/dt = +dr*r - dm*m + kf*(f**2) - kr*m*f;
:
          imr  0 2 +0.1      vcr : dr = +0.1;
          imm  0 2 -1.0      vcm : dm = +1.0;
          imf  0 2 +0.1*vcf  vcf : kf = +0.1;
          immf 0 2 -1.0*vcm  vcf : kr = +1.0;
:
: df/dt = dr*r + 2*dm*m - kr*m*f - 2*kf*(f**2) - lf*f + p
:
          ifr  0 3 +0.1      vcr : dr = +0.1;
          ifm  0 3 +2.0      vcm : dm = +1.0;
          ifmf 0 3 -1.0*vcm  vcf : kr = +1.0;
          ifff 0 3 -0.2*vcf  vcf : kf = +0.1;
lf=1.0e3;
.lf=1.0e2 : redefine lf;
          iff  0 3 -1.0*lf   vcf : lf = 1000;
:
: ebomb /tab2/ 0 1,  10  1,   10  0,   20  0;
:
:           gp   0 3  0   j=1e+4*ebomb(time);
:
  *modify  v1=84.99,   v2=1.674,  v3=9.975  : initial condition;
:            r(0)        m(0)       f(0)
  *time  0  10                 : variable order variable step integration;
  *tr vnall *plot(+50) v1 v2 v3  >    : linear time scale
            *plot(-50) v1 v2 v3  >    : logarithmic time scale
            *plot(+50) control.st        control.or 0 10  >
            *plot(-50) control.st        control.or 0 10     *probe ;
:                    integration step   intgr. method order
:
  *run  hold cycle=500  minstep=1e-20  step=1n
: *end
:
: variation of parameter lf
: -------------------------
  *modify vnall=0, iall=0                    : reset solution   ;
  *modify  v1=84.99,   v2=1.674,  v3=9.975  : initial condition;
  .lf=2.0e2
  *run  hold cycle=500  minstep=1e-20  step=1n
: -------------------------

                                lines deleted

: -------------------------
  *modify vnall=0, iall=0                    : reset solution   ;
  *modify  v1=84.99,   v2=1.674,  v3=9.975  : initial condition;
  .lf=1.0e4
  *run       cycle=500  minstep=1e-20  step=1n
: -------------------------
  *end
```

The actual electronic circuit analysis program used (NAP2) is based on the extended node equations formulation. The integration method used is a modified Gear method with variable order variable step integration. The size of the program is: 256.143 kbytes. The computer used is IBM AT compatible. Operating system: DOS 3.30. Main processor: Intel 80386. Co-processor: Intel 80287. Norton computing index relative to IBM/XT: 25.6. Disk index: 3.4.

```
================================================================
Run statistics:
---------------
All simulations are performed with a relative convergence criteria
of 1e-6.
================================================================
Task:                a) Simulation of the stiff system over [0,10]
-----                ------------------------------------------

initial integration-step = minstep = 10*1e-6 = 10usec
-----------------------------------------------------

final time ..................................    10.00 sec
nr. of integrationsteps .....................    54
nr. of iterations ...........................   125
nr. of rejected steps .......................     1
max. nr. of iterations pr. integration step ..    25
nr. of NO CONVERGENCE .......................     0
total cpu-time consumption ..................     4.56 sec

order             0     1     2     3     4     5     6
ORDER COUNT       1     4    12    11    11     8     6

order 0 = Forward Euler, order 1, 2, ... 6 = modified Gear method

solution at final time 10 sec

              VNALL
                    1    3.174401D+01   r(10)
                    2    3.478937D+00   m(10)
                    3    1.009811D-02   f(10)

================================================================
Task:                b) Parameter variation of lf from 1.0e2 to 1.0e4
-----                ------------------------------------------

*MODIFY  V1=84.99,    V2=1.674,  V3=9.975   : INITIAL CONDITION;
         r(0)         m(0)       f(0)

initial integration-step = 1nsec
minimum integration-step = 1e-20
--------------------------------
```

```
final time ....................................    10.00 sec
max. nr. of iterations pr. integration step ..    25
total cpu-time consumption ....................    24.99 sec

LF=1.0E2
--------
nr. of integrationsteps .......................    87
nr. of iterations .............................   153
nr. of rejected steps .........................     1
nr. of NO CONVERGENCE .........................     0
ORDER COUNT       1     1    15    14    12    11    32
solution at final time
                        1    3.549103D+01
                        2    3.478331D+00
                        3    1.015861D-01
LF=2.0E2
--------
nr. of integrationsteps .......................    82
nr. of iterations .............................   145
nr. of rejected steps .........................     0
nr. of NO CONVERGENCE .........................     0
ORDER COUNT       1     3    13    15    12    14    23
solution at final time
                        1    3.353713D+01
                        2    3.487958D+00
                        3    5.078264D-02
LF=5.0E2
--------
nr. of integrationsteps .......................    81
nr. of iterations .............................   145
nr. of rejected steps .........................     0
nr. of NO CONVERGENCE .........................     0
ORDER COUNT       1     4    12    13    14    15    21
solution at final time
                        1    3.221956D+01
                        2    3.483658D+00
                        3    2.024122D-02
LF=1.0E3
--------
nr. of integrationsteps .......................    79
nr. of iterations .............................   139
nr. of rejected steps .........................     0
nr. of NO CONVERGENCE .........................     0
ORDER COUNT       1     4    15    13    11    12    22
solution at final time
                        1    3.173990D+01
                        2    3.479105D+00
                        3    1.009804D-02
```

```
LF=2.0E3
--------
nr. of integrationsteps .....................   77
nr. of iterations ...........................  137
nr. of rejected steps .......................    1
nr. of NO CONVERGENCE .......................    0
ORDER COUNT       1     5    11    12    12    13    22
solution at final time
                        1    3.149317D+01
                        2    3.475384D+00
                        3    5.041528D-03
LF=5.0E3
--------
nr. of integrationsteps .....................   74
nr. of iterations ...........................  125
nr. of rejected steps .......................    0
nr. of NO CONVERGENCE .......................    0
ORDER COUNT       1     6    11    13    12    12    18
solution at final time
                        1    3.135069D+01
                        2    3.474231D+00
                        3    2.015346D-03
LF=1.0E4
--------
nr. of integrationsteps .....................   72
nr. of iterations ...........................  119
nr. of rejected steps .......................    1
nr. of NO CONVERGENCE .......................    0
ORDER COUNT       3     4    12    11    12    10    19
solution at final time
                        1    3.129986D+01
                        2    3.472832D+00
                        3    1.007225D-03

================================================================
Task:                          c) Steady state analysis, p(t)=1.0e4
-----                          ------------------------------------

*MODIFY  V1=0,   V2=0,   V3=0 : INITIAL CONDITION;
         r(0)    m(0)    f(0)

initial integration-step = 1nsec
minimum integration-step = 1e-20
--------------------------------

final time ..................................... 1000.00 sec
max. nr. of iterations pr. integration step ..   25
total cpu-time consumption ..................    88.10 sec
```

```
LF=1000
-------
nr. of integrationsteps ..................... 2896
nr. of iterations ........................... 1.83e+4
nr. of rejected steps ....................... 985
nr. of NO CONVERGENCE ....................... 3
ORDER COUNT     113  1595   913   173    25    25    51
solution at final time
                        1    9.998667D+02  r(1000)
                        2    9.998793D+00  m(1000)
                        3    9.999996D+00  f(1000)

================================================================
Task:              c) Steady state analysis, p(t)=0 at time 10 sec
-----              ------------------------------------------
                                  p(t)=1.0e+4  for  0 < t < 10 sec

*MODIFY  V1=0,   V2=0,   V3=0 : INITIAL CONDITION;
         r(0)    m(0)    f(0)

initial integration-step = 1nsec
minimum integration-step = 1e-20
--------------------------------

final time ..................................  100.00 sec
max. nr. of iterations pr. integration step ..  25
total cpu-time consumption ..................  11.64 sec

LF=1000
-------
nr. of integrationsteps ..................... 333
nr. of iterations ........................... 859
nr. of rejected steps ....................... 18
nr. of NO CONVERGENCE ....................... 0
ORDER COUNT       2    58   111    50    26    31    54
solution at final time
                        1    2.338100D-02
                        2    2.597860D-03
                        3    7.534555D-06

================================================================
Task:                          c) Check of given initial condition
-----                          ----------------------------------
                                  p(t)=1.0e+4  for  0 < t < 10 sec

*MODIFY  V1=0,   V2=0,   V3=0 : INITIAL CONDITION;
         r(0)    m(0)    f(0)

initial integration-step = 1nsec
minimum integration-step = 1e-20
--------------------------------

final time ..................................  10.00 sec
max. nr. of iterations pr. integration step ..  25
total cpu-time consumption ..................  8.74 sec
```

```
LF=1000
-------
nr. of integrationsteps ..................... 215
nr. of iterations ........................... 542
nr. of rejected steps ....................... 18
nr. of NO CONVERGENCE ....................... 0
ORDER COUNT       1     1    60    53    23    25    51
solution at final time
                        1    8.498983D+01     r(0) 84.99  ok
                        2    1.674199D+00     m(0) 1.674  ok
                        3    9.948318D+00     f(0) 9.975   ?
```

==========

References:

Erik Lindberg, ANP3 & NAP2 - A package for Circuits and Systems Simulation, pages 686-700 in R.A. Adey (Edt.), Engineering Software II, CML Publications, Southhampton 1981.

Erik Lindberg, Circuits and Systems Simulation by means of Electronic Circuit Modeling, SIMS 83 - Simulation Today and Tomorrow, 25. anniversary - Scandinavian Simulation Society, Odense, Denmark, May 30 - June 1, 1983, 28p.

Erik Lindberg, Analysis Programs for Analog Circuits and Systems, ECCTD-83, 6'th European Conference on Circuit Theory and Design, Sept. 4 to 9, 1983, Stuttgart, GFR, Proceedings page 433-435.

Erik Lindberg and Thomas Rübner-Petersen, The Theory behind NAP2, Report IT-32, October 1981, Inst. of Circuit Theory and Telecommunication, 343 Tech. Univ. Denmark, DK-2800 Lyngby, Denmark, 71p.

==========

Comments on figures:

Fig. 1 shows the result of task (a) Simple simulation of the system in the interval [0, 10] sec with the given initial conditions. (A = linear, B = logarithmic time scale).

Fig. 2 shows the result of task (b) Parameter variation of lf in the interval [1.0e2, 1.0e4].

Fig. 3 shows the result of task (c1) Calculation of steady state during constant bombardment p(t)=1.0e4.

Fig. 4 shows the result of task (c2) Calculation of steady state with bombardment p(t)=1.0e4 in the interval [0, 10] sec.

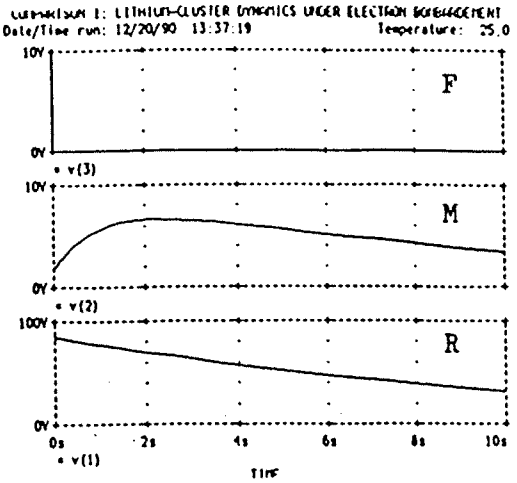


Fig. 1A

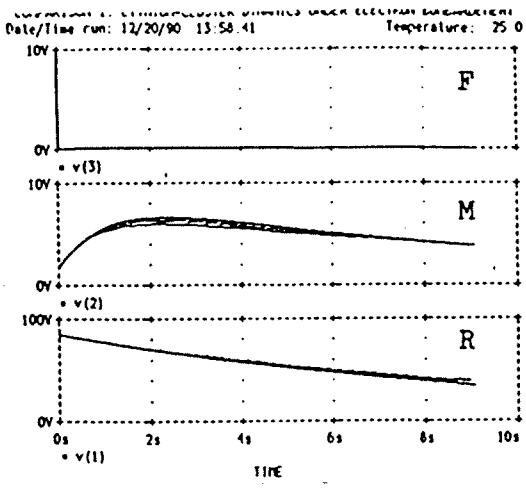


Fig. 2A

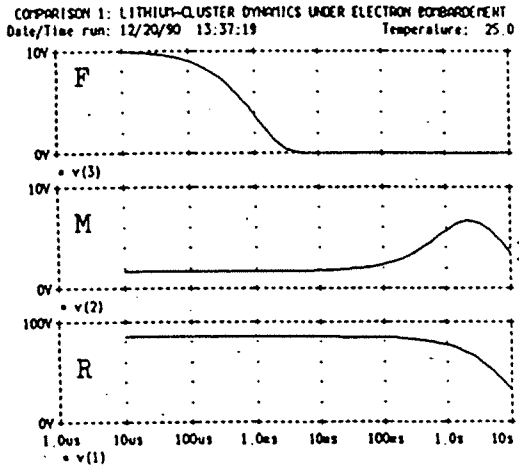


Fig. 1B

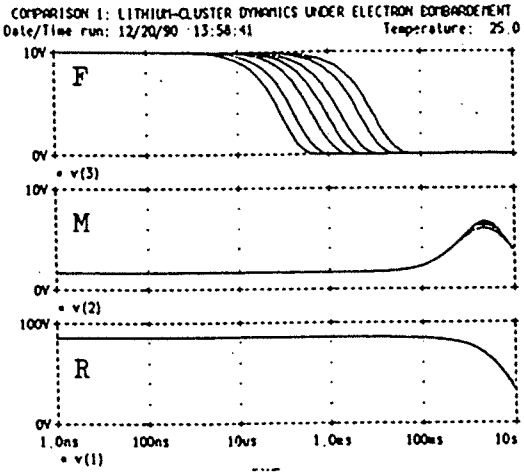


Fig. 2B

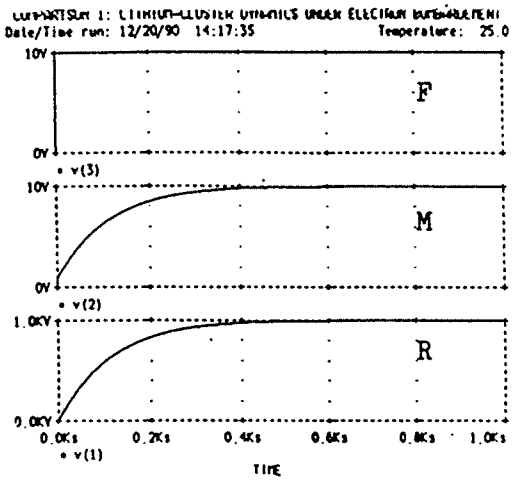


Fig. 3A

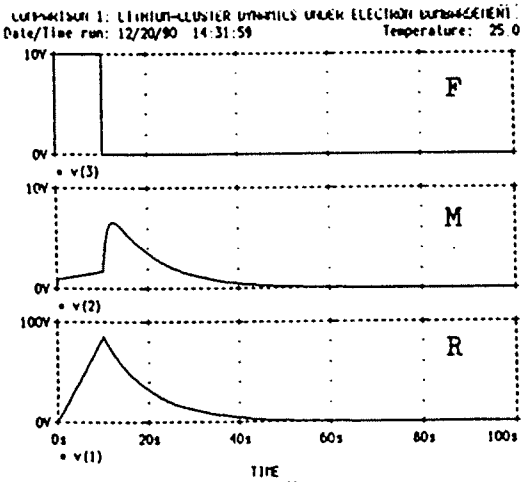


Fig. 4A

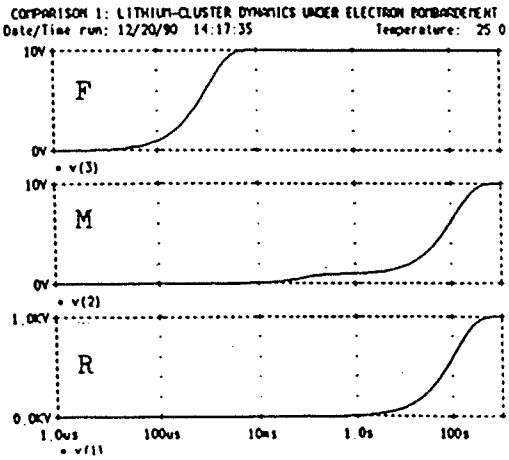


Fig. 3B

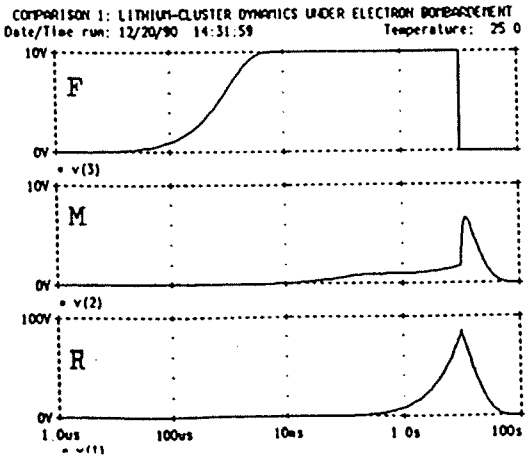


Fig. 4B

Comparison 1 - ACSL

ACSL (Advanced Continuous Simulation Language) is a widely used language obeying the CSSL-68 standard for simulation languages. ACSL consists of an ACSL precompiler translating ACSL syntax into FORTRAN and a runtime interpreter handling the generated simulation object program.

Model Description:

```
PROGRAM EUROSIM EXAMPLE No. 1
' Language ACSL Level 9, Mitchell & Gauthier Ass., U.S.A.'
' prepared by Dr. Ingrid Bausch-Gall, January 2nd, 1991 '
CONSTANT kr = 1., kf = 0.1, lf = 1000., dr = 0.1, dm = 1., p = 0.
CONSTANT fnull = 9.975, mnull = 1.674, rnull = 84.99 $ 'init. cond.'
ALGORITHM IALG = 2      $ 'take Gears stiff for integration '
CINTERVAL CINT = 0.05   $ 'store results at multiples of CINT'
CONSTANT TEND = 10.     $ 'simulation time'
' ------- model equations ------- '
    r = integ(-dr*r + kr*m*f,rnull)
    m = integ(dr*r - dm*m + kf*f*f -kr*m*f,mnull)
    f = integ(dr*r + 2.*dm*m-kr*m*f-2.*kf*f*f-lf*f + p,fnull)
TERMT(T.gt.TEND)        $ 'stop at simulation time'
END
```

ACSL-Runtime-Commands:

```
' a) Comparison of computer time '
prepar t,r,m,f                  $ 'store results of these variables'
s ialg = 1                      $ 'calc. with ADAMS-Moulton method'
spare $ start $ spare           $ 'give computer time'
s ialg = 2                      $ 'choose now Gear's stiff'
spare $ start $ spare
s ialg = 9                      $ 'one step Runge-Kutta order 4/5'
spare $ start $ spare
' b) Parameterstudies '
s ialg = 2                      $ 'choose Gears Stiff for parameterstudies'
s lf = 1.e2
start
s nrwitg = .t.                  $ 'write all results on one file'
s lf = 1.e3
start
s lf = 1.e4
start
s title = 'Example EUROSIM 1, Parameterstudies '
s title(11) = 'lf = 1.e2 (1), 1.e3 (2), 1.e4 (3)'
s ftsplt = .t.,symcpl = .t.,npccpl = 40
plot f,'xhi' = 10.,'char' = '1'  $ plot results
' c) Calculate steady state result '
s p = 1.e4
analyz 'list' = .t.,'trim'
s p = 0.
analyz 'trim'
stop
```

Results:

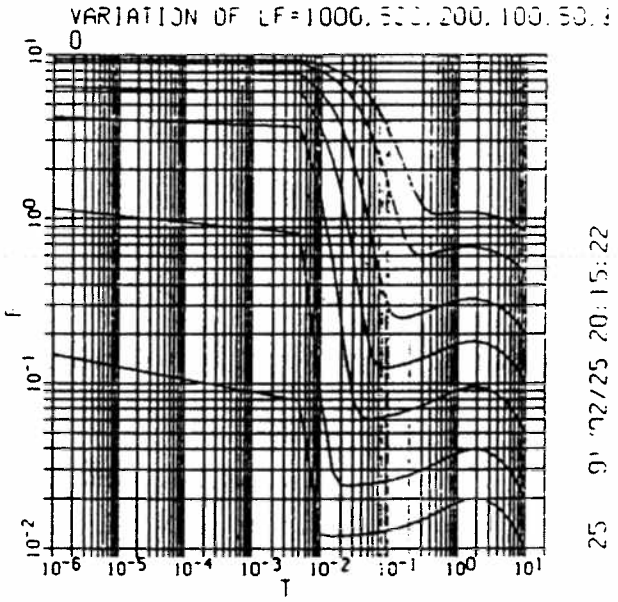
All calculations have been done on a Commodore PC-40(AT) with 12 MHz and a 80287 numeric co-processor.

Comparison of computer time (task a):

Adams-Moulton-Predictor-Corrector Method, IALG = 1	155.055 sec.
Gear's Stiff, IALG = 2	3.460 sec.
Runge-Kutta order 4/5 with stepsize control, IALG = 9	55.035 sec.

Parameterstudies:

The parameter sweep may be fromulated either "manually" at runtime level (see runtime commands) or automatically by programming a loop in the model description. The following figure shows the results of the parameter sweep with seven different values.



Calculate steady state result for $l_f = 1000$:

ACSL offers within the frequency domain analysis the TRIM command for the calculation of steady states (by means of iterative solution of $0 = \dot{x} = f(x)$). The results in this iteration (see also runtime commands) are:

p = 1.E4 gives as last iteration:

```
Newton step 0.24366500 Steep desc step 0.11443300 mu 0
State vector - iteration number 11
    F 10.0000000      M 10.0000000      R 1000.00000
Derivative vector - residual is 5.3226E-05 previous 0.02483470
    Scaled residual is 9.9485E-05 previous 0.04599450
    Z09996 5.1546E-05  Z09997 5.4854E-05  Z09998-5.3751E-05
```

p = 0. gives as last iteration:

```
Newton step 0.12913000 Steep desc step 0.06764160 mu 0
State vector - iteration number 8
    F-1.5045E-12      M-1.5373E-09      R 1.3290E-07
Derivative vector - residual is 1.3339E-08 previous 0.01348860
Scaled residual is 2.5906E-08 previous 0.02502220
    Z09996 1.1720E-08  Z09997 1.4827E-08  Z09998-1.3290E-08
```

Ingrid Bausch-Gall, BAUSCH-GALL GmbH, Wohlfartstraße 21b, D - 8000 München. Tel: +49-(0)89 3232625. Fax: +49-(0)89 3231063

Comparison 1 - FSIMUL

Description of FSIMUL

The blockoriented simulation package FSIMUL was developed at the Lehrstuhl für Regelungssysteme und Steuerungstechnik, Universität Bochum, FRG. The first usable version ran on a PDP 11 around 1975, the first effective PC version 1986 the actual version 1990 with windows, pulldown-menus, and comfortable editor functions. (Reference: K.H. Fasol, K. Diekmann (ed.): Simulation in der Regelungstechnik, Springer Verlag 1990.)

The numerical integration algorithms used are:

- Adams-Bashfort (2nd order) - AB
- predictor-corrector method (Adams-Bashfort Moulton) - PECE
- implicit method of Heun
- explicit method, Runge-Kutta (4th order) - RK4

Model description

The model (EUROSIM no. 0, November 1990, p. 25) was programmed on a 80386DX-25 w/ 80387 AT-type system, memory size 640 kB, VGA graphics board.

Model description (listing and graphical representation):

```
FSIMUL   IBM   5.0
-------------------------------------------------------------------
file: d:\fsimul\sim\lithium.sim
model: Lithium-Cluster Dynamics under Electron Bombardement
-------------------------------------------------------------------
parameters     no.   ,typ,inputs                    ;commentary

K    :1000.0    1    ,CON,                          ;lf=1000.0
K    :1.0000    2    ,CON,                          ;kr=1.0
K    :0.1000    3    ,CON,                          ;kf=0.1
K    :0.1000    4    ,CON,                          ;dr=0.1
K    :1.0000    5    ,CON,                          ;dm=1.0
K    :10000.    6    ,CON,                          ;p=1.0E4

IC   :84.990    10   ,INT,-40 60                     ;r(t)
IC   :1.6740    20   ,INT,40 -50 70 -60             ;m(t)
IC   : 9.975    30   ,INT,40 100 -60 -90 -80        ;f(t)

                40   ,MUL,4 10                      ;dr*r
                50   ,MUL,5 20                      ;dm*m
                60   ,MUL,3 20 30                   ;kr*m*f
                70   ,MUL,3 30 30                   ;kf*f^2
                80   ,MUL,1 30                      ;lf*f

K    :2.0000    90   ,GAI,70                       ;2*kf*f^2
K    :2.0000   100   ,GAI,50                       ;2*dm*m

output: block no. 30

time parameters:  endtime:  10.0 sec.
                  stepsize: h=5.0E-4
```

Lithium-Cluster under Electron Bombardment

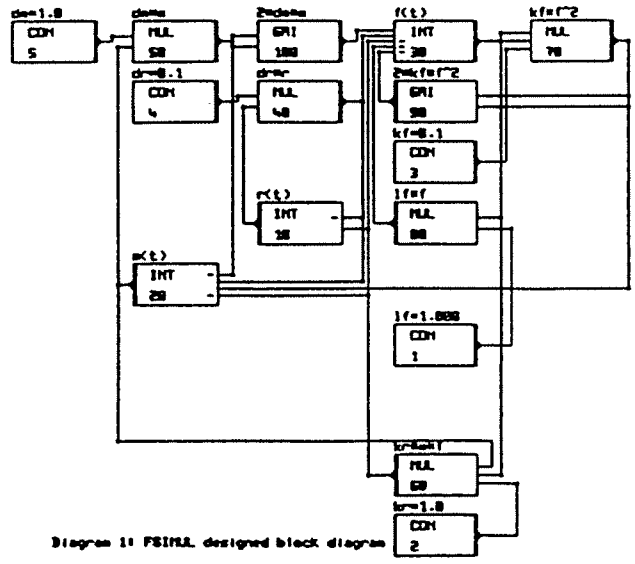


Diagram 1: FSIMUL designed block diagram

Results of the tasks

a) table of computing time (in sec.) depending on the integration algorithms with different stepsize

method	h=5.0E-4	h=1.0E-3	h=2.0E-3	h=2.5E-3
AB	104	-	-	-
PECE	163	-	-	-
Heun	182	90	-	-
RK4	187	93	48	39

(-) :numerically instable

b) parameter variation of l_f, the terminal values are:

l_f	f(t = 10 sec.)
1.0E2	0.1015
2.0E2	0.05076
5.0E2	0.02025
1.0E3	0.0101
2.0E3	0.005044
5.0E3	0.002016
1.0E4	0.001008

c) calculation of steady states (l_f = 1000), calculations in the time domain result in:

- during constant bombardment (p(t) = 1.0E4)
  f(t = 95 sec.) = 9.99
  f(t = 313 sec.) = 9.999
  f(t = 435 sec.) = 10.0
- without bombardment (p(t) = 0.0)
  f(t = 0.0023 sec.) = 1.005
  f(t = 0.0046 sec.) = 0.111
  f(t = 33.25 sec.) = 0.00101
  f(t = 79 sec.) = 1.0E-5

The figure shows the results of the parameter sweep:

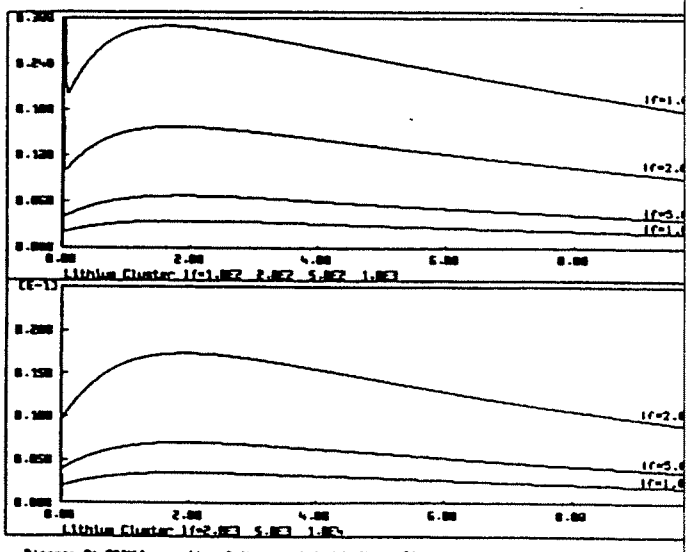


Diagram 2: FSIMUL results of the simulated Lithium-Cluster

K.H. Fasol, Lehstuhl für Regelungssysteme und Steuerungstechnik, Ruhr-Universität Bochum, Universitätsstraße 150, Geb. IB 3/152. Postfach 10 21 48. D - 4630 Bochum

Description of FSIMUL

The blockorient ed simulation package FSIMUL was developed at the Lehrstuhl für Regelsysteme und Steuerungstechnik, Universität Bochum, FRG. The first usable version run on a PDP 11 around 1975. First effective PC version 1986. Actual version 1990 with windows, pulldown-menues, and comfortable editor functions.
(Reference: K.H. Fasol, K. Diekmann (ed.): Simulation in der Regelungstechnik, Springer Verlag 1990)
The used numerical integration algorithms are:

- Adams-Bashfort (2nd. order)
- predictor-corrector method (Adams-Bashfort Moulton)
- implicit method of Heun
- explicit method, Runge-Kutta (4th. order)

Model description

The model (EUROSIM no. 0, November 1990, p. 25) was programed on a 80386DX-25 w/ 80387 AT-type system, memory size 640 kB, VGA graphics board.

FSIMUL-listing of the demanded task:

```
FSIMUL   IBM   5.0
-------------------------------------------------------------
file: d:\fsimul\sim\lithium.sim
model: Lithium-Cluster Dynamics under Electron Bombardement
-------------------------------------------------------------
parameters     no.   ,typ,inputs                    ;commentary

K   :1000.0    1     ,CON,                          ;lf=1000.0
K   :1.0000    2     ,CON,                          ;kr=1.0
K   :0.1000    3     ,CON,                          ;kf=0.1
K   :0.1000    4     ,CON,                          ;dr=0.1
K   :1.0000    5     ,CON,                          ;dm=1.0
K   :10000.    6     ,CON,                          ;p=1.0E4

IC  :84.990    10    ,INT,-40 60                    ;r(t)
IC  :1.6740    20    ,INT,40 -50 70 -60             ;m(t)
IC  : 9.975    30    ,INT,40 100 -60 -90 -80        ;f(t)

               40    ,MUL,4 10                      ;dr*r
               50    ,MUL,5 20                      ;dm*m
               60    ,MUL,2 20 30                   ;kr*m*f
               70    ,MUL,3 30 30                   ;kf*f^2
               80    ,MUL,1 30                      ;lf*f

K   :2.0000    90    ,GAI,70                        ;2*kf*f^2
K   :2.0000   100    ,GAI,50                        ;2*dm*m

output: block no. 30

time parameters:  endtime:  10.0 sec.
                  stepsize: h=5.0E-4
```

Results of the tasks

a) table of computing time (in sec.) depending on the integration algorithms with different stepsize

method	h=5.0E-4	h=1.0E-3	h=2.0E-3	h=2.5E-3
AB	104	-	-	-
PECE	163	-	-	-
Heun	182	90	-	-
RK4	187	93	48	39

(-) :numerically instable

b) parameter variation of l_f

l_f	f(t=10 sec.)
1.0E2	0.1015
2.0E2	0.05076
5.0E2	0.02025
1.0E3	0.0101
2.0E3	0.005044
5.0E3	0.002016
1.0E4	0.001008

c) calculation of steady states
(dr/dt=dm/dt=df/dt=0 $f=p/l_f$ l_f=1.0E3)

- during constant bombardment (p(t)=1.0E4)
  f(t=95 sec.) = 9.99
  f(t=313 sec.) = 9.999
  f(t=435 sec.) = 10.0

- without bombardment (p(t)=0.0)
  f(t=0.0023 sec.) = 1.005
  f(t=0.0046 sec.) = 0.111
  f(t=33.25 sec.) = 0.00101
  f(t=79 sec.) = 1.0E-5

Lithium-Cluster under Electron Bombardment

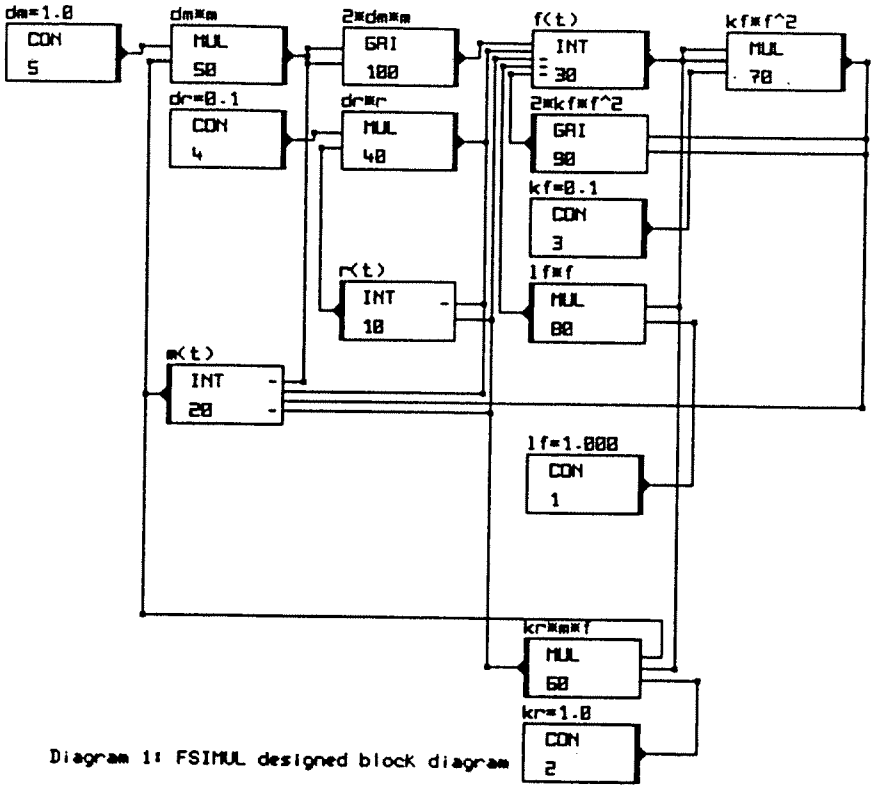


Diagram 1: FSIMUL designed block diagram

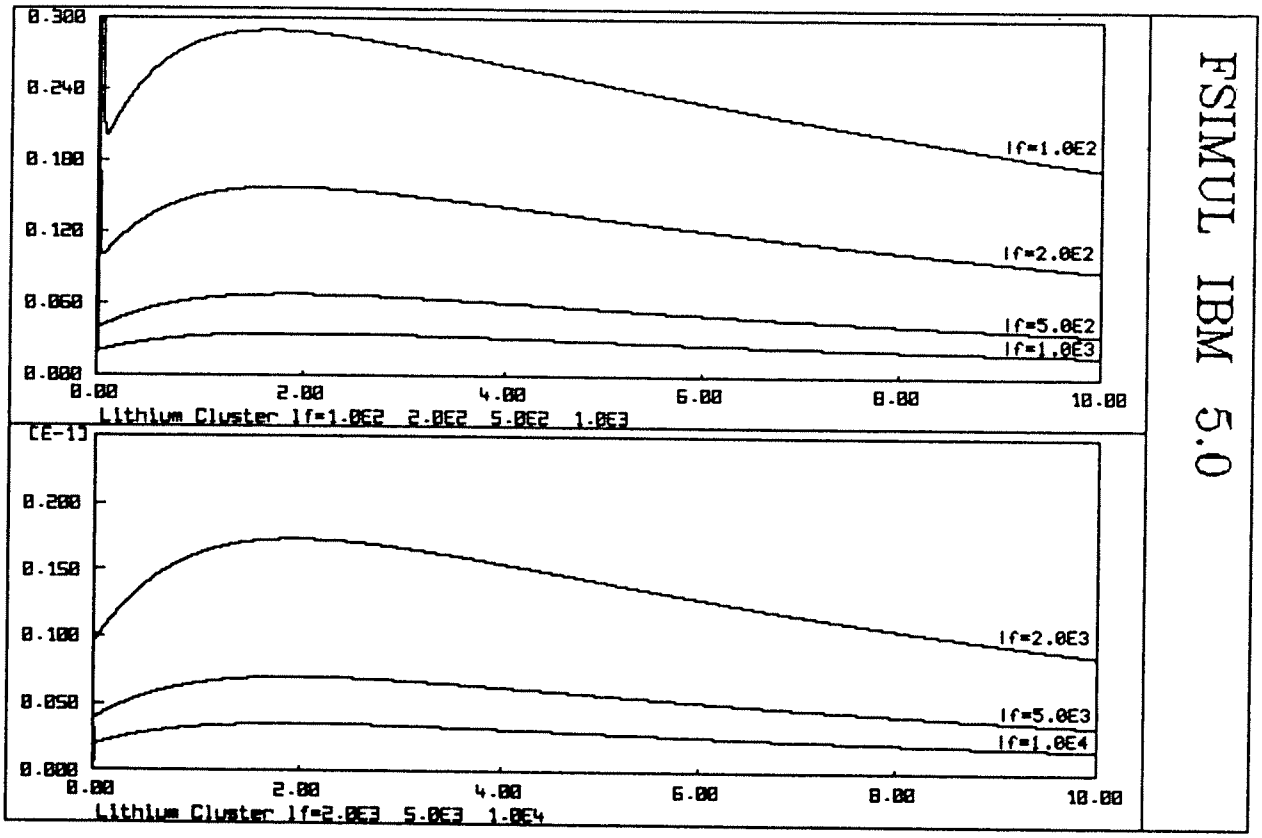


Diagram 2: FSIMUL results of the simulated Lithium-Cluster

Comparison 1 - SIMUL_R

SIMUL_R is a compiling simulation language for continuous and discrete systems. The system offers grafical and textual modelling, using one or more models in one simulation program. Examinations are done by using menus and/or a strong runtime interpreter.

The interpreter allows the usage of loops, command files (recursive, too) and arbitrary expressions with assignments and displaying. A special feature are user defined functions, which enable the user to add new commands to the system (commands for steady state, zero search, continuous and discrete optimization, statistical evaluations are available as well).

A huge grafical library supports among others moving plots, 3D-plots, niveau lines, cross plots (for displaying solutions of PDEs), animation for both, continuous and discrete systems.

SIMUL_R is an open system as it allows data input and output from and to other systems, including user input during simulation (by keys or grafical) as well as hardware in the loop.

Model description:

```
Lithium_Cluster {
 CONSTANT kr = 1, kf = 0.1, lf = 1000, dr = 0.1, dm = 1, p = 0;
 CONSTANT r0 = 84.99, m0 = 1.674, f0 = 9.975;
 CONSTANT tend = 10;

 DYNAMIC {
  DERIVATIVE {
   dr_r  = dr * r;
   kr_m_f = kr * m * f;
   dm_m  = dm * m;
   kf_f2  = kf * f * f;

   r = INTEG (- dr_r + kr_m_f, r0);
   m = INTEG (dr_r - dm_m + kf_f2 - kr_m_f, m0);
   f = INTEG (dr_r + 2*dm_m - kr_m_f - 2*kf_f2 - lf*f + p, f0);
  }
  TERMINATE t > = tend;          " termination condition "
 }
}
```

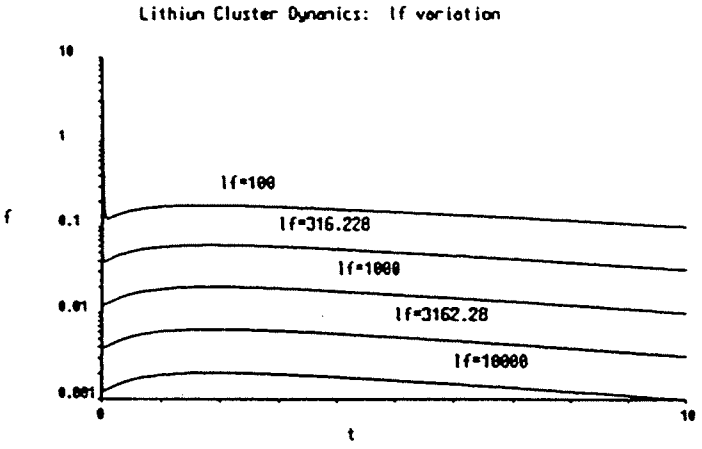
a) A relative comparison of some of SIMUL_R's integration algorithms (examinations are performed with SIMUL_R 1.13) results in:

Integration alg.	step width	time (rel to Euler)	rel. ac
Euler	0.001	1	$< 10^{-4}$
Euler (improved)	0.001	0.74	$< 10^{-4}$
Runge Kutta 4th	0.002	1.90	$< 10^{-4}$
implicit Euler	0.003	5.00	$< 10^{-4}$
implicit Euler	0.1	0.22 (!)	$< 10^{-2}$
Adams-Bashforth-Moulton (initial step width)	0.01	2.5	$< 10^{-4}$

b) The commands for the desired parameter sweep are:

```
prepare t,f,lf;                  " specify values to be prepared "
xline = 9; ynum = 3; yline = 3;  " plot legends "
number_text = true;
plot_text = 'Lithium Cluster Dynamics: lf variation';
#horiz_screen                    " use horizontal plot legends "
#for lf_log = 2,4,.5#            " for loop: with exponents "
 lf = exp(lf_log*log(10));       " compute value 10lf_log for lf "
 cint = 1/lf;                    " set accurate step width "
 cstep = (int)lf/10;             " each cstepth point is recorded "
 start;                          " start simulation run "
  " plot f logarithmic over (0.001,10), using t over (0,tend)
    as x-axis, writing lf = ... to special positions of the curve "
 plot! (t(0,tend)) *f(0.001,10) = lf_log*2-2 : 'lf = '(lf);
 plot_del = false;               " prevent deletion of last plot "
 axes_new = false;               " avoid drawing new axes twice "
#end
plot;                            " recall the last plot "
```

The figure contains the corresponding plot.



SIMUL_R's TSCHEDULE command could have been used to set the step width to a higher value after the first computation steps (for integration algorithms with constant step width).

c) The commands for the steady state analysis and the results printed are:

```
lf = 1000;
p = 10000;
STEADY_STATE;
disp 'steady state for p = ',p,':',r, m, f;
```

steady state for p = 10000 : 1000 10 10

```
p = 0;
STEADY_STATE;
disp 'steady state for p = ',p,':',r, m, f;
```

steady state for p = 0 : 0 6.75016e-014 -1.38778e-017

For information and comments, please phone or fax or write to

R. Ruzicka, SIMUTECH, Hadikgasse 150, A-1140 Vienna, Austria. Tel: +43-(0)222-82 03 87; Fax: +43-(0)222-82 93 91.

SIMUL_R

SIMUL_R is a compiling simulation language for continuous a discrete systems. The system offers grafical and textual modellin using one or more models in one simulation program. Examinations a done by using menus and/or a strong runtime interpreter.

The interpreter allows the usage of loops, command fil (recursive, too) and arbitrary expressions with assignments and di playing. A special feature are userdefined functions, which enable t user to add new commands to the system (commands for steady state, ze search, continuous and discrete optimization, statistical evaluatio are available as well).

A huge grafical library supports among others moving plots, 3 plots, niveau lines, cross plots (for displaying solutions of PDEs animation for both, continuous and discrete systems.

SIMUL_R is an open system as it allows data input and output fr and to other systems, including user input during simulation (by ke or grafical) as well as hardware in the loop.

Fig. 1 shows the simple model for Comparison 1.

```
Lithium_Cluster {

   CONSTANT kr=1, kf=0.1, lf=1000, dr=0.1, dm=1, p=0;
   CONSTANT r0=84.99, m0=1.674, f0=9.975;
   CONSTANT tend=10;

   DYNAMIC {
      DERIVATIVE {
         dr_r   = dr * r;
         kr_m_f = kr * m * f;
         dm_m   = dm * m;
         kf_f2  = kf * f * f;

         r = INTEG (- dr_r + kr_m_f, r0);
         m = INTEG (dr_r - dm_m + kf_f2 - kr_m_f, m0);
         f = INTEG (dr_r + 2*dm_m - kr_m_f - 2*kf_f2 - lf*f + p, f0);
      }
      TERMINATE t>=tend;                    " termination condition "
   }
}
```

Fig. 1 SIMUL_R model for Comparison 1.

Fig. 2 contains a comparison of some of SIMUL_R's integrati algorithms (examinations are performed with SIMUL_R 1.13).

Integration alg.	step width	time (rel to Euler)	rel. accuracy
Euler	0.001	1	$< 10^{-4}$
Euler (improved)	0.001	0.74	$< 10^{-4}$
Runge Kutta 4th	0.002	1.90	$< 10^{-4}$
implicit Euler	0.003	5.00	$< 10^{-4}$
implicit Euler	0.1	0.22 (!)	$< 10^{-2}$
Adams-Bashforth-Moulton (initial step width)	0.01	2.5	$< 10^{-4}$

Fig. 2 Comparison of integration algorithms.

ig.- 3 shows the commands for the desired parameter variation, Fig. 4 ontains the corresponding plot. SIMUL_R's TSCHEDULE command could have een used to set the step width to a higher value after the first omputation steps (for integration algorithms with constant step width).

```
repare t,f,lf;                          " specify values to be prepared "
line=9; ynum=3; yline=3;                " plot legends "
umber_text=true;
lot_text='Lithium Cluster Dynamics:  lf variation';
horiz_screen                            " use horizontal plot legends "
for lf_log=2,4,.5#                      " for loop: with exponents "
  lf=exp(lf_log*log(10));               " compute value 10^lf_log for lf "
  cint=1/lf;                            " set accurate step width "
  cstep=(int)lf/10;                     " each cstep^th point is recorded "
  start;                                " start simulation run "
    " plot  f logarithmic over (0.001,10),  using t over  (0,tend)
      as x-axis, writing lf=... to special positions of the curve "
  plot! (t(0,tend)) *f(0.001,10) = lf_log*2-2 : 'lf='(lf);
  plot_del=false;                       " prevent deletion of last plot "
  axes_new=false;                       " avoid drawing new axes twice "
end
lot;                                    " recall the last plot "
```

Fig. 3 Commands for parameter variation.

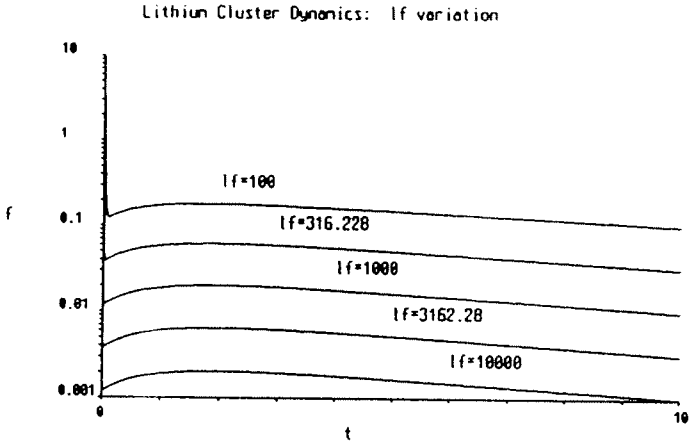


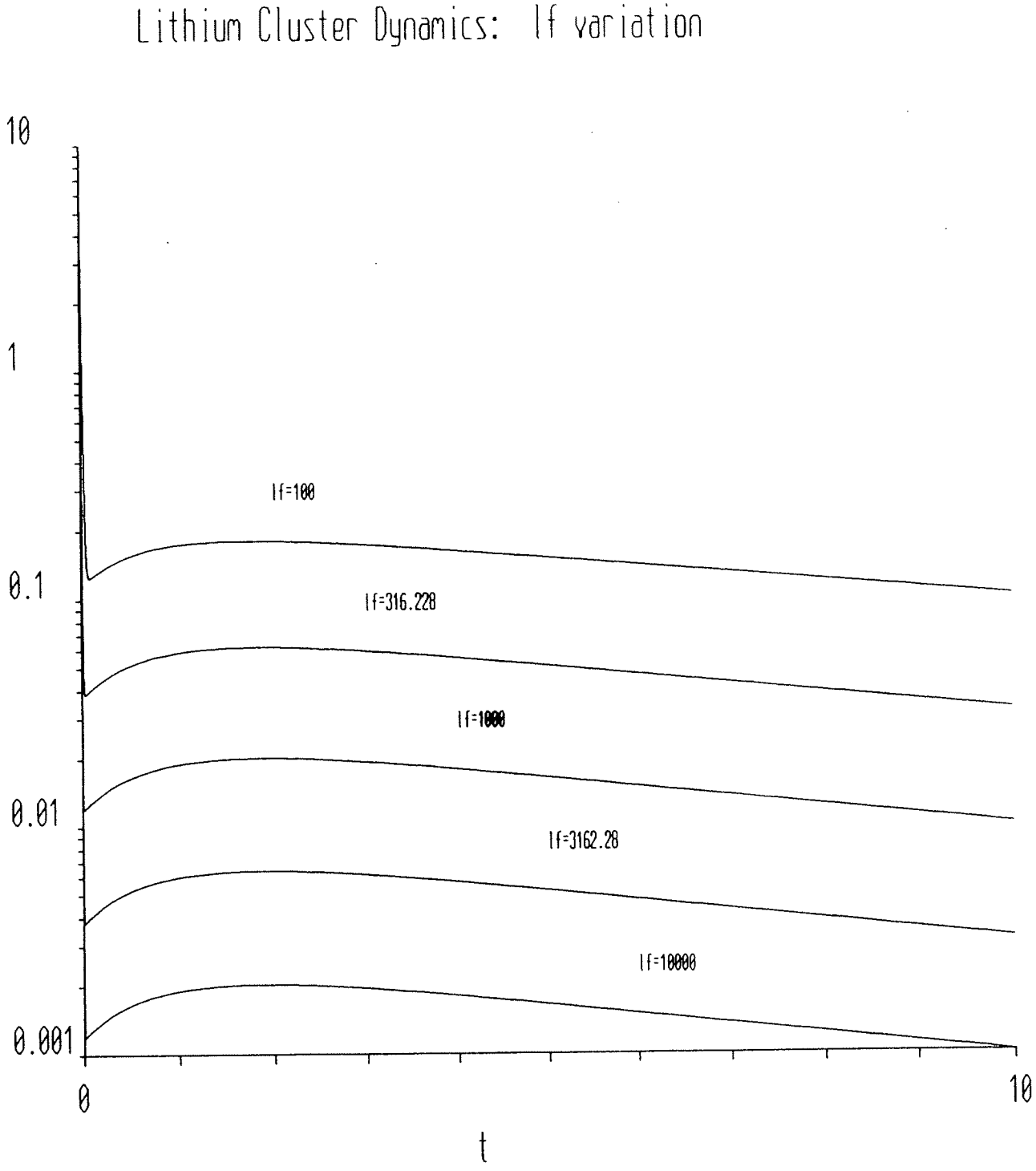
Fig. 4 Plot of parameter variation.

ig. 5 shows the commands for the steady state analysis and the esults printed.

```
:=1000;
:10000;
:EADY_STATE;
.sp 'steady state for p =',p,':',r, m, f;

  steady state for p = 10000    : 1000    10    10

:0;
:EADY_STATE;
.sp 'steady state for p =',p,':',r, m, f;

  steady state for p = 0    : 0    6.75016e-014    -1.38778e-017
```

Fig. 5 Commands and results for steady states.

)r information and comments, please phone or fax or write to
:MUTECH, Hadikgasse 150, A-1140 Vienna, Austria.
:l A-(0)222-82 03 87; Fax A-(0)222-82 93 91.


Lithium Cluster Dynamics: If variation
10
1
0.1
0.01
0.001
If=100
If=316.228
If=1000
If=3162.28
If=10000
0
10
t

Comparison 1 - XANALOG

XANALOG is a block-oriented simulation system. A ersion is available for IBM PC/AT (or 100% Compati- ble), Compaq 386 (or 100% Compatible) and IBM PS/2 Models 50, 60, 70, 80 and 30-286.

Model Description

The model is described in terms of the XANALOG block diagram of Figure 1.

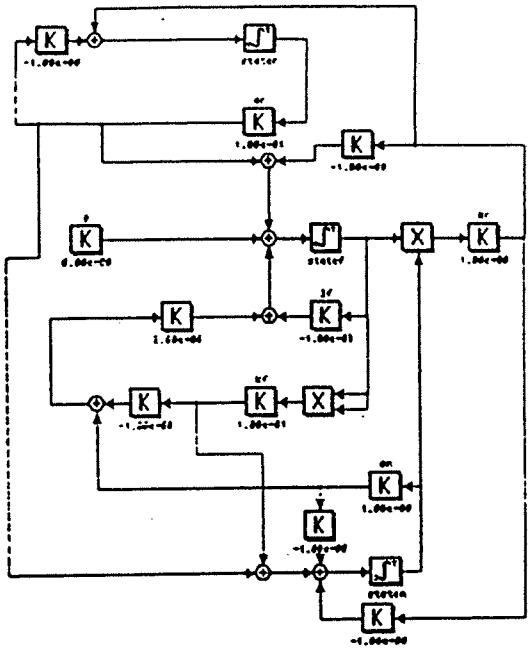


Figure 1

Results

All calculations were done using an NCR PC (80286 processor with 80287 numeric co-processor).

Comparison of Computer Time (task a):

Integration Method	Step Size (sec)	Computing Time (sec)
RK4	0.001	225
	0.002	112
	0.0025	88
	0.003	Numerically unstable
Euler	0.001	82
	0.002	Numerically unstable
Modified Euler	0.001	118
	0.002	Numerically unstable

Variation of Parameter I_f (task b).

Simulations were carried out for values of I_f of 100, 200, 500 and 1000. The results are shown in Figures 2 and 3. Figure 2 is a graph of 2*ln(f) versus time on a linear scale. Figure 3 is a graph of 2*ln(f) versus time on a logarithmic scale. In both cases the top curve represents the response for parameter $I_f = 100$, with the lower curves showing corresponding results for $I_f = 200$, 500 and 1000 respectively. These two figures show very clearly the stiff nature of this simulation problem. They also provide an illustration of two of the many different forms of graphical presentation possible with the facilities of the XANALOG Time Domain Post-Processor.

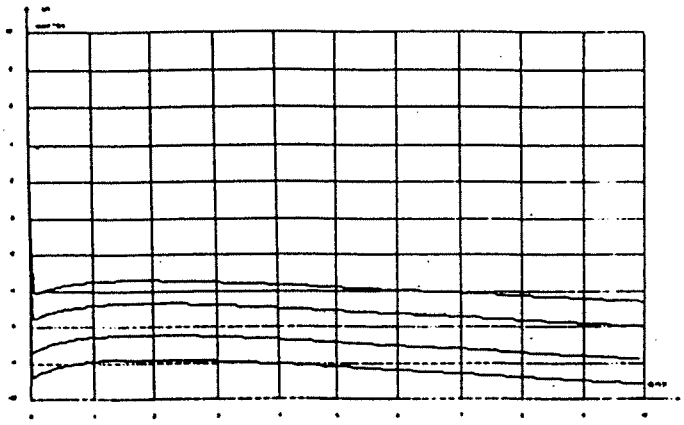


Figure 2

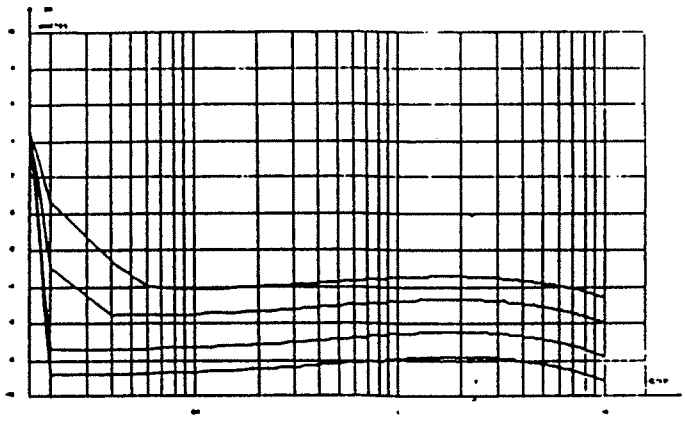


Figure 3

Calculation of Steady State (task c).

Calculations in the time domain for $I_f = 1000$ resulted in the following:

During constant bombardment (p = 10000)

f(t = 100 sec.) = 9.98991

Without bombardment (p = 0)

f(t = 100 sec.) = 1.27722E-6

D. Murray-Smith, Department of Electronics and Electrical Engineering, University of Glasgow, Glasgow G12 8QQ, Scotland, U.K.

Comparison of Simulation Software

Comparison 1 XANALOG

XANALOG is a block-oriented simulation system. A version is available for IBM PC/AT (or 100% Compatible), Compaq 386 (or 100% Compatible) and IBM PS/2 Models 50, 60, 70, 80 and 2 * 25.

Model description

The model (EUROSIM Simulation News Europe No 1, November 1990, p. 26) is described in terms of the XANALOG block diagram of Figure 1.

Results

All calculations were done using an NCR PC (80286 processor with 80287 numeric co processor)

Comparison of Computer Time (task a).

Integration Method	Step Size (sec)	Computing Time (sec)
RK4	0.001	229
	0.002	112
	0.0025	88
	0.003	Numerically unstable
Euler	0.001	82
	0.002	Numerically unstable
Modified Euler	0.001	118
	0.002	Numerically unstable

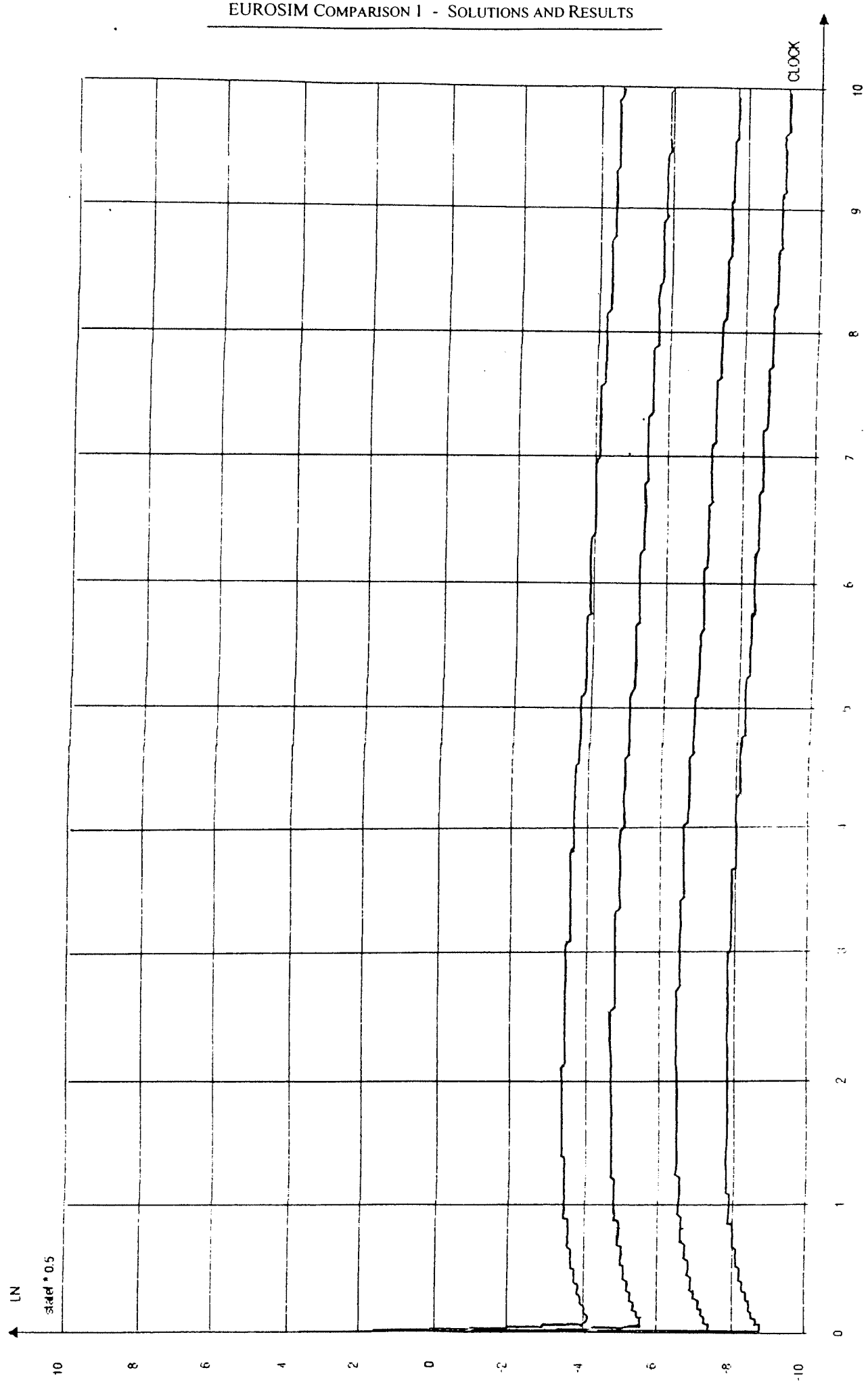
Variation of Parameter lf (task b).

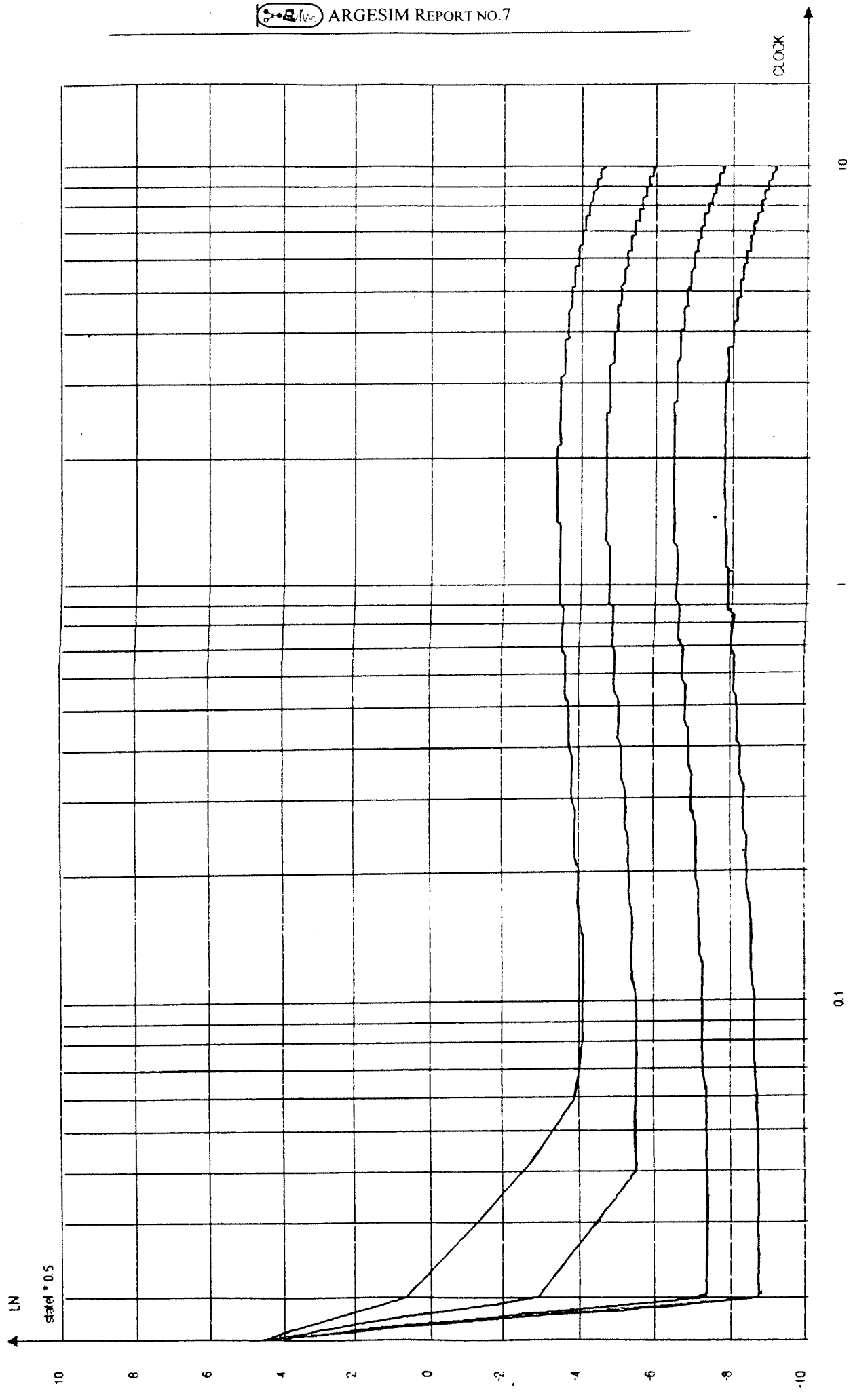
Simulations were carried out for values of lf of 100, 200, 500 and 1000. The results are shown in figures 2 and 3. Figure 2 is a graph of 2*ln(f) versus time on a linear scale. Figure 3 is a graph of 2*ln(f) versus time on a logarithmic scale. In both cases the top curve represents the response for parameter lf=100, with the lower curves showing corresponding results for lf=200, 500 and 1000 respectively. These two figures show very clearly the stiff nature of this simulation problem. They also provide an illustration of two of the many different forms of graphical presentation possible with the facilities of the XANALOG Time Domain Post-Processor.

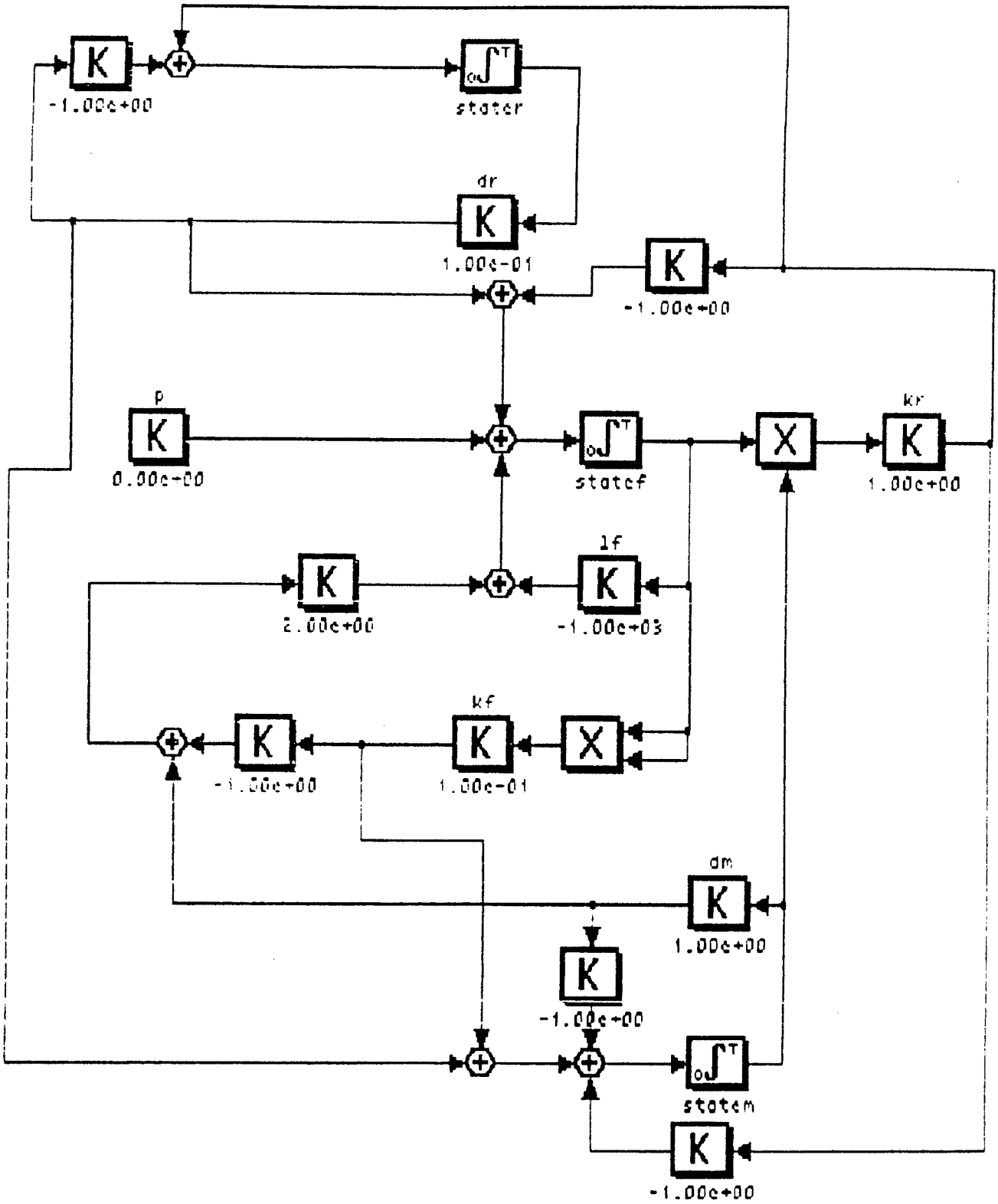
Calculation of Steady State (task c).

Calculations in the time domain for lf=1000 resulted in the following

During constant bombardment (p=10.00) f(t=100 sec) = 0.98991
Without bombardment (p=0) f(t=100 sec.) = 1.27722E-6


CLOCK
LN
Scale * 0.5
10
8
6
4
2
0
-2
-4
-6
-8
-10
0
1
2
3
4
5
6
7
8
9
10


CLOCK
10
1
0.1
LN
stahl *0.5
10
8
6
4
2
0
-2
-4
-6
-8
-10


K
-1.00e+00
statef
dr
K
1.00e-01
K
-1.00e+00
p
K
0.00e-00
statef
X
kr
K
1.00e+00
K
2.00e+00
lf
K
-1.00e+03
K
-1.00e+00
kf
K
1.00e-01
X
dm
K
1.00e+00
K
-1.00e+00
statem
K
-1.00e+00

Comparison 1 - HYBSYS

The development of the Hybrid Simulation System HYBSYS has been started 12 years ago at the Technical University of Vienna, Austria, on a hybrid machine. Now the latest version 7.0 runs on AT-compatible PCs under DOS 3.2 or higher and on UNIX-based workstations with the X-Window-System. HYBSYS is a simulation environment that supports modelling, identification, and optimization, working interpretative. So there is no need of any FORTRAN or C-Compiler, although tested models can be compiled in memory for faster run.

Model description:

```
par
    kr = 1, kf = .1, lf = 1000, dr = .1, dm = 1;
    f0 = 9.975, m0 = 1.674, r0 = 84.99;
    p = 0.0;
end
var
    f,m,r;
    krmf,kff2,dmm,drr;
end
equ
    krmf = mult(kr*m,f);
    kff2 = mult(kf*f,f);
    dmm = mult(dm,m);
    drr = mult(dr,r);
    r = integ(r0,-drr,krmf);
    m = integ(m0,drr,-dmm,kff2,-krmf);
    f = integ(f0,drr,2*dmm,-krmf,-2*kff2,-lf*f,p);
end
run.mtd = 7
run.step = 1.e-5;
plot.xaxtyp = 4; plot.zaxtyp = 4; plot.zlog = 1;
plot.xtext = "T"; plot.ztext = "LOG10( F )";
plot.htext = "LITHIUM-CLUSTER DYNAMICS";
plot.axmode = 0; plot.xsctyp = *; plot.zsctyp = *;
plot.xmin = 0; plot.xmax = 10.;
plot.zmin = 1.e-3; plot.zmax = 10.;
run.ssize = 5000;
mtd smmo:etime = 9;
```

To accelerate the calculation (larger stepsize after tend = 1) and to measure the time the macro LCD1.HYB has been used:

```
tend = 1;
etime;f;t0 = 1;tend = 10;run.ic = 0;plot.s = 1;
ndt = 1000;run.step = 1.e-4;f;etime:
t0 = 0;run.ic = 1;run.step = 1.e-5;ndt = 100;
```

The model was tested on a DECStation 3100 (MIPS R2000 processor, R2010 coprocessor, 16.67 MHz) under Ultrix 3.2 and X-Windows X11R4.

Results of the tasks:

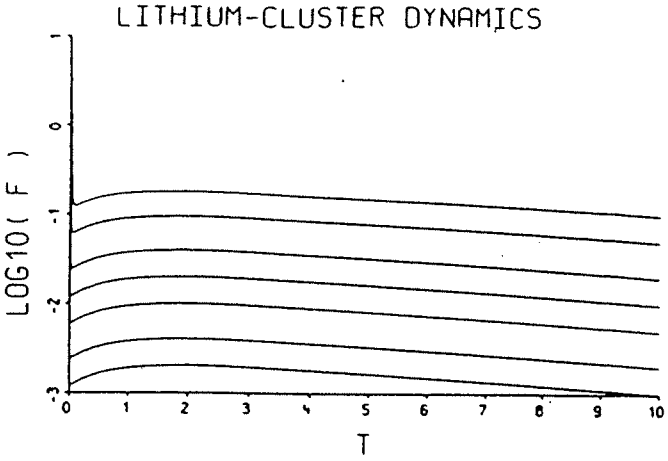
a)

method	step	time in sec.
1 (Euler)	1.E-04	8.47
4 (Runge Kutta 4th order)	2.E-04	9.31
7 (Runge Kutta Fehlberg)	1.E-05*	9.98
7 (same, with LCD1.HYB)	1.E-05*	9.38
8 (Adams Moulton)	1.E-05*	16.80
8 (same, with LCD1.HYB)	1.E-05*	18.00

* initial stepsize

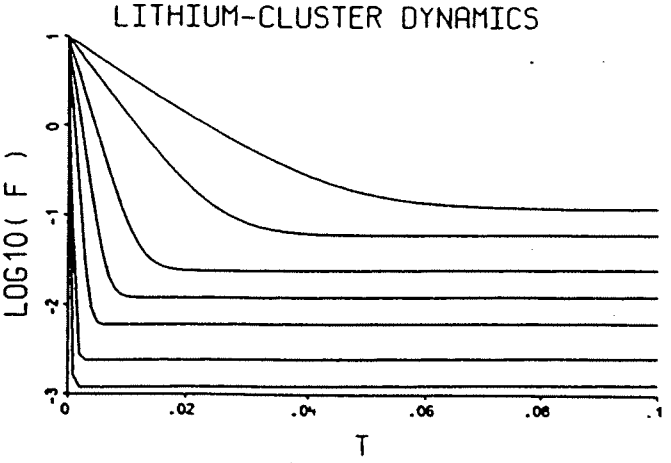
b) The command for the parameter loop is:

lf = {100,200,500,1000,2000,5000,10000}! lcd1; lf, f:



here the 'etime' command in LCD1.HYB is not necessary; the command for the next figure is:

lf = {100,200,500,1000,2000,5000,10000}! f



c) The following parameters and commands for the steady state analysis

```
p = 0.0              (p = 1000.0)
lf = 1000
trim.ceps = 1.e-5
trim.dmax = 100
trim, l
```

deliver these results:

```
Solution after 15 Evaluations
r = .4684E-03
m = -.7858E-08
f = -.2328E-09
```

respectively:

```
Solution after 34 Evaluations
r = .1000E+04
m = .1000E+02
f = .1000E+02
```

For further information, please contact:

Dietmar Solar, Schönbrunnerstraße 65, A - 1050 Vienna, Austria, Tel: +43-(0)222 5562864

Comparison 1 HYBSYS

The development of the Hybride-Simulation-System HYBSYS has been started 12 years ago at the Technical University of Vienna, Austria, on a hybride machine.
Now the latest version 7.0 runs on every AT-compatible PC under DOS 3.2 or higher and on UNIX-based workstations with the X-Window-System.
HYBSYS is a simulation environment that supports modelling, identification and optimization working interpretative. So there is no need of any FORTRAN or C-Compiler, although tested models can be compiled in memory for faster run.

Model description:

```
par
        kr = 1, kf = .1, lf = 1000, dr = .1, dm = 1;
        f0 = 9.975, m0 = 1.674, r0 = 84.99;
        p  = 0.0;
end
var
        f,m,r;
        krmf,kff2,dmm,drr;
end
equ
        krmf = mult(kr*m,f);
        kff2 = mult(kf*f,f);
        dmm = mult(dm,m);
        drr = mult(dr,r);
        r = integ(r0,-drr,krmf);
        m = integ(m0,drr,-dmm,kff2,-krmf);
        f = integ(f0,drr,2*dmm,-krmf,-2*kff2,-lf*f,p);
end
run.mtd = 7
run.step = 1.e-5;
plot.xaxtyp = 4;
plot.zaxtyp = 4;
plot.zlog = 1;
plot.xtext = "T"; plot.ztext = "LOG10( F )";
plot.htext = "LITHIUM-CLUSTER DYNAMICS";
plot.axmode = 0; plot.xsctyp = *; plot.zsctyp = *;
plot.xmin = 0; plot.xmax = 10.;
plot.zmin = 1.e-3; plot.zmax = 10.;
run.ssize = 5000;
mtd smmo:etime=9;
```

To accelerate the calculation (smaller stepsize after tend=1) and to measure the time the macro LCD1.HYB has been used:

```
tend=1;
etime;f;t0=1;tend=10;run.ic=0;plot.s=1;ndt=1000;run.step=1.e-4;f;etime:
t0=0;run.ic=1;run.step=1.e-5;ndt=100;
```

The model was tested on a DECStation 3100 (MIPS R2000 processor, R2010 coprocessor, 16.67 Mhz) under Ultrix 3.2 and X-Windows X11R4.

a)

method	step size	time in sec.
1 (Euler)	1.E-04	8.47
4 (Ruge Kutta 4th order)	2.E-04	9.31
7 (Runge Kutta Fehlberg)	1.E-05*	9.98
7 (same, with LCD1.HYB)	1.E-05*	9.38
8 (Adams Moulton)	1.E-05*	16.80
8 (same, with LCD1.HYB)	1.E-05*	18.00

(* initial step size)

b) The command for the parameterloop is:

```
lf=(100,200,500,1000,2000,5000,10000)! lcd1; lf, f:
```

see graphic 1, here the 'etime' command in LCD1.HYB is not necessary; the command for graphic number 2 is:

```
lf=(100,200,500,1000,2000,5000,10000)! f
```

c) The following parameters and commands for the steady state analysis

```
p  = 0.0   (p = 1000.0)
lf = 1000
trim.ceps = 1.e-5
trim.dmax = 100
trim, l
```

deliver these results:

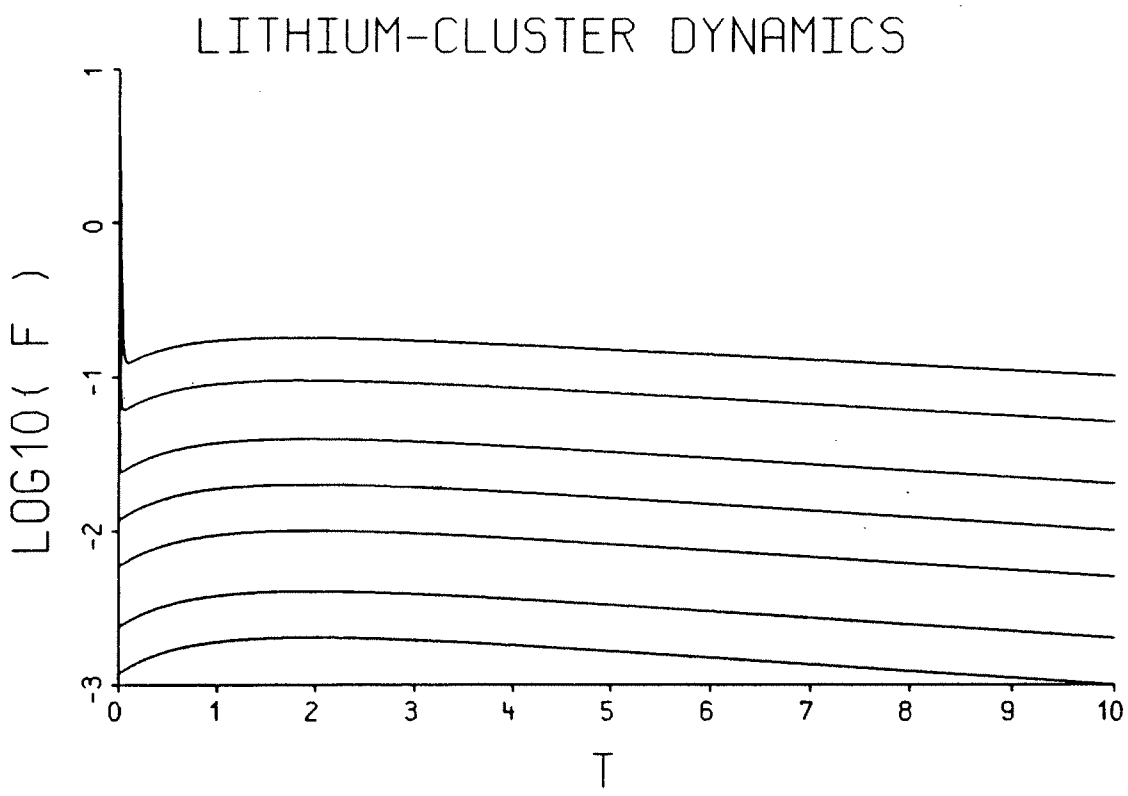
```
   Solution after   15 Evaluations

   r        =  .4684E-03    r = INTEG(r0,-drr,krmf)       r = -.4684E-04
   m        = -.7858E-08    m = INTEG(m0,drr,-dmm,kff2,-krmf)       m =  .4685E-04
   f        = -.2328E-09    f = INTEG(f0,drr,C0000*dmm,-krmf,C0001*kff2,-lf*f,p)       f =  .4705E-04
```

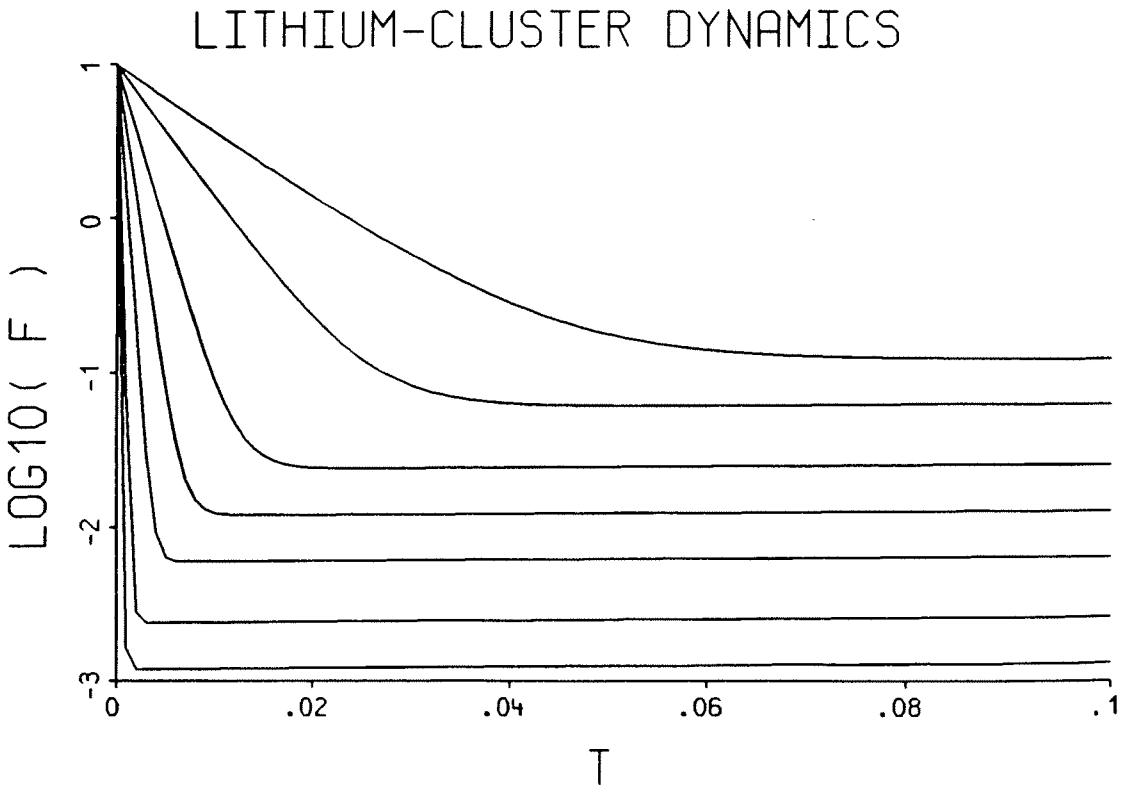
respectively:

```
   Solution after   34 Evaluations

   r        =  .1000E+04    r = INTEG(r0,-drr,krmf)       r = -.2289E-04
   m        =  .1000E+02    m = INTEG(m0,drr,-dmm,kff2,-krmf)       m =  .2289E-04
   f        =  .1000E+02    f = INTEG(f0,drr,C0000*dmm,-krmf,C0001*kff2,-lf*f,p)       f =  .0000E+00
```

For further information, please contact:

Dietmar Solar
Schoenbrunnerstr.65
A - 1050 Vienna, Austria
Tel: A-(0)222 5562864
E-mail: andreas@atvws1.tuwien.ac.at


LITHIUM-CLUSTER DYNAMICS
LOG10(F)
1
0
-1
-2
-3
0
1
2
3
4
5
6
7
8
9
10
T


LITHIUM-CLUSTER DYNAMICS
LOG10(F)
1
0
-1
-2
-3
0
.02
.04
.06
.08
.1
T

Comparison 1 - ESL

The ESL Simulation Software

ESL is a continuous systems simulation software environment, designed originally to meet the requirements of the European Space Agency for simulating spacecraft subsystems.

ESL provides two completely different user interfaces: a conventional programming language to specify a simulation; or a mouse driven graphical input facility (IMP) which allows a block diagram to be constructed to define a simulation. Either interface may be used, without the need to understand the other, to undertake complete simulation projects. For some applications a mixture of the two approaches is an ideal answer. Both routes provide excellent integrity of a simulation, and ESL IMP provides fully checked automatically generated code.

ESL is a "natural model definition language", having the following characteristics: separate experiment and model specification sections; a submodel concept; unambiguous model definition code; clear definition of non-linearities, or discontinuities; full matrix, vector, and array slice support; optional transfer function notation; linearization features, steady-state finders; and, of great importance, strict variable usage rules rigorously imposed by the ESL compiler subsystem.

An Interpreter provides fast turn-round during program development, and a Translator efficient production simulation runs. Following a simulation postmortem graphic analysis is performed by the DISP (display) subsystem.

Model Description

A commented listing of the ESL Benchmark Program is presented below. Note in particular - separate model and experiment regions; presentation of differential equations in dynamic region and analysis region in which the steady-state requirements are specified.

```
STUDY
 MODEL REACTION(:=REAL:p,lf);
-- The model defines the dynamics of the system
    REAL:f,m,r;
    CONSTANT REAL:kr/1.0/,kf/0.1/,dr/0.1/,dm/1.0/;
    INITIAL
    f:=9.975;              -- Initialization of states
    m:=1.674;
    r:=84.99;
    DYNAMIC
-- Differential equations of system
    r':=-dr*r+kr*m*f;
    m':=dr*r-dm*m+kf*f*f-kr*m*f;
    f':=dr*r+2.0*dm*m-kr*m*f-2.0*kf*f*f-lf*f+p;
    STEP
        PLOT t,f,0,TFIN,0,100; -- plot while computing
        PREPARE "lithium",t,r,f,m; -- save data for postmortem plot
    ANALYSIS
        TRIM [r,m,f]:=[r',m',f']; -- define parameters for steady-state
        PRINT "Steady state for p =",p:8.1," r,m,f =",r:8.1,m:8.1,f:8.1;
    END REACTION;
-- EXPERIMENT - the following code defines the experiment to
be carried out
    REAL:p/0.0/,lf,loglf;
    CINT:=0.1; -- defines maximum integration step length
    ALGO:=GEAR1; -- defines Gear's integration algorithm
-- Parameter variation of lf from 1.0E2 to 1.0E4 in logarithmic steps
    FOR loglf:=2.0..4.0 STEP 0.5
    LOOP
        lf:=10.0**loglf;
        REACTION(:=p,lf); -- call model with specified values of p
        PRINT "lf =",lf;
    END_LOOP;
-- Compute steady states for p = 1.0E4
    ALGO:=LIN1; -- defines "analysis" call of model to find steady-
state
    lf:=1.0E3;
    p:=1.0E4;
    REACTION(:=p,lf);
-- Compute steady states for p = 0.0
    p:=0.0;
    REACTION(:=p,lf);
END_STUDY
```

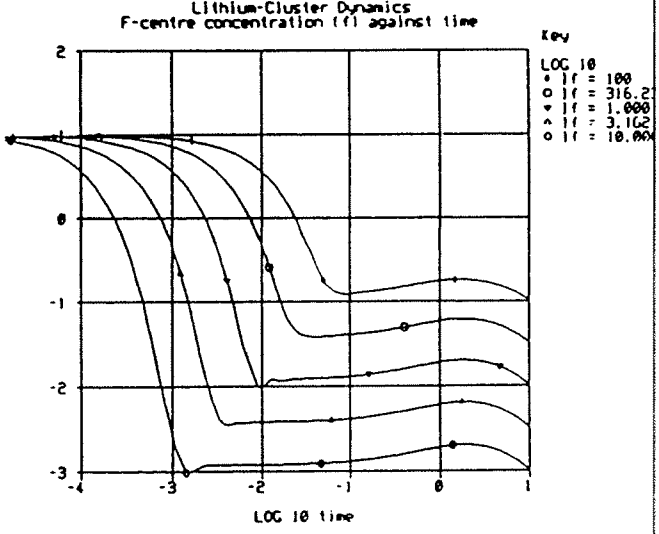
Results

(a) Comparison of integration algorithms. The stiff system was simulated over a 10s period using each of the seven integration algorithms available in ESL. Computation times for a 16MHz 386SX PC with 387 coprocessor are presented in the table below.

algorithm	max step length	computation time (s)
5th order v/step (Sarafyan)	0.1	10.00
4th order f/step Runge-Kutta	0.001	12.00
2nd order f/step Runge-Kutta	0.001	8.00
2nd order stiff (Gourlay)	0.1	0.32
Gear's stiff algorithm	0.1	0.20
Gear with diagonal Jacobian	0.1	0.25
Adams Bashforth	0.1	21.00

These results demonstrate dramatically the efficiency of the algorithms designed specifically for solving systems of stiff equations.

(b) Parameter sweep. The following figure, produced by the ESL display package, presents a plot of F-centre concentration (f) against time for a variation of l_f from 1.0E2 to 1.0E4.



(c) Steady state calculation. The ESL steady state finder returns the following steady states, which, by inspection of the equations, are clearly correct:

p	r	m	f
1.0E4	1000	10	10
0	0	0	0

D. Irving, ISIM Simulation, Frederick Road, Salford M6 6BY, U.K.

ESL Simulation Language Implementation of Lithium-Cluster Dynamics Benchmark

The ESL Simulation Software

ESL is a continuous systems simulation software environment, designed originally to meet the requirements of the European Space Agency for simulating spacecraft subsystems.

ESL provides two completely different user interfaces: a conventional programming language to specify a simulation; or a mouse driven graphical input facility (IMP) which allows a block diagram to be constructed to define a simulation. Either interface may be used, without the need to understand the other, to undertake complete simulation projects. For some applications a mixture of the two approaches is an ideal answer. Both routes provide excellent integrity of a simulation, and ESL IMP provides fully checked automatically generated code.

ESL is a "natural model definition language", having the following characteristics: separate experiment and model specification sections; a submodel concept; unambiguous model definition code; clear definition of non-linearities, or discontinuities; full matrix, vector, and array slice support; optional transfer function notation; linearization features, steady-state finders; and, of great importance, strict variable usage rules rigorously imposed by the ESL compiler subsystem.

An Interpreter provides fast turn-round during program development, and a Translator efficient production simulation runs. Following a simulation post-mortem graphic analysis is performed by the DISP (display) subsystem.

Model Description

A commented listing of the ESL Benchmark Program is presented below. Note in particular - separate model and experiment regions; presentation of differential equations in dynamic region and analysis region in which the steady-state requirements are specified.

```
STUDY
  MODEL REACTION(:=REAL:p,lf);
-- The model defines the dynamics of the system
    REAL:f,m,r;
    CONSTANT REAL:kr/1.0/,kf/0.1/,dr/0.1/,dm/1.0/;
    INITIAL
      f:=9.975;          -- Initialization of states
      m:=1.674;
      r:=84.99;
    DYNAMIC
-- Differential equations of system
      r':=-dr*r+kr*m*f;
      m':=dr*r-dm*m+kf*f*f-kr*m*f;
      f':=dr*r+2.0*dm*m-kr*m*f-2.0*kf*f*f-lf*f+p;
      STEP
        PLOT t,f,0,TFIN,0,100;       -- plot while computing
        PREPARE "lithium",t,r,f,m;  -- save data for postmortem plot
    ANALYSIS
      TRIM [r,m,f]:=[r',m',f'];      -- define parameters for steady-state
      PRINT "Steady state for p =",p:8.1,"  r,m,f =",r:8.1,m:8.1,f:8.1;
  END REACTION;
--
-- EXPERIMENT  - the following code defines the experiment to be carried out
  REAL:p/0.0/,lf,loglf;
  CINT:=0.1;       -- defines maximum integration step length
  ALGO:=GEAR1;     -- defines Gear's integration algorithm
-- Parameter variation of lf from 1.0E2 to 1.0E4 in logarithmic steps
  FOR loglf:=2.0..4.0 STEP 0.5
  LOOP
    lf:=10.0**loglf;
    REACTION(:=p,lf);   -- call model with specified values of p and lf
    PRINT "lf =",lf;
  END_LOOP;
```

```
-- Compute steady states for p = 1.0E4
  ALGO:=LIN1;      -- defines "analysis" call of model to find steady-state
  lf:=1.0E3;
  p:=1.0E4;
  REACTION(:=p,lf);
-- Compute steady states for p = 0.0
  p:=0.0;
  REACTION(:=p,lf);
END_STUDY
```

Results

(a) Comparison of integration algorithms

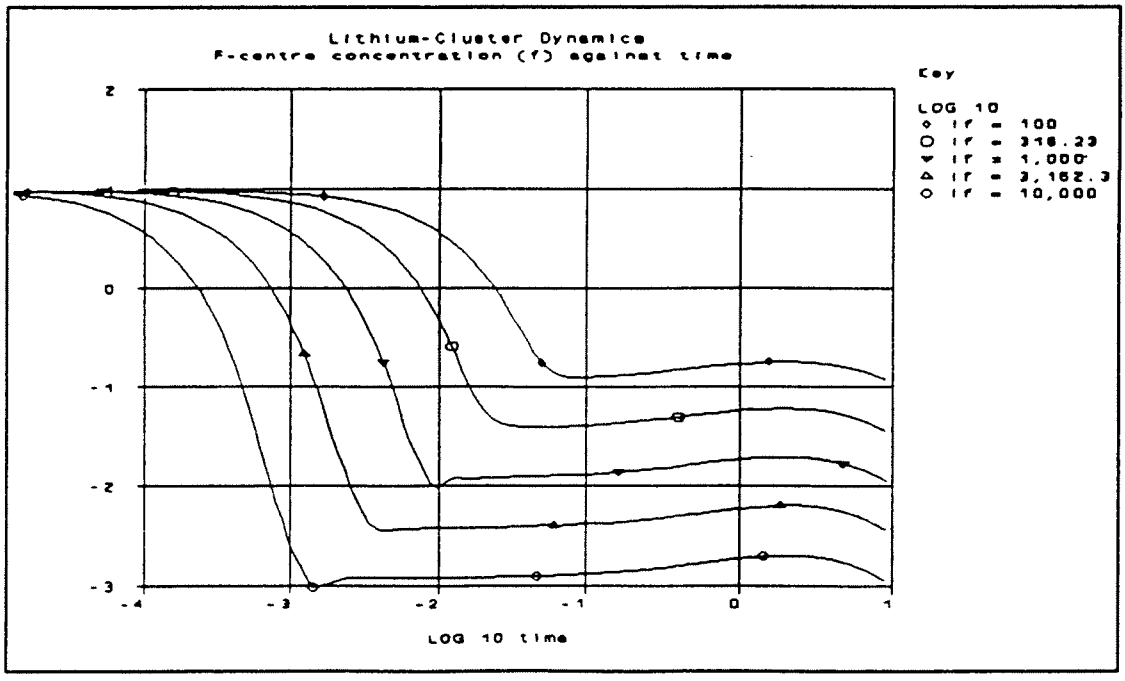
The stiff system was simulated over a 10s period using each of the seven integration algorithms available in ESL. Computation times for a 16MHz 386SX PC with 387 coprocessor are presented in the table below.

algorithm	max step length	computation time (s)
5th order v/step (Sarafyan)	0.1	10.00
4th order f/step Runge-Kutta	0.001	12.00
2nd order f/step Runge-Kutta	0.001	8.00
2nd order stiff (Gourlay)	0.1	0.32
Gear's stiff algorithm	0.1	0.20
Gear with diagonal Jacobian	0.1	0.25
Adams Bashforth	0.1	21.00

These results demonstrate dramatically the efficiency of the algorithms designed specifically for solving systems of stiff equations.

(b) Parameter sweep

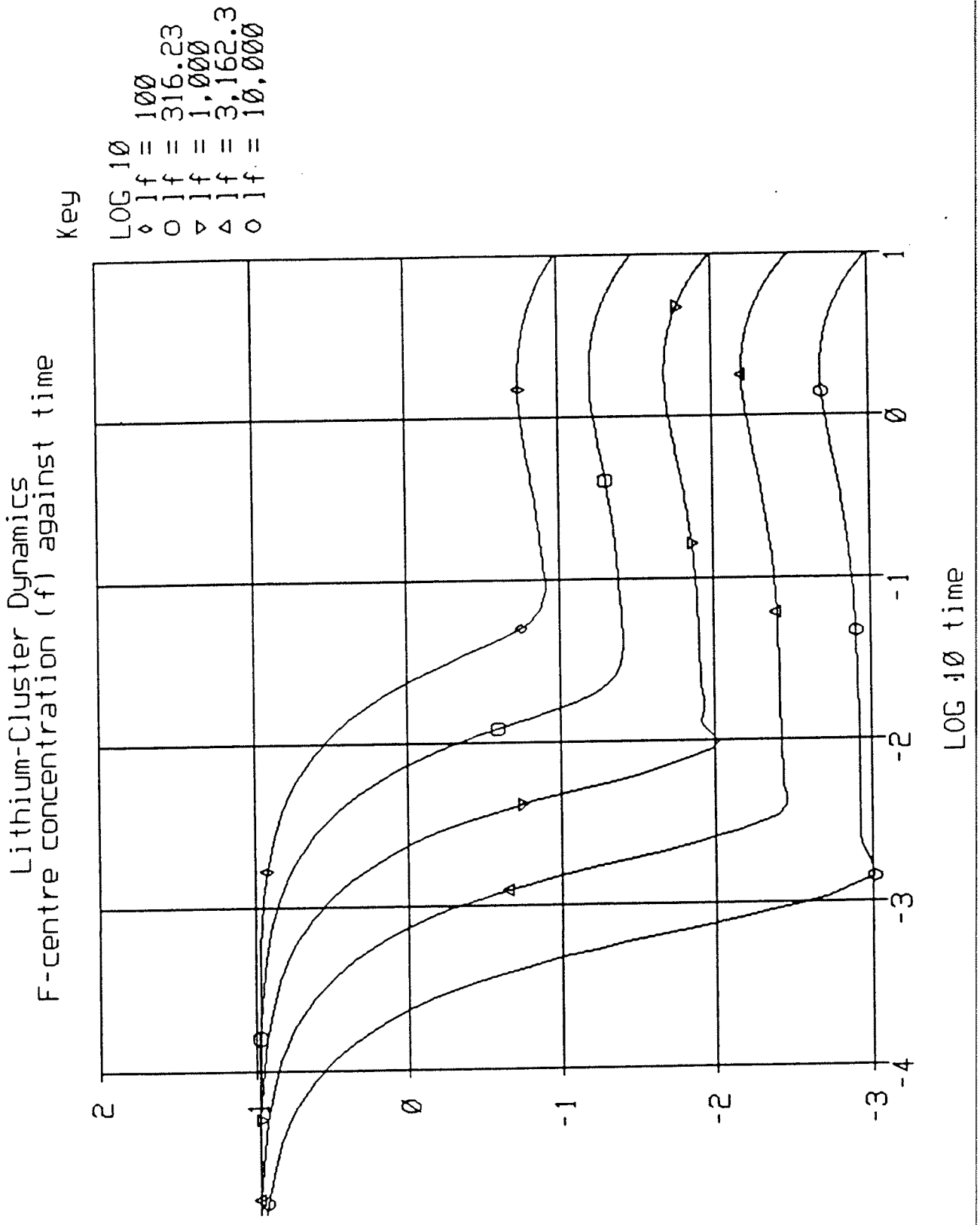
The following figure, produced by the ESL display package, presents a plot of F-centre concentration (f) against time for a variation of l_f from 1.0E2 to 1.0E4.



(c) Steady state calculation

The ESL steady state finder returns the following steady states, which, by inspection of the equations, are clearly correct:

p	r	m	f
1.0E4	1000	10	10
0	0	0	0


Lithium-Cluster Dynamics
F-centre concentration (f) against time
Key
LOG 10 lf = 100
lf = 316.23
lf = 1,000
lf = 3,162.3
lf = 10,000
2
1
0
-1
-2
-3
1
0
-1
-2
-3
-4
LOG 10 time

Comparison 1 - SIL

SIL is a general purpose simulation system with a mathematically oriented user-interface. It is designed to solve (in general) differential-algebraic equations eventually with discontinuities. It can handle discrete systems as well. The results are displayed graphically during the solution phase.

The SIL language is freeformat and statement oriented. It is specially designed for the description of simulation models. Below the "comparison 1" model is given in the SIL language. This model also includes auxillary statements needed for logarithmic scaling of the axes.

```
BEGIN
VARIABLE r(84.99), f(9.975), m(1.674),
         LOG10, LOGr, LOGf, LOGm, LOGtime;
PARAMETER Kr(1), Kf(0.1), Lf(1000), Dr(0.1), Dm(1), p(0);
DERIVATIVE rdot(r), mdot(m), df(f);
TIME T(0:10);
METHOD := 139;  (* Stiff option for integrator  *)
ABSERROR := 0; RELERROR := 1.0E-5;

(* The equations  *)

rdot := -Drer + Krmff;
mdot :=  Drer - Krmff - Dmem + Lfmf;
df   :=  Drer - Krmff + 2Dmem - 2Kffff -Lf*f + p;

(* Output statements *)

LOG10 := 1/LOG(10);
LOGr := LOG(r)*LOG10;
LOGf := LOG(f)*LOG10;
LOGm := LOG(m)*LOG10;
LOGtime := LOG(T+1.0E-20)*LOG10;
WRITE(1000,LOGr, LOGf, LOGm, LOGtime);

PLOT(1000,LOGf(LOGtime))
END.
```

Figure 1: SIL model.

The below screendump shows the results from running this model.

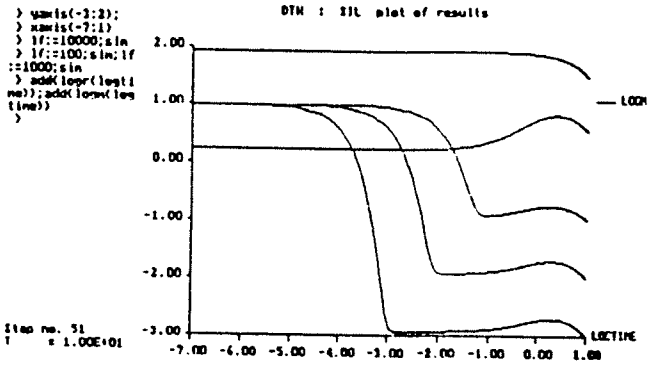


Figure 2: Screendump.

It takes less than 5 minutes (including the screendump) on an 8 MHz IBM PC/XT-286 with a 6 MHz co-processor to produce the above results. Specially for Lf = 10000 it is essential to use relative error tolerance in order to avoid f being negative. In the below table the CPU time (in seconds) is given for solving the problem (Lf = 1000) with different relative accuracies. IBM is the above XT-286 and NCR is a 16 MHz 80386 with a 16 MHz 80387.

RELERROR	CPU-seconds		#STEPS	
	IBM	NCR	ACCEPT	REJECT
1.0E-2	6.92	2.64	33	3
1.0E-4	13.68	4.01	50	3
1.0E-6	21.59	5.60	57	2
1.0E-8	31.74	7.58	61	2
1.0E-10	50.80	11.43	86	2

Table 1: CPU-time for different accuracy requirements.

In order to compute the steady state solution the model is changed to a pure algebraic problem (the derivatives are set to zero) and the model is run "in batch mode"; that is, the results are written to a .LST file. Below this file is shown for this problem; notice that the solution time is only 0.72 seconds on the XT-286.

```
SIL VERSION 2.0 (880808)
DTN          90-07-06                                  91-05-08  15:03:05  PAGE 01

  1 1-  BEGIN
  2 --  VARIABLE r(84.99), f(9.975), m(1.674),
  3 --  PARAMETER Kr(1), Kf(0.1), Lf(1000), Dr(0.1), Dm(1), p(1E-06);
  4 --
  5 --  (* The equations  *)
  6 --
  7 --  1.0E-6 := -Drer + Krmff;
  8 --  1.0E-6 :=  Drer - Krmff - Dmem + Lfmf;
  9 --  1.0E-6 :=  Drer - Krmff + 2Dmem - 2Kffff -Lf*f + p;
 10 --
 11 --  (* Output statements *)
 12 --  WRITE(m,r,f)
 13 -1  END.

  1.92  SECONDS IN COMPILATION

MODEL CONSISTS OF :
   6 PARAMETERS
   3 IMPLICIT STATIC VARIABLES


SIMULATION STATISTICS:

   NUMBER OF ACCEPTED STEPS       :   0
   TOTAL NUMBER OF FUNCTION CALLS :   1
   NUMBER OF ALGEBRAIC ITERATIONS :   9

SIMULATION OPTIONS USED:

   DEBUG              :   1
   METHOD             :  139
   MAXORDER           :   10
   MAXCPU             :  0.0
   INITIAL TIME       : 0.000E+0000
   FINAL TIME         : 0.000E+0000
   MAXIMUM STEPSIZE   : 0.000E+0000
   INITIAL STEPSIZE   : 0.000E+0000
   ABSERROR           : 1.000E-0005
   RELERROR           : 1.000E-0005

PARAMETER VALUES :
KR          =  1.000000E+0000 KF          =  1.000000E-0001 LF          =  1.000000E+0003
DR          =  1.000000E-0001 DM          =  1.000000E+0000 P           =  1.000000E-0006

          S I L   S I M U L A T I O N   R E S U L T S

   Time          M             R             F

0.000000E+0000  1.000000E+0001  1.000000E+0003  1.000000E+0001

  0.72  SECONDS in execution

 214.5  KBytes left in Lmem Heap memory
```

Figure 3: Steady state solution.

Reference:
SIL - a Simulation Language, Users Guide.
Niels Houbak, Lecture Notes in Computer Science. Vol 426.
1990 Springer Verlag.

Niels Houbak.
Lab. for Energetics, Build. 403, DTH
DK-2800 LYNGBY, DENMARK.

Comparison 1 - 386-MATLAB

MATLAB is a C-based general tool for mathematical and engineering calculations with limited capabilities for simulation of non-linear equation systems. Versions are available for PCs, workstations and mainframes.

Model Description: The model may be transformed to the vector/matrix equation

$$\frac{dx}{dt} = Ax + Bu \qquad \text{with } x^t = [r,m,f],\ u^t = [mf^2, p] \text{ and}$$

$$A = \begin{bmatrix} -d_r & 0 & 0 \\ d_r & -d_m & 0 \\ d_r & 2d_m & -l_f \end{bmatrix} \quad B = \begin{bmatrix} k_r & 0 & 0 \\ -k_r & k_f & 0 \\ -k_r & -2k_f & 1 \end{bmatrix}$$

and it is implemented in the following m-file:

```
function xs = lodueb1(x)
p = par(1,7);
A = par(1:3,1:3);
B = par(1:3,4:6);
u = [x(2)*x(3) ; x(3)^2; p] ;
xs= A*x + B*u;
```

Results: All calculations were done on an IBM PS/S Model 80 (80386 processor with an 80387 numeric coprocessor) using 386-MATLAB. MATLAB contains two variable step integration routines based on the Runge-Kutta method: ODE23 and ODE45.

The routines as supplied result in the message 'SINGULARITY LIKELY' because of a too large initial Δt (one hundredth of $t_{final}-t_{start}$). This is corrected using the approach shown in the following instructions:

```
% First integrate using the ODE23 routine.
t0 = 0;
tf = 0;
dt = 0.1;
x0 = [9.975 1.674 84.99]';
ts = clock;
tol= 1.0e-4;
tra= 1; % Trace the integration on the screen.
while tf <= 10.0,
        t0 = tf ;
        tf = t0 + dt ;
        if t0 > 0.01, tf = t0 + 9*dt; end
        if t0 > 0.99, tf = t0 + 10*dt; end
        diary off;
        [t,x1] = ode23('loduebl',t0,tf,x0,tol,tra);
        diary on;
        axis([-4 1 -3 2]);
        loglog(t,x1),...
        title('Lithium Cluster Dynamics under Electron Bombardement'),...
        xlabel(time, s'),ylabel('Cluster Concentrations'),...
        pause(10),hold on;
        x0 = x1(length(x1),:)';
        clear t x1;
end;
el = etime(clock,ts)
```

From table 1 it is evident, that 386-MATLAB is not a time efficient simulation tool, even though it does get the task done with high accuracy.

Simulations were also performed for five logarithmically spaced values of l_f from 100 to 10000. The results are shown in figure 1 as a double logarithmic plot of f versus t. This task was performed overnight and lasted 13,970 seconds including plotting.

Integration Method	File Type	Elapsed Time	Tolerance
ODE23	MEX-file	739s	10^{-4}
ODE45	MEX-file	563s	10^{-5}
ODE45	MEX-file	752s	10^{-6}
ODE45	MEX-file	579s	10^{-7}

Table 1: Comparison of simulation times for task a. The elapsed time includes display of t, Δt and x on the screen every integration interval, and for the first and second also plotting of the results on the screen.

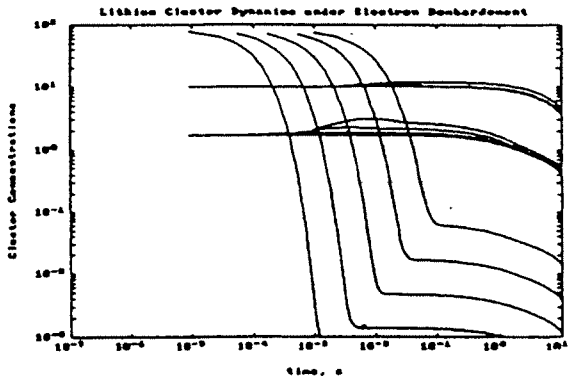


Fig. 1: r, m and f as a function of time for different values of l_f

The steady states for l_f = 1000 and two values of p are shown in table 2. The results were obtained with the following MATLAB instructions:

```
% Now calculate the steady states for two different values of p.
details = zeros(16,1);

details(1,1) = 2; % Collect statistical information on the solution.
% First solve for p = 0 - the trivial solution satisfies this case.
fpar(1,7)=0;
ts=clock;
[x1,termcode] = njfsolve('loduеb2',[1 1 1]',details,fpar)
e6=etime(clock,ts)
% Then for p = 10000.
fpar(1,7)=10000;
ts=clock;
[x2,termcode] = njfsolve('loduеb2',[1 1 1]',details,fpar)
e7=etime(clock,ts)
```

Inspection of the model reveals, that for p = 0 the origin is a solution to the steady state problem. The iterative solution of the steady state equations in this case gave better results and was much faster.

p_c	r_{ss}	m_{ss}	f_{ss}
0	≈0	≈0	≈0
10000	1000	10	10

Table 2: Calculation of steady states for different bombardement rates

Conclusion: Even though the problem could be solved using MATLAB the simulations took a large amount of time and several tricks were needed to work around array size limitation, especially using PC-MATLAB. However, a special simulation tool called SIMULAB has been developed with good interfaces to MATLAB. Both MATLAB and SIMULAB are developed by The MathWorks, Inc., and MATLAB has become the defacto standard for many applications within control engineering and signal processing.

Niels Jensen The PDDC Group, Department of Chemical Engineering, Technical University of Denmark; Lyngby, Denmark

Comparison 1 - 386-MATLAB

Niels Jensen
The PDDC Group, Department of Chemical Engineering, Technical University of Denmark
Lyngby, Denmark

1 Introduction

MATLAB is a C-based general tool for mathematical and engineering calculations with limited capabilities for simulation of non-linear equation systems. Versions are available for many personal computers and workstations and for the Cray super computer.

2 Model Description

The model as given in reference [1] may be transformed to the following vector/matrix equation

$$\frac{dx}{dt} = Ax + Bu \tag{1}$$

with $x^t = [r, m, f]$, $u^t = [mf, f^2, p]$ and

$$A = \begin{bmatrix} -d_r & 0 & 0 \\ d_r & -d_m & 0 \\ d_r & 2d_m & -l_f \end{bmatrix} \quad B = \begin{bmatrix} k_r & 0 & 0 \\ -k_r & k_f & 0 \\ -k_r & -2k_f & 1 \end{bmatrix} \tag{2}$$

and it is implemented in the following m-file:

```
function xs = lcdueb1(x)
p = par(1,7);
A = par(1:3,1:3);
B = par(1:3,4:6);
u = [x(2)*x(3); x(3)^2; p];
xs= A*x + B*u;
```

3 Results

All calculations were done on an IBM PS/S Model 80 (80386 processor with an 80387 numeric coprocessor) using 386-MATLAB version . MATLAB contains two variable step integration routines based on the Runge-Kutta method: ODE23 and ODE45. The routines as supplied results in the message 'SINGULARITY LIKELY' because of a too large initial Δt (one hundredth of $t_{final} - t_{start}$). This is corrected using the approach shown in the following instructions and the results in table 1 were obtained:

Integration Method	File Type	Elapsed Time	Tolerance
ODE23	MEX-file	739s	10^{-4}
ODE45	MEX-file	563s	10^{-5}
ODE45	MEX-file	752s	10^{-6}
ODE45	MEX-file	579s	10^{-7}

Table 1: Comparison of simulation times for task a. The elapsed time includes display of t, Δt and x on the screen every integration interval, and for the first and second also plotting of the results on the screen.

```
%       First integrate using the ODE23 routine.
t0 = 0;
tf = 0;
dt = 0.1;
x0 = [9.975 1.674 84.99]';
ts = clock;
tol= 1.0e-4;
tra= 1;  % Trace the integration on the screen.
while tf <= 10.0,
  t0 = tf;
  tf = t0 + dt;
  if t0 > 0.01, tf = t0 + 9*dt; end
  if t0 > 0.99, tf = t0 + 10*dt; end
  diary off;
  [t,x1] = ode23('lcdueb1',t0,tf,x0,tol,tra);
  diary on;
  axis([-4 1 -3 2]);
  loglog(t,x1),...
  title('Lithium Cluster Dynamics under Electron Bombardement'),...
  xlabel('time, s'),ylabel('Cluster Concentrations'),...
  pause(10),hold on;
  x0 = x1(length(x1),:)';
  clear t x1;
end;
e1 = etime(clock,ts)
```

From the tabel it is evident, that 386-MATLAB is not a time efficient simulation tool, even though it does get the task done with high accuracy. Simulations were also performed for five logaritmically space values of l_f from 100 to 10000. The results are shown in figure 1 as a double logaritmic plot of f versus t. This task was performed overnigth and lasted 13,970 seconds including plotting.

The steady states for $l_f = 1000$ and two values of p are shown in table 2. The results were obtained with the following MATLAB instructions

```
%     Now calculate the steady states for two different values of p.
details       = zeros(16,1);
```

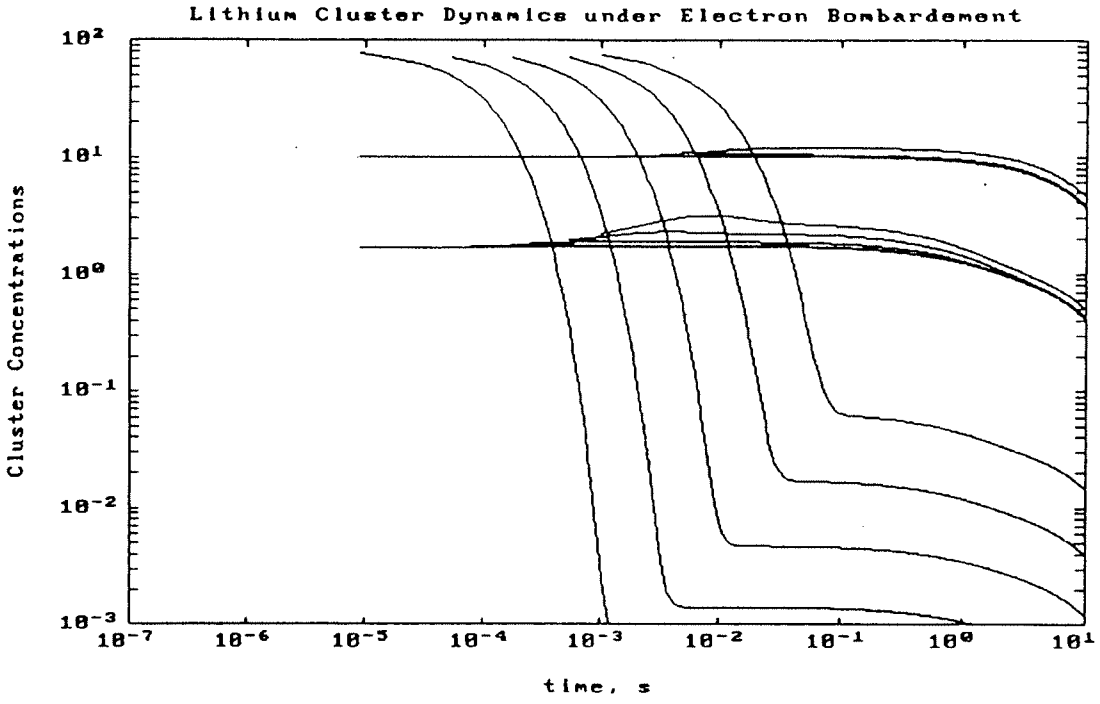


Figure 1: r,m and f as a function of time for different values of l_f.

p_c	r_{ss}	m_{ss}	f_{ss}
0	≈ 0	≈ 0	≈ 0
10000	1000	10	10

Table 2: Calculation of steady states for different electron bombardement rates.

```
details(1,1)  = 2;  % Collect statistical information on the solution.
%
% First solve for p = 0 - the trivial solution satisfies this case.
fpar(1,7)=0;
ts=clock;
[x1,termcode] = njfsolve('lcdueb2',[1 1 1]',details,fpar)
e6=etime(clock,ts)
% Then for p = 10000.
fpar(1,7)=10000;
ts=clock;
[x2,termcode] = njfsolve('lcdueb2',[1 1 1]',details,fpar)
e7=etime(clock,ts)
```

Inspection of the model reveals, that for $p = 0$ the origin is a solution to the steady state problem. The iterative solution of the steady state equations in this case gave better results and was much faster (< 20 seconds) than simulation until a steady state was reached.

4 Conclusion

Even though the problem could be solved using MATLAB-386 and even PC-MATLAB the simulations took a large amount off time and several trick were needed to work around array size limitation, especially using PC-MATLAB. However, a special simulation tool called SIMULAB has been developed with good interfaces to MATLAB. Both MATLAB and SIMULAB are developed by The MathWorks, Inc., and MATLAB has become the defacto standard for many application within control engineering and signal processing.

References

[1] "Comparison of Simulation Software", EUROSIM - European Simulation News, Number 2, p.20, July 1991.

[2] "MATLAB for MS-DOS Personal Computers - User's Guide", The MathWorks, Inc., South Natick, MA 01760, U.S.A., 1989.

Comparison 1 - SIMULAB

SIMULAB is a general purpose nonlinear dynamic simulation package which has been written as an extension to the widely used MATLAB software for scientific and engineering, numerical calculations. It is available to run under X-Windows on a wide range of Workstations, on Macintosh and will shortly be released for 386 PCs.

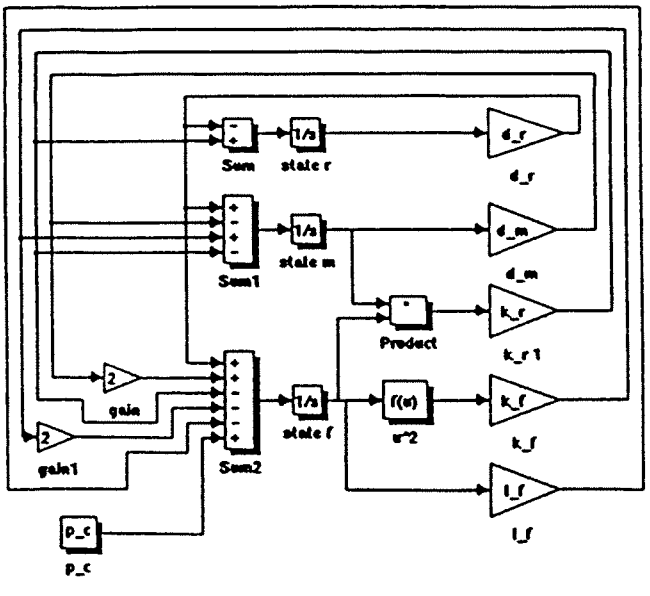
SIMULAB and MATLAB together provide a complete environment for both model development and simulation. Models may be developed in a block diagram window or as a file containing differential and algebraic equations, or using a combination of these two approaches. A block diagram model can be constructed within the menu and mouse driven environment by selecting and connecting up blocks from the standard libraries. If the required block is not in the SIMULAB library then it is usually an easy task to either customise an existing block or design a completely new one.

The menu driven interface generally provides the most rapid route for prototyping and model development but when traceability is important or for complex simulation runs, any of the menu or mouse commands may be run instead from the MATLAB command line or from a command file.

Flexibility and extendability are key features of the package, by using MATLAB function files, the user can automate simulation runs or even write new integration or analysis functions. All models are stored as text files allowing them to be easily transferred between different machine architectures and apart from available virtual memory, there is no limit to model size or complexity.

Model Description

The following figure shows a block diagram description of the Lithium-cluster model as implemented in SIMULAB. The parameter values are stored in the MATLAB workspace allowing successive simulation runs with varying parameter values to be performed with ease.



Lithium-cluster dynamics under electron bombardment

Results

All calculations were performed on a Sun 4 Workstation running under X-Windows.

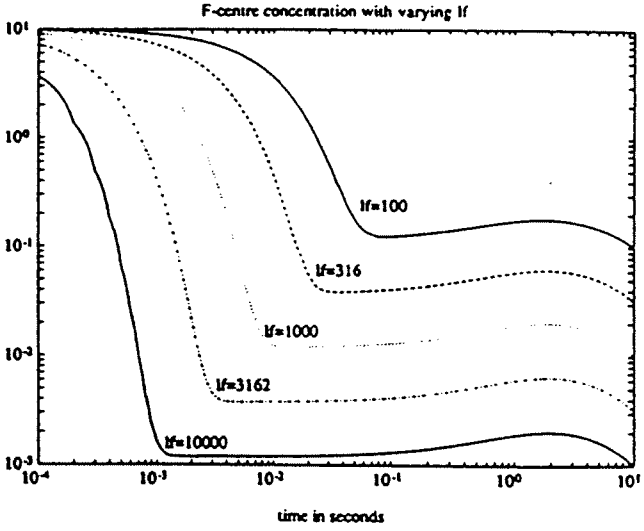
Comparison of computer time (task a)

Simulation runs with the various integration algorithms were all performed with variable step length algorithms, a relative error tolerance of 1e-3 and the minimum and maximum allowed step lengths set to 0.0001 and 1 seconds respectively. The table gives the simulation time for each method as well as the number of integration steps required to achieve the specified tolerance. The Linsim method is one which extracts the linear dynamics of a system leaving only the nonlinear dynamics to be simulated. This method is extremely efficient when the system to be simulated is linear or nearly linear.

Integration method	number of integration steps	computation time in seconds
RK fifth order	2732	10.40
Gear	47	0.37
Linsim	87	0.19
Adams	7363	45.80

Parameter variation of l_f (task b)

The system was simulated over 10 seconds with values of l_f equal to 100, 316, 1000, 3162 and 10000. The following shows a plot with a logarithmic scale on both axes of the variation of the concentration of F-centres against time.



Calculation of steady state (task c)

SIMULAB provides a trim function which allows rapid and straightforward calculation of the steady state. The following command returns the equilibrium value of the state vector x (as well as values of inputs, outputs and state derivatives)

```
[x u y dx] = trim('lithium_model')
```

The values for the individual states are:

p	r	m	f
1e4	1000	10	10
0	0	0	0

David Maclay, IAS Cambridge Control, Jeffereys Building, Cowley Road, Cambridge CB4 4WS, England. Tel: +44 (0)223 420722.

File: esm_report.txt Printed Tue Oct 15 17:49:24 1991 Login: DAVID Page: 1

Simulation of Lithium-Cluster Example using SIMULAB

SIMULAB is a general purpose nonlinear dynamic simulation package which has been written as an extension to the widely used MATLAB software for scientific and engineering, numerical calculations. It is available to run under X-Windows on a wide range of Workstations, on Macintosh and will shortly be released for 386 PCs.

SIMULAB and MATLAB together provide a complete environment for both model development and simulation. Models may be developed in a block diagram window or as a file containing differential and algebraic equations, or using a combination of these two approaches. A block diagram model can be constructed within the menu and mouse driven environment by selecting and connecting up blocks from the standard libraries. If the required block is not in the SIMULAB library then it is usually an easy task to either customise an existing block or design a completely new one.

The menu driven interface generally provides the most rapid route for prototyping and model development but when traceability is important or for complex simulation runs, any of the menu or mouse commands may be run instead from the MATLAB command line or from a command file.

Flexibility and extendability are key features of the package, by using MATLAB function files, the user can automate simulation runs or even write new integration or analysis functions. All models are stored as text files allowing them to be easily transferred between different machine architectures and apart from available virtual memory, there is no limit to model size or complexity.

Model Description

Figure 1 shows a block diagram description of the Lithium-cluster model as implemented in SIMULAB. The parameter values are stored in the MATLAB Workspace allowing succesive simulation runs with varying parameter values to be performed with ease.

Results

All calculations were performed on a Sun 4 Workstation running under X-Windows.

Comparison of computer time (task a)

Simulation runs with the various integration algorithms were all performed with variable step length algorithms, a relative error tolerance of 1e-3 and the minimum and maximum allowed step lengths set to 0.0001 and 1 seconds respectively. The table gives the simulation time for each method as well as the number of integration steps required to achieve the specified tolerance. The linsim method is one which extracts the linear dynamics of a system leaving only the nonlinear dynamics to be simulated. This method is extremely efficient when the system to be simulated is linear or nearly linear.

Integration Method	Number of integration steps	Computation time in seconds
RK fifth order	2732	10.4
Gear	47	0.37
Linsim	87	0.19
Adams	7363	45.8

Parameter variation of l_f (task b)

The system was simulated over 10 seconds with values of l_f equal to 100, 316, 1000, 3162 and 10000. Plot 1 shows a plot with a logarithmic scale on both axes of the variation of the the concentration of F-centres against time.

Calculation of steady state (task c)

SIMULAB provides a trim function which allows rapid and straightforward calculation of the steady state. The following command returns the equilibrium value of the state vector x (as well as values of inputs, outputs and state derivatives)

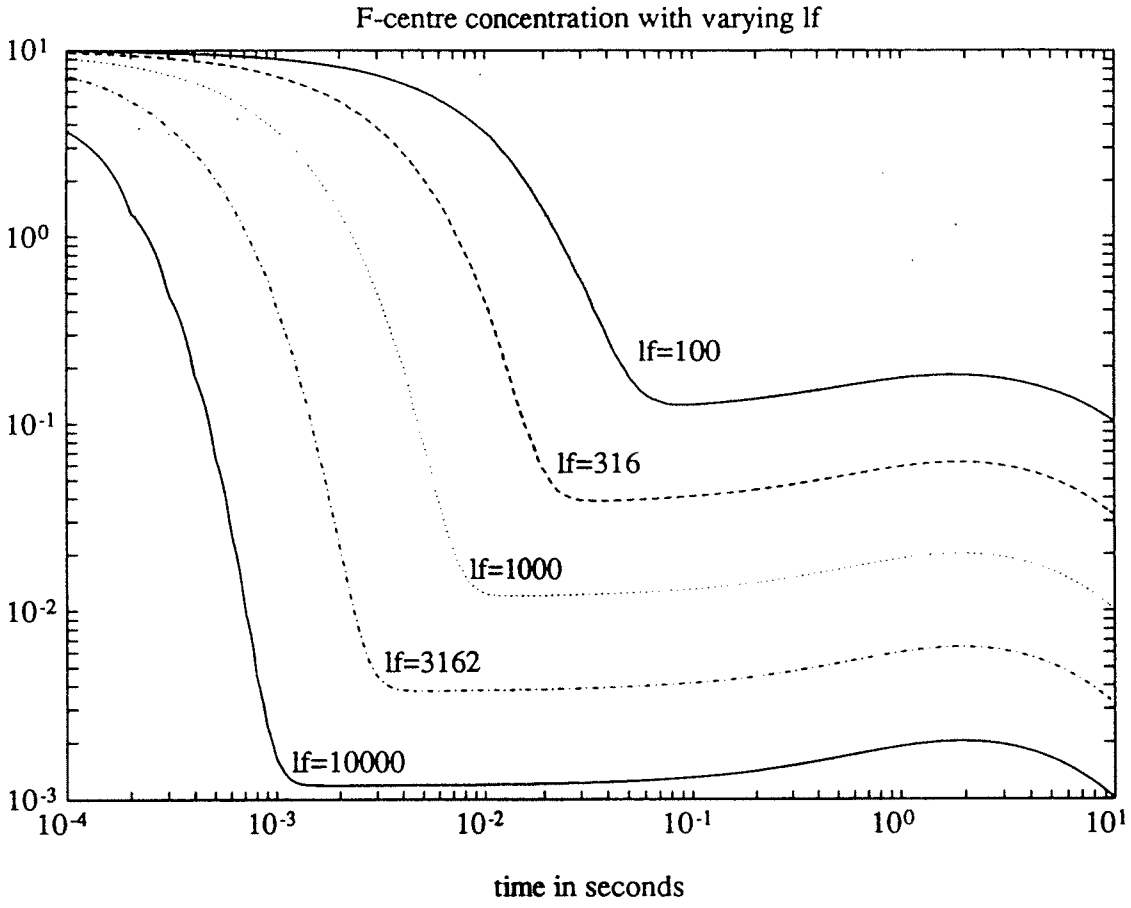
```
[x u y dx] = trim('lithium_model')
```

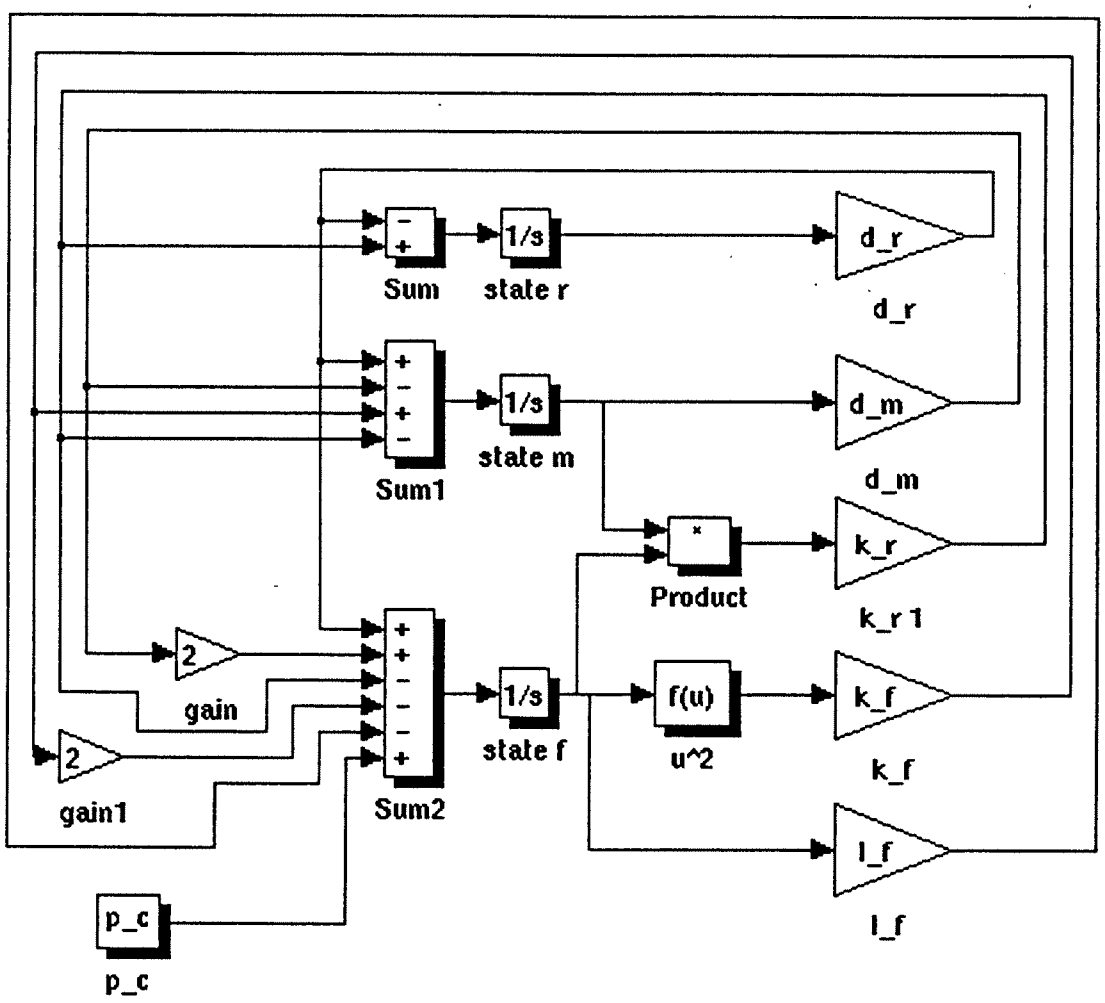
The values for the individual states are:

p	r	m	f
1e4	1000	10	10
0	0	0	0

File: esm_report.txt Printed Tue Oct 15 17:49:24 1991 Login: DAVID Page: 2

```
81
82
83      David Maclay, IAS Cambridge Control, Jefferey's Building, Cowley Road,
84      Cambridge CB4 4WS, England. tel +44 (0)223 420722.
85
```


F-centre concentration with varying lf
10^1
10^0
10^{-1}
10^{-2}
10^{-3}
10^{-4}
10^{-3}
10^{-2}
10^{-1}
10^0
10^1
lf=100
lf=316
lf=1000
lf=3162
lf=10000
time in seconds



Lithium-cluster dynamics under electron bombardment

Comparison 1 - DYNAST

About DYNAST

DYNAST is a package for solving sets of nonlinear implicit-form algebro-differential equations as well as for analysis of block and/or port diagrams, which can be submitted also in a graphical form.

The advantage of the port diagrams stems from the fact that their structure corresponds directly to the structure of the modeled real dynamic systems. Models of fairly complex systems can be set up from submodels of real components stored in DYNAST submodel libraries in a kit-like way.

No compilation of problem specification is required and any algebraic loops in the diagrams make no problems. For linear or automatically linearized diagrams, DYNAST provides also frequency analysis and yields both the time- and frequency-domain results in a semi-symbolic form.

The IBM PC version is supported by a graphical user interface and documentation environment based on OrCAD, AutoCAD and TeX systems. There are DYNAST versions for eight-bit CP/M computers, minicomputers and mainframes.

DYNAST has been around for about six years and it is used already by numerous academic as well as industrial institutions for applications ranging from design problems in various engineering disciplines up to medicine diagnostics and economic predictions.

DYNAST is distributed by DYN, Nad lesikem 27, CS-160 00 Prague 6, CSFR, Tel: +42-2-311 79 04.

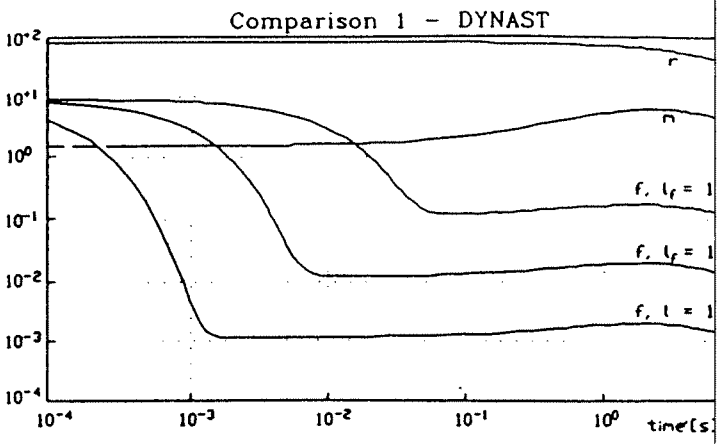
Results

All the tasks of the comparison 1 problem can be solved in one DYNAST run when specifying them by the following input data:

```
*SYSTEM; *: EUROSIM Comparison 1
kr = 1; kf = .1; lf = 1000; dr = .1; dm = 1; p = 0;
SYSVAR r, m, f;
0 = - VD.r - dr*r + kr*m*f;
0 = - VD.m + dr*r - dm*m + kf*f**2 - kr*m*f;
0 = - VD.f + dr*r + 2*dm*m - kr*m*f - 2*kf*f**2 - lf*f + p;
*TR; TR 0 12;                    :transient analysis for 0 < t < 12
INIT f = 9.975, m = 1.674, r = 84.99;
PRINT r, m, f; RUN eps = 1E-6;
MODIFY lf = 1E2;
INIT f = 9.975, m = 1.674, r = 84.99; RUN eps = 1E-6;
MODIFY lf = 1E4;
INIT f = 9.975, m = 1.674, r = 84.99; RUN eps = 1E-6;
RESET;
DC; PRINT r, m, f;               :steady-state analysis
MODIFY lf = 1E3; RUN;
MODIFY p = 1E4; RUN; *END;
```

The following plot displays results obtained for the tasks a) and b).

The problem was solved on an IBM AT/386-387 of Norton computing index 30.1 using DYNAST version running both with and without numeric coprocessor.



The transient as well as the steady-state solutions were computed using the same algorithm, which is based on the combination of Gear's and Newton-Raphson's methods modified by Rubner-Petersen.

The task a) was solved in the time interval $0 < t < 12$s for two different values of the permissible relative truncation error: 1E-3 (default value) and 1E-6 (see the input data). In the former case the solution took 60 integration steps (0 of them rejected) and 66 iterations. The computation required 2.25s of CPU time. The last solution vector (at t = 12s) was:

r = 2.60340E+01 m = 2.85984E+00 f = 8.30013E-04

The latter, enhanced accuracy solution took 120 integration steps (3 were rejected) and 141 iterations. After CPU time of 4.45s the last solution vector was:

r = 2.60517E+01 m = 2.86178E+00 f = 8.30576E-03

All the computations were done with the default initial steplength equal to 1E-5 times the specified interval of time. Any decrease of this value did not have any effect on the final solution vector. The solution of task c) took just one iteration and required 0.16s of CPU time for p = 0. It resulted in the vector

r = 0.00000E+00 m = 0.00000E+00 f = 0.00000E+00

For p = 1E4 it took 3 iterations and 0.06s of CPU time only (no resetting of input data was necessary in this case). The result was:

r = 1.00000E+03 m = 1.00000E+01 f = 1.00000E+01

To verify the steady-state analysis results, the differential equations were solved with the permissible error 1E-6 in the interval $0 < t < 2000$s for p = 0 as well as for p = 1E4. The last solution vectors were

r = 3.05780E-14 m = 3.39755E-15 f = 9.85388E-18

and

r = 1.00000E+03 m = 1.00000E+01 f = 1.00000E+01

respectively. The former case statistics was 123 steps, 140 iterations and 8.24s. The latter case asked for 81 steps, 83 iterations and 5.87s.

Herman Mann, Dept. of Mech. Eng. and Robotics, Free University of Brussels, CP 165, Ave Roosevelt 50, B-1050 Brussels, Belgium

Comparison 1 - DYNAST

About DYNAST

DYNAST is a package for solving sets of nonlinear implicit-form algebro-differential equations as well as for analysis of block and/or port diagrams, which can be submitted also in a graphical form.

The advantage of the port diagrams stems from the fact that their structure corresponds directly to the structure of the modeled real dynamic systems. Models of fairly komplex systems can be set up from submodels of real components stored in DYNAST submodel libraries in a kit-like way.

No compilation of problem specification is required and any algebraic loops in the diagrams make no problems. For linear or automatically linearized diagrams, DYNAST provides also frequency analysis and yields both the time- and frequency-domain results in a semisymbolic form.

The IBM PC version is supported by a graphical user interphace and documentation environment based on OrCAD, AutoCAD and TeX systems. There are DYNAST versions for eight-bit CP/M computers, minicomputers and mainframes.

DYNAST has been around for about six years and it is used already by numerous academic as well as industrial institutions for applications ranging from design problems in various engineering disciplines up to medicine diagnostics and economic predictions.

DYNAST is distributed by DYN, Nad lesikem 27, CS-160 00 Prague 6, CSFR, phone: 0042-2-311 79 04.

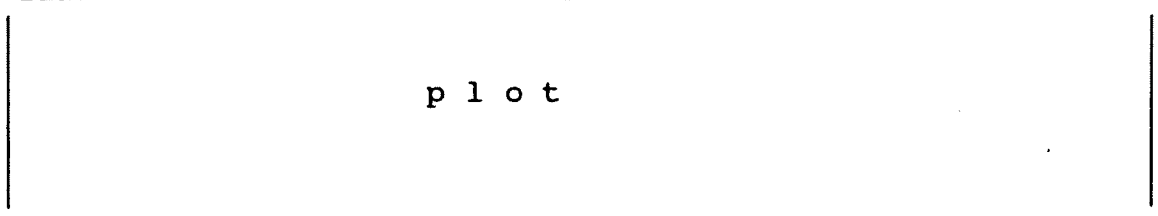
Comparison 1 results

All the tasks of Comparison 1 problem can be solved in one DYNAST run when specifying them by the following input data:

```
*SYSTEM; *: EUROSIM Comparison 1
kr = 1; kf = .1; lf = 1000; dr = .1; dm = 1; p = 0;
SYSVAR r, m, f;
0 = - VD.r - dr*r + kr*m*f;
0 = - VD.m + dr*r - dm*m + kf*f**2 - kr*m*f;
0 = - VD.f + dr*r + 2*dm*m - kr*m*f - 2*kf*f**2 - lf*f + p;
*TR; TR 0 12;              :transient analysis for 0 < t < 12
INIT f = 9.975, m = 1.674, r = 84.99;
PRINT r, m, f; RUN eps = 1E-6;
MODIFY lf = 1E2;
INIT f = 9.975, m = 1.674, r = 84.99; RUN eps = 1E-6;
MODIFY lf = 1E4;
INIT f = 9.975, m = 1.674, r = 84.99; RUN eps = 1E-6;
RESET;
DC; PRINT r, m, f;         :steady-state analysis
MODIFY lf = 1E3; RUN;
MODIFY p  = 1E4; RUN; *END;
```

The following plot displays results obtained for the

tasks a) and b):



The problem was solved on IBM AT/386-387 of Norton computing index 30.1 using DYNAST version running both with and without numeric coprocessor.

The transient as well as the steady-state solutions were computed using the same algorithm, which is based on the combination of Gear's and Newton-Raphson's methods modified by Rubner-Petersen.

The task a) was solved in the time interval $0 < t < 12s$ for two different values of the permisible relative truncation error: 1E-3 (default value) and 1E-6 (see the input data). In the former case the solution took 60 integration steps (0 of them rejected) and 66 iterations. The computation recquired 2.25s of CPU time. The last solution vector (at t = 12s) was:

r = 2.60340E+01 m = 2.85984E+00 f = 8.30013E-04

The latter, enhanced accuracy solution took 120 integration steps (3 were rejected) and 141 iterations. After CPU time of 4.45s the last solution vector was:

r = 2.60517E+01 m = 2.86178E+00 f = 8.30576E-03

All the computations were done with the default initial steplength equal to 1E-5 times the specified interval of time. Any decrease of this value did not have any effect on the final solution vector. The solution of task c) took just one iteration and recquired 0.16s of CPU time for p = 0. It resulted in vector

r = 0.00000E+00 m = 0.00000E+00 f = 0.00000E+00

For p = 1E4 it took 3 iterations and 0.06s of CPU time only (no resetting of input data was necessary in this case). The result was:

r = 1.00000E+03 m = 1.00000E+01 f = 1.00000E+01

To verify the steady-state analysis results, the differential equations were solved with the permissible error 1E-6 in the interval $0 < t < 2000s$ for p = 0 as well as for p = 1E4. The last solution vectors were

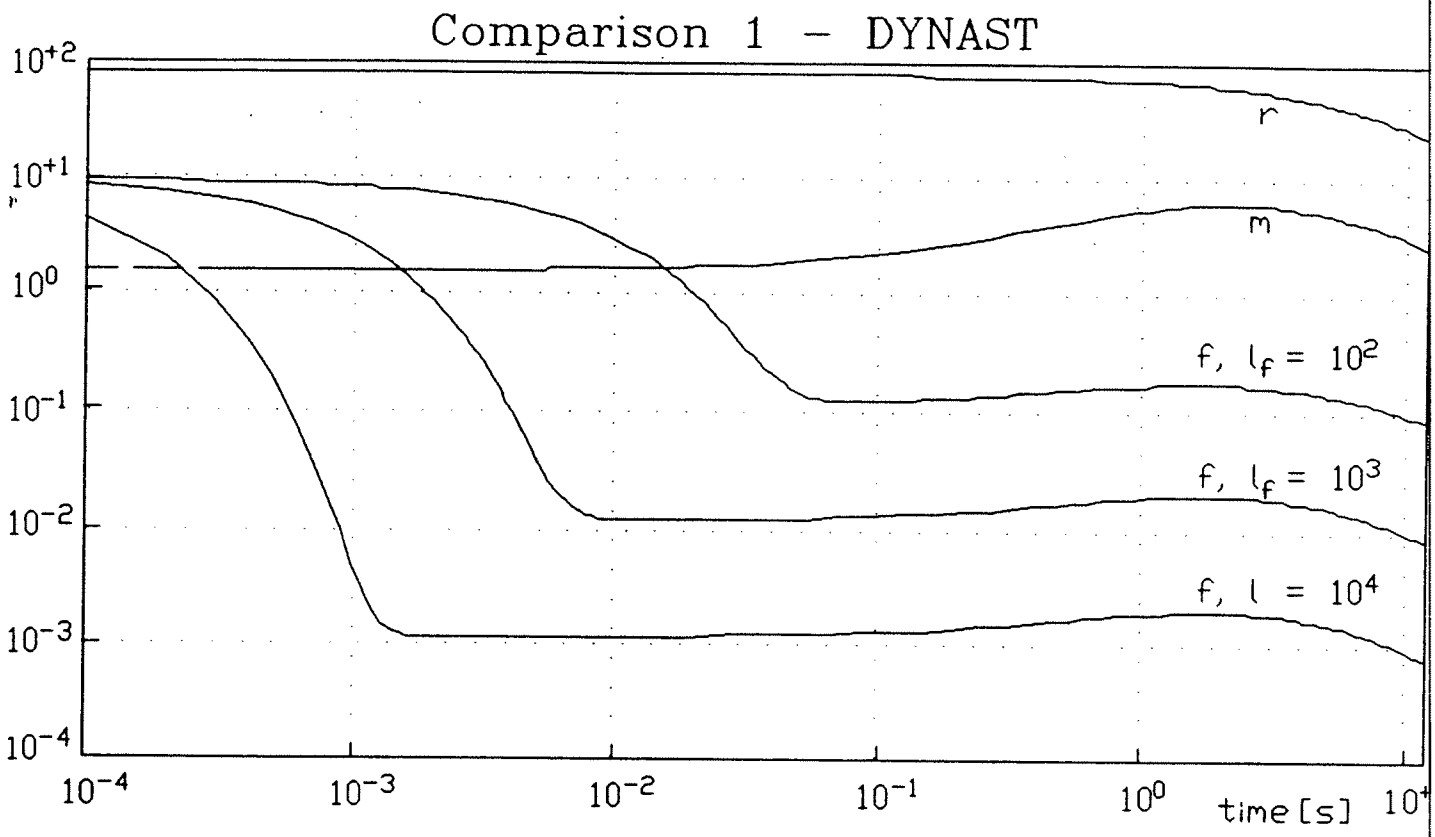
r = 3.05780E-14 m = 3.39755E-15 f = 9.85388E-18

and

r = 1.00000E+03 m = 1.00000E+01 f = 1.00000E+01,

respectively. The former case statistics was 123 steps, 140 iterations and 8.24s. The latter case asked for 81 steps, 83 iterations and 5.87s.

Herman Mann,
Dept. of Mech. Eng. and Robotics
Free University of Brussels, CP 165,
Ave Roosevelt 50, B-1050 Brussels, Belgium


Comparison 1 – DYNAST
10+2
10+1
10^0
10^-1
10^-2
10^-3
10^-4
r
m
f, l_f = 10^2
f, l_f = 10^3
f, l = 10^4
10^-4
10^-3
10^-2
10^-1
10^0
time [s]
10+

Comparison 1 - PROSIGN

PROSIGN (Process Design) is a software package designed for the simulation of continuous and discrete time nonlinear systems with a free number of inputs and outputs.

Modelling may be carried out in three different ways:

- graphically- -block oriented
  (based on the Standard-Library)
- graphically- - component oriented
  (based on libraries like Mechanic, Electric, ...)
- textual - equation oriented
  (based on PSL, the PROSIGN Simulation Language)

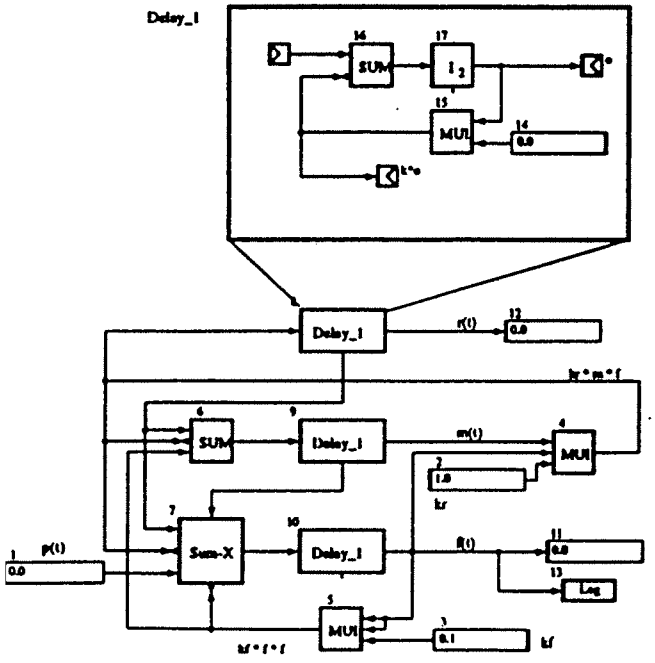
Since all methods can be combined, the use needn't choose one method. Using PROSIGN, modelling is always done in that way which is most time saving and most obvious with respect to the simulation problem to be solved.

PROSIGN works with fixed or variable step size, alternatively. In the variable case the calculations are performed with a userdefinable degree of accuracy.

A special feature of PROSIGN is the code generator producing Modula-2, Fortran or C codes.

Model description

The Lithium-Cluster model is built with elements taken from the PROSIGN standard library. The resulting block diagram is shown in the following figure:



Results

a) Computing Time:

Computing time depends on the integration method and the step size control. PROSIGN offers 8 methods of different orders which may be used with fixed or variable step size.

Here the variable case is chosen. The computing time for a 10 seconds simulation time is shown for 2 integration methods in the table below:

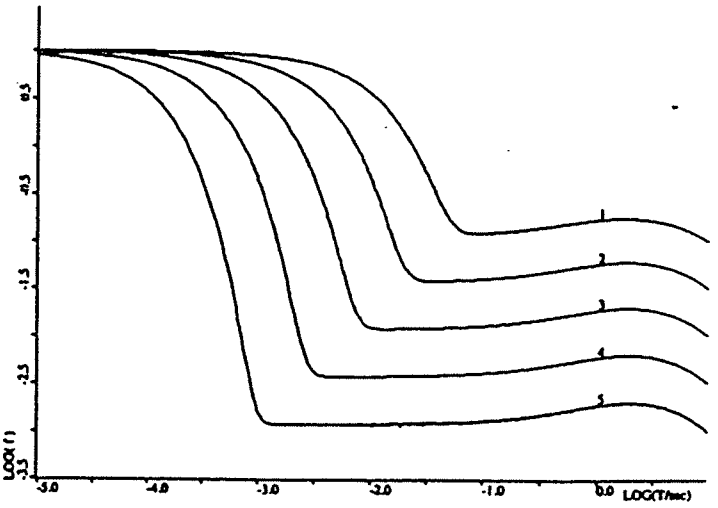
Algorithm	max. step size	computing time (sec)
2nd order (Simpson)	0.001	470
4th order (Adams-Bashforth)	0.0025	204

The generated Fortran program reduces the simulation time to 40 sec using the fixed step size 0.0005 sec in conjunction with the Simpson integration method.

b) Variation of Parameter l_f:

The following figure presents the results of the F-centre concentration against time.

Curve	lf
1	100
2	316.2
3	1000
4	3162
5	10000

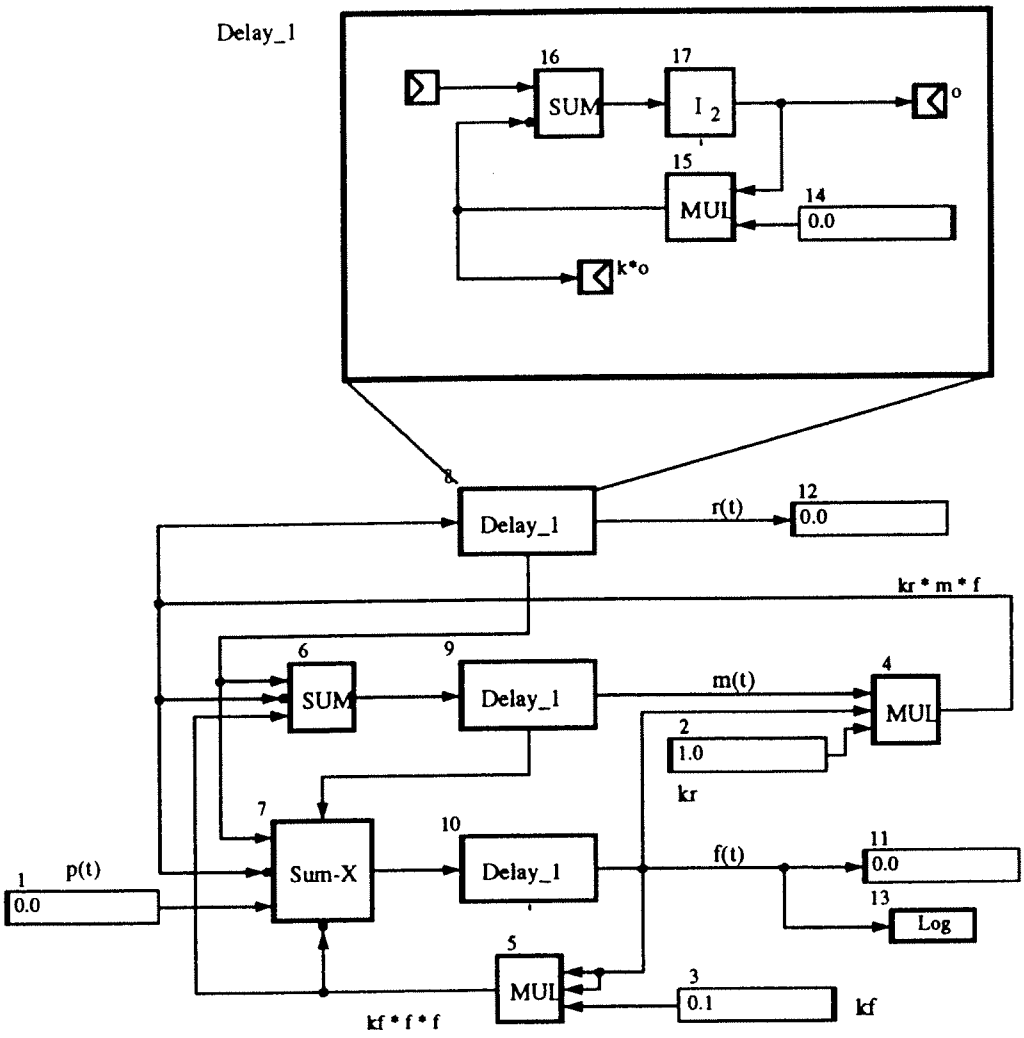


c) Steady states:

The steady state values directly result from a PROSIGN steady state model. They are summarized in the following table:

p	r	m	f
0	0	0	0
10000	1000	10	10

Helmuth Stahl, R&O Software-Technik GmbH, Planegger Straße 16-18, D-8034 Germering, Tel.: +49-(0)89 - 840080, Fax: +49-(0)89 - 8400813


Delay_1
16
SUM
17
I 2
o
15
MUL
14
0.0
k*o
8
Delay_1
r(t)
12
0.0
kr * m * f
6
SUM
9
Delay_1
m(t)
4
MUL
2
1.0
kr
7
Sum-X
10
Delay_1
f(t)
11
0.0
13
Log
1
p(t)
0.0
5
MUL
3
0.1
kf
kf * f * f

Comparison 1 - DESIRE

DESIRE/387, DESIRE/387 for AT clones and the newer DESIRE/X are direct-executing dynamic-system simulation packages which compile readable, screen-edited programs directly into memory in a small fraction of a second, so that there are no annoying translation delays. Programs admit up to 1500 state variables and can be in matrix form. DESIRE/NEUNET and DESIRE/X also solve neural-network programs.

For smooth integration with a logarithmic time scale, we replaced each given differential equation

dx/dtime = expression with dx/dt = expression*tt

where

$$tt = \ln(10)*10^{(t+to)}$$

is the time, and the new independent variable

$$t = \log(time) + t_o$$

produces a logarithmic time scale shifted by any desired amount t_o.

In our graphs, $t_o = 3$, so that

the abscissa marker 0 corresponds to time = 0.001
the abscissa marker 2 corresponds to time = 0.1
the abscissa marker 4 corresponds to time = 10

The program listings and graphs below are direct EGA screen prints obtained with a personal computer; VGA output is also available. If you need more elaborate graphs, you can make programmed or command-mode calls to commercially available graph-plotting programs without leaving DESIRE.

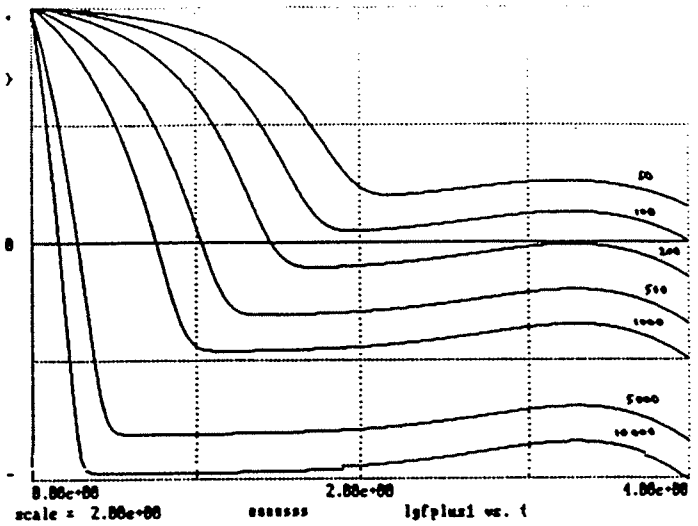
The time taken to produce the first curve on CRT was

14 sec on a cache-less 16 MHz 80386/7 (Toshiba 5100)
30 sec on a 12-MHz 80286/7 AT clone
2.2 sec on a 40 MHz SUN 4c workstation (XWindow graphics)

With display turned off, these computation times decreased to 10 sec, 22 sec and 1.7 sec, respectively. 14 different integration rules can be selected. Gear integration produced results more quickly than fixed- or variable-step Runge-Kutta methods in spite of the fact that the latter are written in assembly language individually optimized for the 80386/7 and 80286/7, while the Gear routine is in PASCAL. The entire SUN program is written in C.

```
--                                   EUROSIM COMPARISON PROBLEM 1
-------------------------------------------------------------------
ln10=ln(10) ;  loge=1/ln10
kr=1 ;  kf=0.1 ;  dr=0.1 ;  dm=1 ;  --          coefficients
f=9.975 ;  m=1.674 ;  r=84.99 ;  --           initial values
p=0
-------------------------------------------------------------------
irule 15 ;  ERRMAX=0.00001 ;  --              GEAR integration
t0=3 ;  --                                 shift log time scale
TMAX=1+t0 ;  NN=6000 ;  DT=0.000001 ;  scale=2
-------------------------------------------------------------------
lf=50 ;  drunr ;  display 2 ;  --                 run and reset
lf=100 ;  drunr ;  lf=200 ;  drunr ;  lf=500 ;  drunr
lf=1000 ;  drunr ;  lf=5000 ;  drunr ;  lf=10000 ;  drun
-------------------------------------------------------------------
DYNAMIC
-------------------------------------------------------------------
A=kr*m*f-dr*r ;  --            we precompute these for speed!
B=kf*f*f-dm*m
tt=ln10*(10^(t-t0)) ;  --                logarithmic time scale
--
d/dt r=A*tt ;  d/dt m=(B-A)*tt
d/dt f=(p-lf*f-A-2*B)*tt
-------------------------------------------------------------------
lgfplus1=loge*ln(f)+1 ;  dispt lgfplus1
```

program listing



results (direct EGA screen prints)

G.A. and T.M. Korn Industrial Consultants, Rt 1, Box 96C, Chelan, WA 98816, USA.

DESIRE Solution of the EUROSIM Comparison I Problem

DESIRE/387, DESIRE/387 for AT clones [1] and the newer DESIRE/X are direct-executing dynamic-system simulation packages which compile readable, screen-edited programs directly into memory in a small fraction of a second, so that there are no annoying translation delays. Programs admit up to 1500 state variables and can be in matrix form. DESIRE/NEUNET and DESIRE/X also solve neural-network programs.

For smooth integration with a logarithmic time scale, we replaced each given differential equation

dx/dtime = expression with dx/dt = expression*tt

where

$$tt = \ln(10)*10^{(t+t0)}$$

is the time, and the new independent variable

$$t = \log(time) + t0$$

produces a logarithmic time scale shifted by any desired amount t0.

In our graphs, t0 = 3, so that

the abscissa marker 0 corresponds to time = 0.001
the abscissa marker 2 corresponds to time = 0.1
the abscissa marker 4 corresponds to time = 10

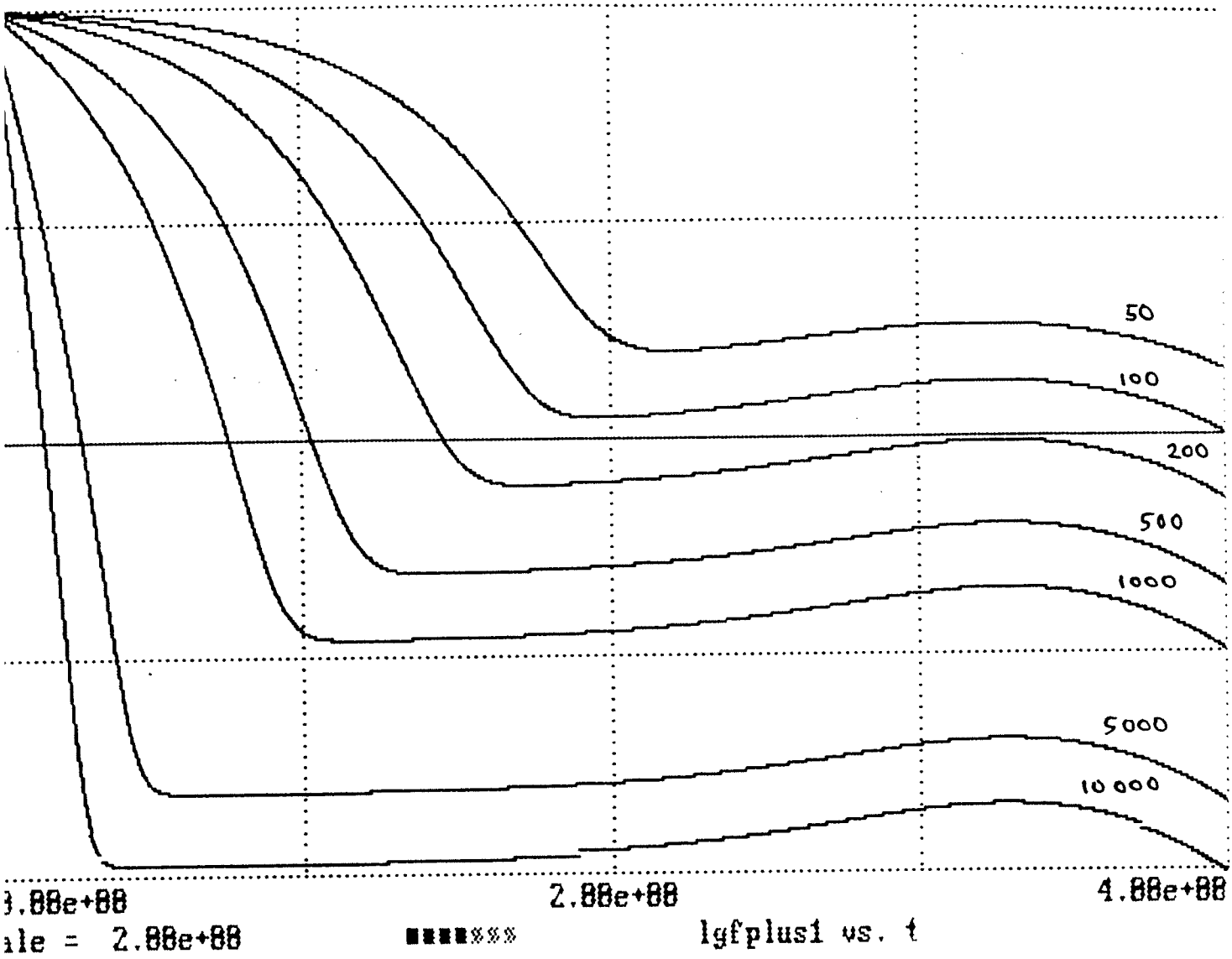
The program listing and graphs below are direct EGA screen prints obtained with a personal computer; VGA output is also available. If you need more elaborate graphs, you can make programmed or command-mode calls to commercially available graph-plotting programs without leaving DESIRE.

The time taken to produce the first curve on the CRT was

14 sec on a cache-less 16 MHz 80386/7 (Toshiba 5100)
30 sec on a 12-MHz 80286/7 AT clone
2.2 sec on a 40-MHz SUN 4c workstation (XWindow graphics)

With displays turned off, these computation times decreased to 10 sec, 22 sec, and 1.7 sec, respectively. 14 different integration rules can be selected. Gear integration produced results more quickly that fixed- or variable-step Runge-Kutta methods in spite of the fact that the latter are written in assembly language individually optimized for the 80386/7 and 80286/7, while the Gear routine is is in PASCAL. The entire SUN program is written in C.

G.A. and T.M.Korn Industrial Consultants
Rt 1, Box 96C, Chelan, WA 98816




```
--                                        EUROSIM COMPARISON PROBLEM 1
-------------------------------------------------------------------------
ln10=ln(10) ¦  loge=1/ln10
kr=1 ¦  kf=0.1 ¦  dr=0.1 ¦  dm=1 ¦  --                    coefficients
f=9.975 ¦  m=1.674 ¦  r=84.99 ¦  --                     initial values
p=0
-------------------------------------------------------------------------
irule 15 ¦  ERRMAX=0.00001 ¦  --                        GEAR integration
t0=3 ¦  --                                          shift log time scale
TMAX=1+t0 ¦  NN=6000 ¦  DT=0.000001 ¦  scale=2
-------------------------------------------------------------------------
lf=50 ¦  drunr  ¦  display 2 ¦  --                        run and reset
lf=100 ¦  drunr  ¦  lf=200 ¦  drunr  ¦  lf=500 ¦  drunr
lf=1000 ¦  drunr  ¦  lf=5000 ¦  drunr  ¦  lf=10000 ¦  drun
-------------------------------------------------------------------------
DYNAMIC
-------------------------------------------------------------------------
A=kr*m*f-dr*r ¦  --                  we precompute these for speed!
B=kf*f*f-dm*m
tt=ln10*(10^(t-t0)) ¦  --                       logarithmic time scale
--
d/dt r=A*tt ¦  d/dt m=(B-A)*tt
d/dt f=(p-lf*f-A-2*B)*tt
-------------------------------------------------------------------------
lgfplus1=loge*ln(f)+1 ¦  dispt lgfplus1
```

DESIRE Solution of the EUROSIM Comparison I Problem

DESIRE/387, DESIRE/387 for AT clones and the newer DESIRE/X are direct-executing dynamic-system simulation packages which compile readable, screen-edited programs directly into memory in a small fraction of a second, so that there are no annoying translation delays. Programs admit up to 1500 state variables and can be in matrix form. DESIRE/NEUNET and DESIRE/X also solve neural-network programs.
For smooth integration with a logarithmic time scale, we replaced each given differential equation

dx/dtime = expression with dx/dt = expression*tt

where

tt = ln(10)*10^(t+t0)

is the time, and the new independent variable

t = log(time) + t0

produces a logarithmic time scale shifted by any desired amount t0.

In our graphs, t0 = 3, so that
the abscissa marker 0 corresponds to time = 0.001
the abscissa marker 2 corresponds to time = 0.1
the abscissa marker 4 corresponds to time = 10

The program listings and graphs below are direct EGA screen prints obtained with a personal computer; VGA output is also available. If you need more elaborate graphs, you can make programmed or command-mode calls to commercially available graph-plotting programs without leaving DESIRE.

The time taken to produce the first curve on CRT was
14 sec on a cache-less 16 MHz 80386/7 (Toshiba 5100)
30 sec on a 12-MHz 80286/7 AT clone
2.2 sec on a 40 MHz SUN 4c workstation (XWindow graphics)

With display turned off, these computation times decreased to 10 sec, 22 sec and 1.7 sec, respectively. 14 different integration rules can be selected. Gear integration produced results more quickly than fixed- or variable-step Runge-Kutta methods in spite of the fact that the latter are written in assembly language individually optimized for the 80386/7 and 80286/7, while the Gear routine is in PASCAL. The entire SUN program is written in C.

G.A. and T.M. Korn Industrial Consultants
Rt1, Box 96C, Chelan, WA 98816

```
--                                   EUROSIM COMPARISON PROBLEM 1
-------------------------------------------------------------------
ln10=ln(10) |  loge=1/ln10
kr=1 |  kf=0.1 |  dr=0.1 |  dm=1 |  --          coefficients
f=9.975 |  m=1.674 |  r=84.99 |  --           initial values
p=0
-------------------------------------------------------------------
irule 15 |  ERRMAX=0.00001 |  --                GEAR integration
t0=3 |  --                                  shift log time scale
TMAX=1+t0 |  NN=6000 |  DT=0.000001 |  scale=2
lf=50 |  drunr  |  display 2 |  --                run and reset
lf=100 |  drunr  |  lf=200 |  drunr  |  lf=500 |  drunr
lf=1000 |  drunr  |  lf=5000 |  drunr  |  lf=10000 |  drun
-------------------------------------------------------------------
DYNAMIC
-------------------------------------------------------------------
A=kr*m*f-dr*r |  --          we precompute these for speed!
B=kf*f*f-dm*m
tt=ln10*(10^(t-t0)) |  --              logarithmic time scale
--
d/dt r=A*tt |  d/dt m=(B-A)*tt
d/dt f=(p-lf*f-A-2*B)*tt
-------------------------------------------------------------------
lgfplus1=loge*ln(f)+1 |  dispt lgfplus1
```

program listing

results (direct EGA screen prints)

fx, 15.2.

Comparison 1 - EXTEND

Description of EXTEND

EXTEND is a general purpose simulation system supporting both continuous and next event modeling. It is library-based and uses a block diagram approach to modeling. You can use libraries of pre-built blocks to set up models with no programming or you can use MODL (a built-in modeling language) to modify existing blocks or create new ones. One of the EXTEND's built-in libraries is the Generic library, which contains general purpose continuous modeling blocks. The blocks can be grouped by their function: basic math, accumulators, decisions, data input/output, data conversion and model debugging.

In version 1.1 EXTEND doesn't support hierarchical modeling. EXTEND runs on Macintosh computers.

EXTEND™ is a product of Imagine That Inc., 151 Bernal Road, Suite 5, San Jose, CA 95119, USA.

Model descripton

The model is described by blocks of EXTEND's Generic library.

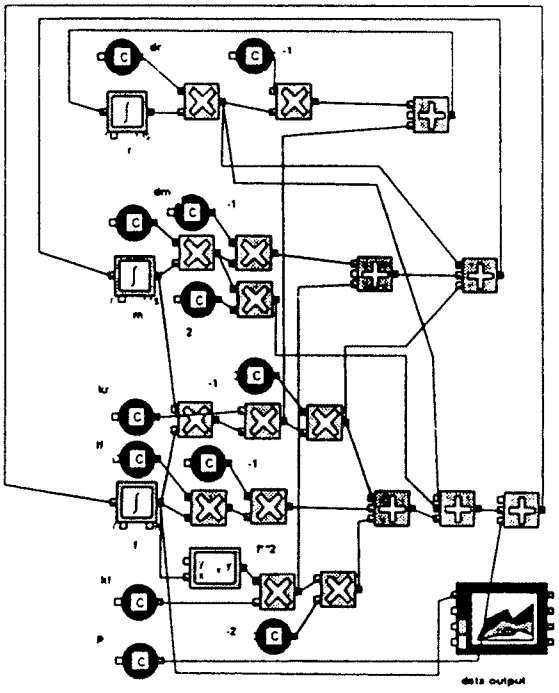


Figure 1

Results

All calculations were done using a Macintosh IIfx.

a) comparison of integration algorithms: The built-in Integrate block of the Generic library supports only two integration methods.

parameter lf= 1000

integration alg.	number of steps time (0,10)	comp. time (min)	numerical
Euler (improved)	10.000	0.5	unstable
Euler (improved)	12.000	1.0	stable
Trapezoidal	20.000	1.45	unstable
Trapezoidal	30.000	2.30	stable

b) variation of parameter lf

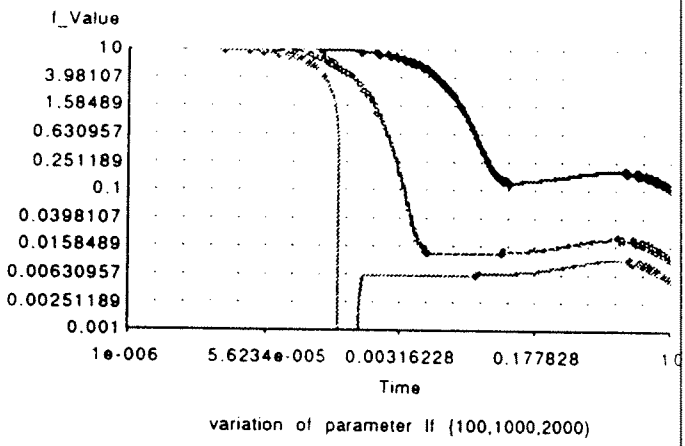


Figure 2

The top curve represents the response for parameter lf=100, with the lower curves showing corresponding results for lf=1000 and 2000 (numerically unstable).

c) calculation of steady states (lf=1000, improved Euler method, number of steps=10000): Figure 3 shows the results of the steady state investigation during constant bombardment (lower curve p(t)=1.0E4) and without bombardment (p(t)=0, numerically unstable).

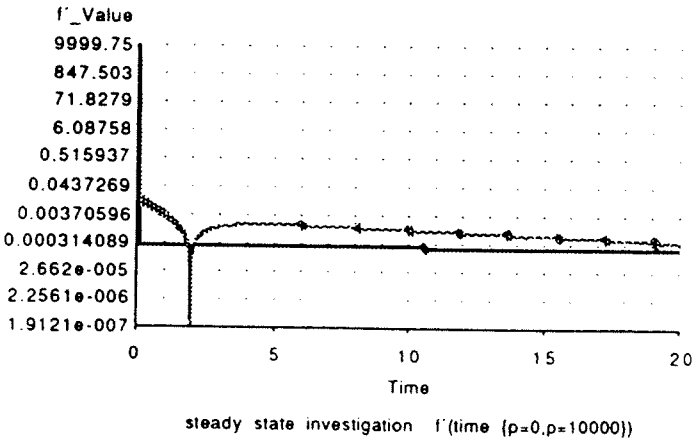


Figure 3

Thorsten Pawletta, Universität Rostock, FB Informatik, Albert-Einstein-Str. 21, D-O-2500 Rostock, Germany; Tel.: +49-(0)381 44424 169;
e-mail: pawel@informatik.uni-rostock.de

Comparison 1 - EXTEND

Description of EXTEND

EXTEND is a general purpose simulation system supporting both continous and next event modeling. It is library-based and uses a block diagram approach to modeling. You can use libraries of pre-built blocks to set up models with no programming or you can use MODL (a built-in modeling language) to modify existing blocks or create new ones. One of the EXTEND´s built-in libraries is the Generic library, which contains general purpose continous modeling blocks. The blocks can be grouped by their function: basic math, accumulators, decisions, data input /output, data conversion and model debugging.
In version 1.1 EXTEND doesn´t support hierarchical modeling.
EXTEND runs on Macintosh computer.
EXTEND™ is a product of Imagine That Inc., 151 Bernal Road, Suite 5, San Jose, CA 95119 USA.

Model descripton

The model is described by blocks of EXTEND´s Generic library.

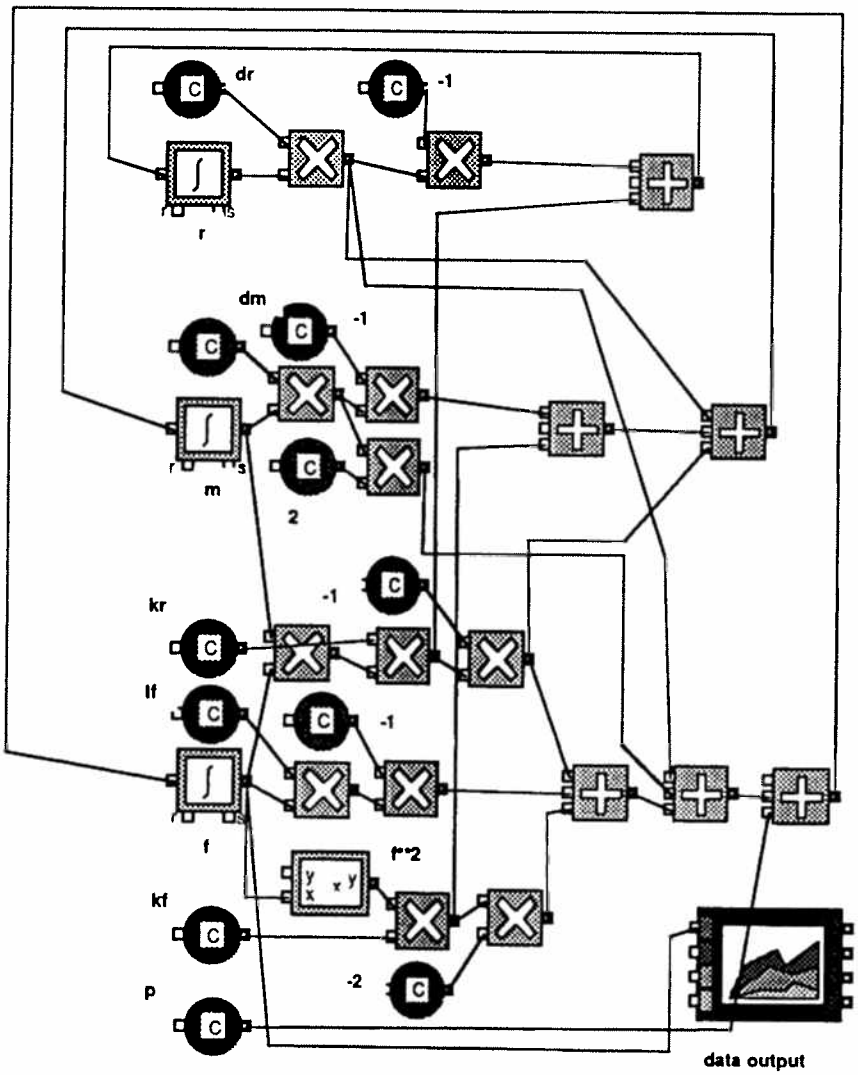


Figure 1

Results

All calculations were done using a Macintosh IIfx.

a) comparison of integration algorithms
The built-in Integrate block of the Generic library supports only two integration methods.

parameter lf= 1000

integration alg.	number of steps time (0,10)	comp.time (min)	numerical
Euler (improved)	10.000	0.5	unstable
Euler (improved)	12.000	1.0	stable
Trapezoidal	20.000	1.45	unstable
Trapezoidal	30.000	2.30	stable

b) variation of parameter lf

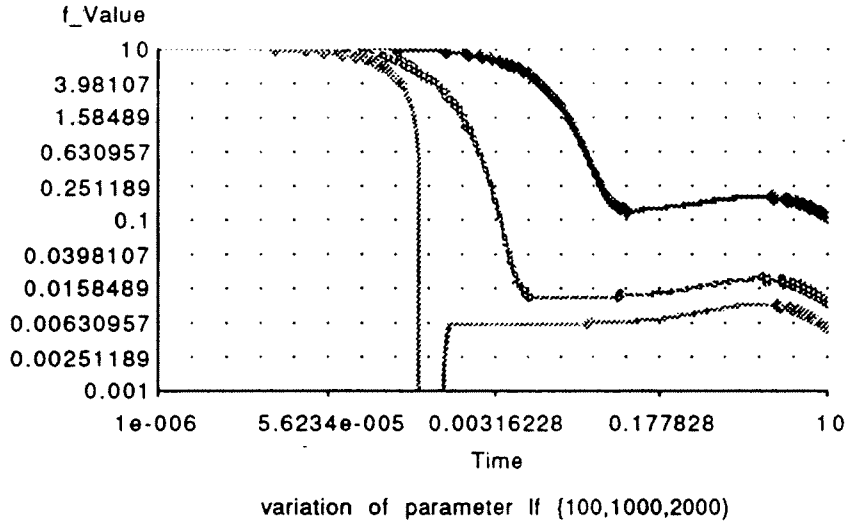


Figure 2

The top curve represents the response for parameter lf=100, with the lower curves showing corresponding results for lf=1000 and 2000 (numerically unstable).

c) calculation of steady states (lf=1000, improved Euler method, number of steps = 10000)
Figure 3 shows the results of the steady state investigation during constant bombardement (lower curve p(t)=1.0E4) and without bombardement (p(t)=0, numerically unstable).

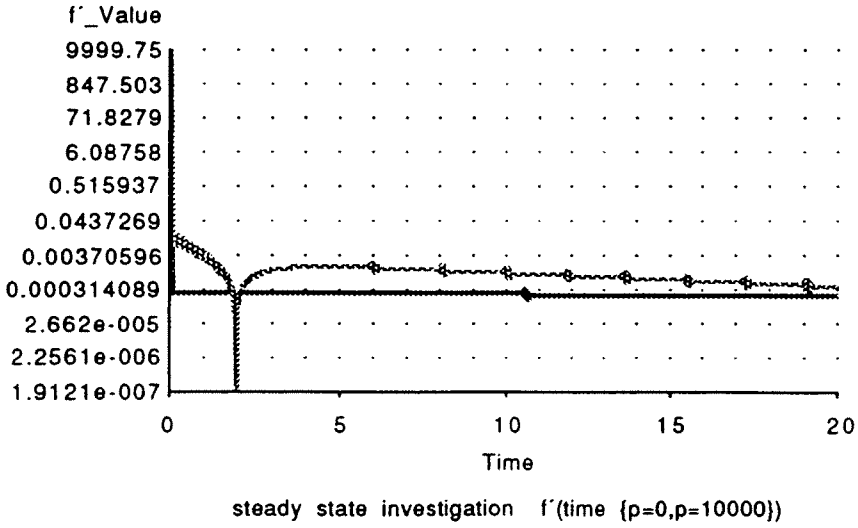


Figure 3

Thorsten Pawletta, Universität Rostock, FB Informatik, Albert-Einstein-Str.21,
D-o-2500 Rostock, Germany;
Tel.: 49- 0381-44424 169; e-mail: pawel@informatik.uni-rostock.de

Comparison 1 - "I Think"

lescription of "I Think"

"I Think" is a special simulation system supporting sys- :m dynamic modeling. You use only a lot of pre-built ntities, such as

converter (constant, built-in item, algebraic equation, graphical function)
stock (various accumulators - reservoir, queue, conveyor, oven)
flow (empties into or drains)
connector (links entities together)

) set up continuous or discrete models. The modeling is ıpported by 55 built-in items. For defining the experimental rocess there are four graph types and identical table types.

raph types:

time series (graph with multiple variables and time on "x" axis)
scatter (a "variable 1" versus "variable 2" plot)
sensitivity (single variable, multiple runs; input parameters "attached")
comparative (multiple runs on the same axis)

The graphical model layout can be used for "ther- ometer" animations. "I Think" allows a fast model con- ruction. The flexibility is limited, because it has not any ot to a modeling or programming language. "I Think" runs ı Macintosh computers and is a trademark of High Perfor- ance Systems Inc.

lodel description

The model is described by items of "I Think" (figure 1) ıd their parametrization (figure 2).

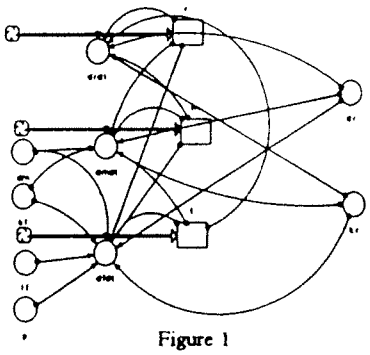


Figure 1

```
) = f(t - dt) + (dfdt) * dt
  INIT f = 9.975
  INFLOWS :
  dfdt = dr*r+2*dm*m-kr*m*f-2*kf*f*f-lf*f+p

t) = m(t - dt) + (dmdt) * dt
  INIT m = 1.674
  INFLOWS:
  dmdt = dr*r-dm*m+kf*f*f-kr*m*f

) = r(t - dt) + (drdt) * dt
  INIT r = 84.99
  INFLOWS:
  drdt = -dr*r+kr*m*f

) = 1
= 0.1
: 0.1
= 1
: 1000
)
```

Figure 2

Results

All calculations were done using a Macintosh IIfx (4 MB RAM, without numeric coprocessor).

a) comparison of integration algorithms: "I Think" supports three integration methods.

parameter lf= 1000, p=0

integration alg.	step width	comp.time (min)	numerical
Euler	1.0E-3	3	unstable
Euler	1.0E-4	7	stable
Runge/Kutta 2	1.0E-3	3.20	unstable
Runge/Kutta 2	1.0E-4	9	stable
Runge/Kutta 4	1.0E-3	4	unstable
Runge/Kutta 4	1.0E-4	12	stable

There are no possibilities to switch off a minimum animation component. That is the reason for the high values of computing time.

b) variation of parameter lf: Runge/Kutta 4; step width=1.0E-4; time interval (0,3)

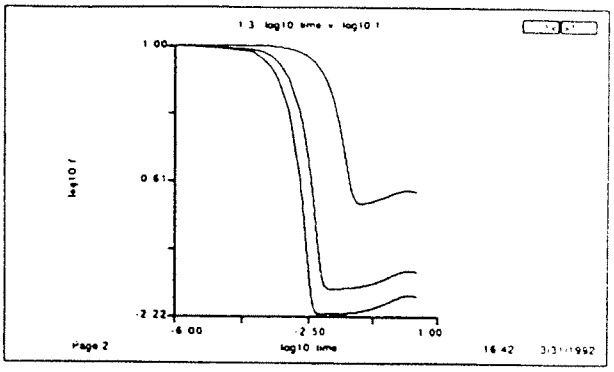


Figure 3

The top curve represents the response for parameter lf=100, with the lower curves showing corresponding results for lf=1000 and lf=2000.

c) calculation of steady states: (lf=1000, Runge/Kutta 4; step width=1.0E-3)

Figure 4 shows the results of the steady state investigation during constant bombardment (curve 2, p(t)=1.0E4) and without bombardment (curve 1, p(t)=0, numerically unstable).

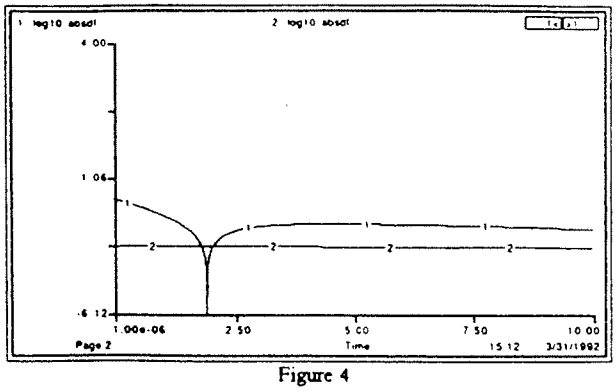


Figure 4

Thorsten Pawletta, Antje Möller, Universität Rostock, FB Informatik, Albert-Einstein-Str. 21, D - O - 2500 Rostock, Germany; Tel.: +49-(0)381-44424 169; e-mail: pawel@informatik.uni-rostock.de

Comparison 1 - "I Think"

Description of "I Think"

"I Think" is a special simulation system supporting system dynamic modeling. You use only a lot of pre-built entities, such as

- converter (constant, builtin item, algebraic equation, graphical function)
- stock (various accumulators - reservoir, queue, conveyor, oven)
- flow (empties into or drains)
- connector (links entities together)

to set up continous or discrete models. The modeling is supported by 55 builtin items.
For defining the experimental process there are four graph types and identical table types.
graph types:

- time series (graph with multiple variables and time on "x" axis)
- scatter (a "variable 1" versus "variable 2" plot)
- sensitivity (single variable, multiple runs; input parameters "attached")
- comparative (multiple runs on the same axis)

The graphical model layout can be used for "thermometer" animations.
"I Think" allows a fast model construction. The flexibility is limited, because it has not any slot to a modeling or programming language.
"I Think" runs on Macintosh computer and is a trademark of High Performance Systems Inc.

Model descripton

The model is described by items of "I Think" (figure 1) and their parametrization (figure 2).

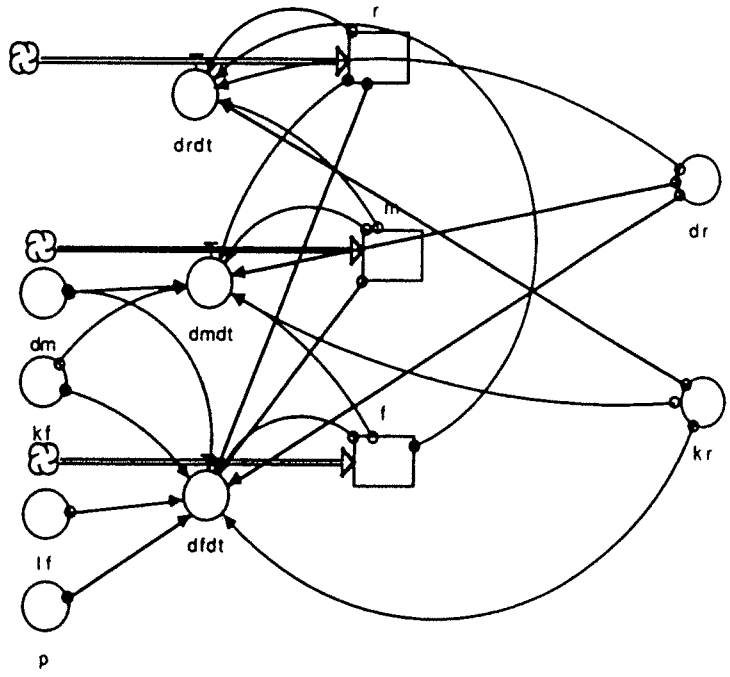


Figure 1

```
f(t) = f(t - dt) + (dfdt) * dt
       INIT f = 9.975
       INFLOWS:
       dfdt = dr*r+2*dm*m-kr*m*f-2
              *kf*f*f-lf*f+p

m(t) = m(t - dt) + (dmdt) * dt
       INIT m = 1.674
       INFLOWS:
       dmdt = dr*r-dm*m+kf*f*f-kr*m*f

r(t) = r(t - dt) + (drdt) * dt
       INIT r = 84.99
       INFLOWS:
       drdt = -dr*r+kr*m*f

dm = 1
dr = 0.1
kf = 0.1
kr = 1
lf = 1000
p = 0
```

Figure 2

Results

All calculations were done using a Macintosh IIfx (4 MB RAM, without numeric coprocessor).

a) comparison of integration algorithms
"I Think" supports three integration methods.

parameter lf= 1000, p=0

integration alg.	step width	comp.time (min)	numerical
Euler	1.0E-3	3	unstable
Euler	1.0E-4	7	stable
Runge/ Kutta 2	1.0E-3	3.20	unstable
Runge/ Kutta 2	1.0E-4	9	stable
Runge/ Kutta 4	1.0E-3	4	unstable
Runge/ Kutta 4	1.0E-4	12	stable

There are no possibility to switch off a minimum animation component. That is the reason for the high values of computing time.

b) variation of parameter lf (Runge/ Kutta 4; step width= 1.0E-4; time interval (0,3))

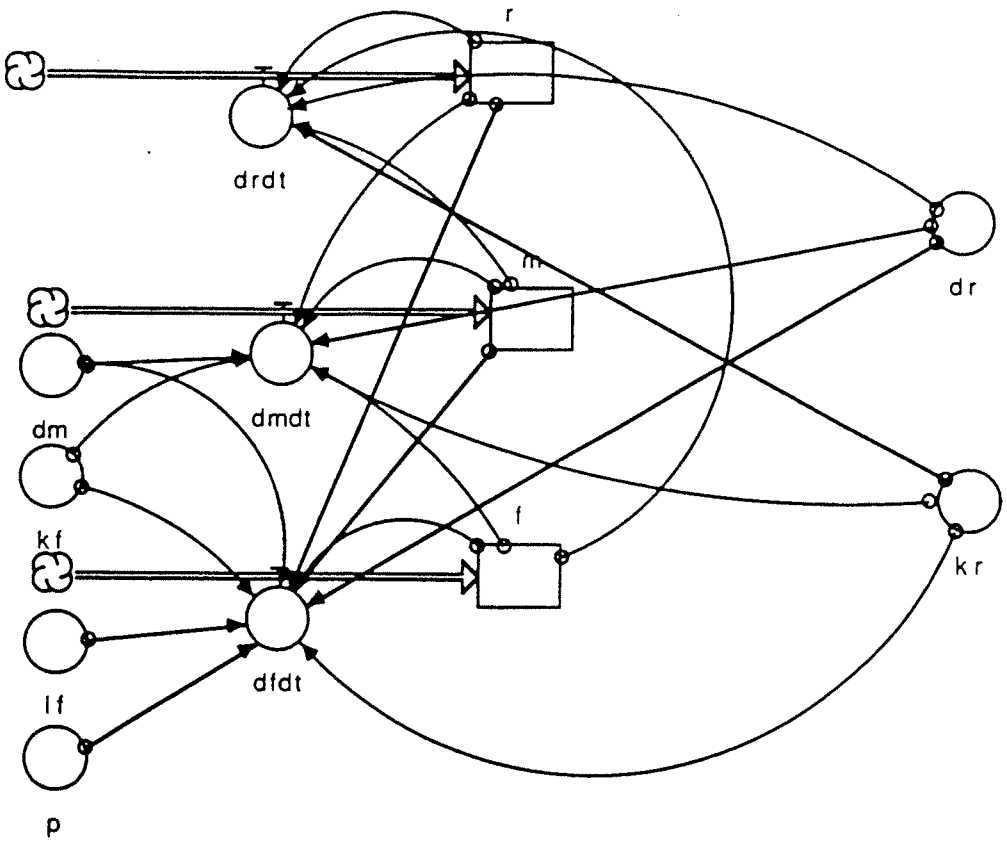
Figure 3

The top curve represents the response for parameter lf=100, with the lower curves showing corresponding results for lf=1000 and lf=2000.

c) calculation of steady states (lf=1000, Runge/ Kutta 4, step width=1.0E-3)
Figure 4 shows the results of the steady state investigation during constant bombardement (curve 2, p(t)=1.0E4) and without bombardement (curve1, p(t)=0, numerically unstable).

Figure 4

Thorsten Pawletta, Antje Möller, Universität Rostock, FB Informatik, Albert-Einstein-Str.21, D-o-2500 Rostock, Germany;
Tel.: 49-0381-44424 169; D-ost 0081-44424 169; e-mail: pawel@informatik.uni-rostock.de


r
drdt
m
dr
dmdt
dm
f
kf
kr
dfdt
lf
p

f(t) = f(t - dt) + (dfdt) * dt
INIT f = 9.975
INFLOWS:
dfdt = dr*r+2*dm*m-kr*m*f-2*kf*f*f-lf*f+p
m(t) = m(t - dt) + (dmdt) * dt
INIT m = 1.674
INFLOWS:
dmdt = dr*r-dm*m+kf*f*f-kr*m*f
r(t) = r(t - dt) + (drdt) * dt
INIT r = 84.99
INFLOWS:
drdt = -dr*r+kr*m*f
dm = 1
dr = 0.1
kf = 0.1
kr = 1
lf = 1000
p = 0

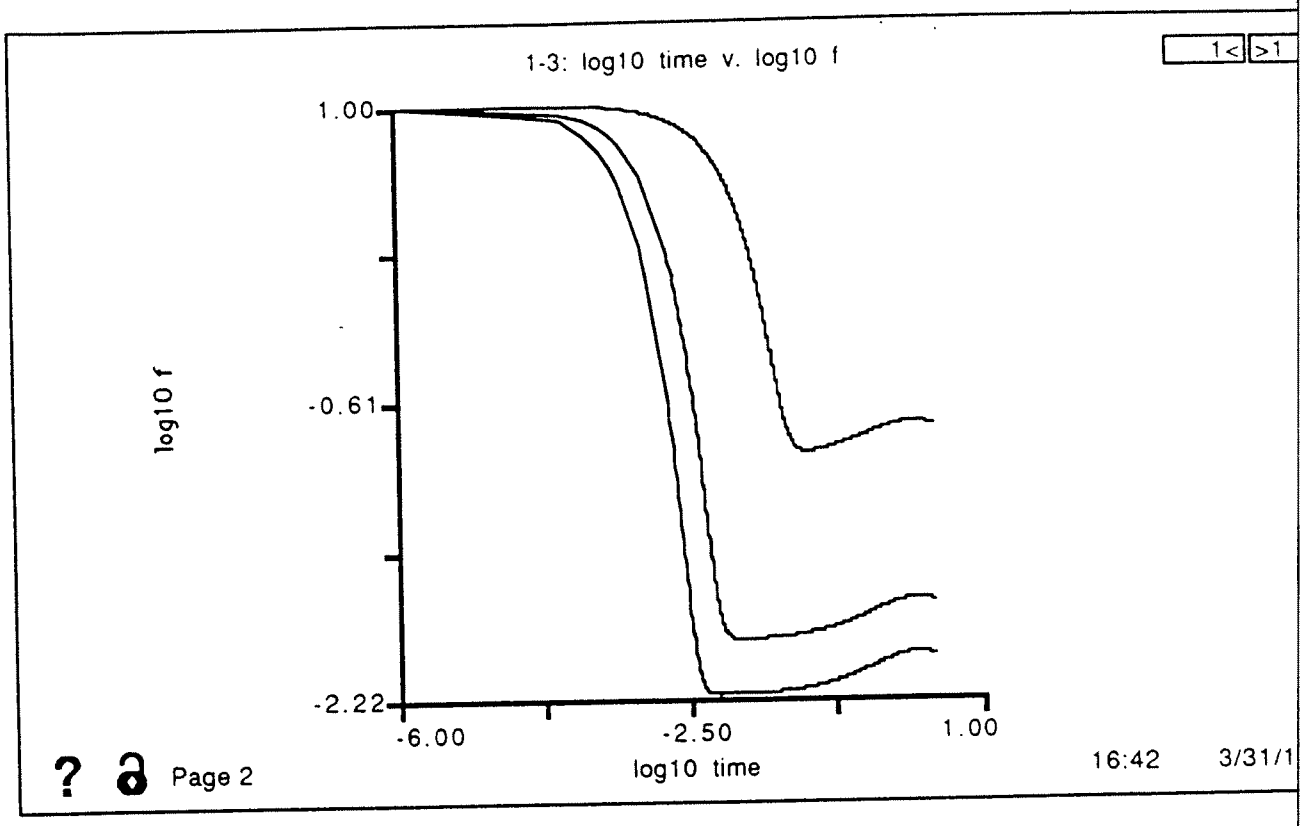
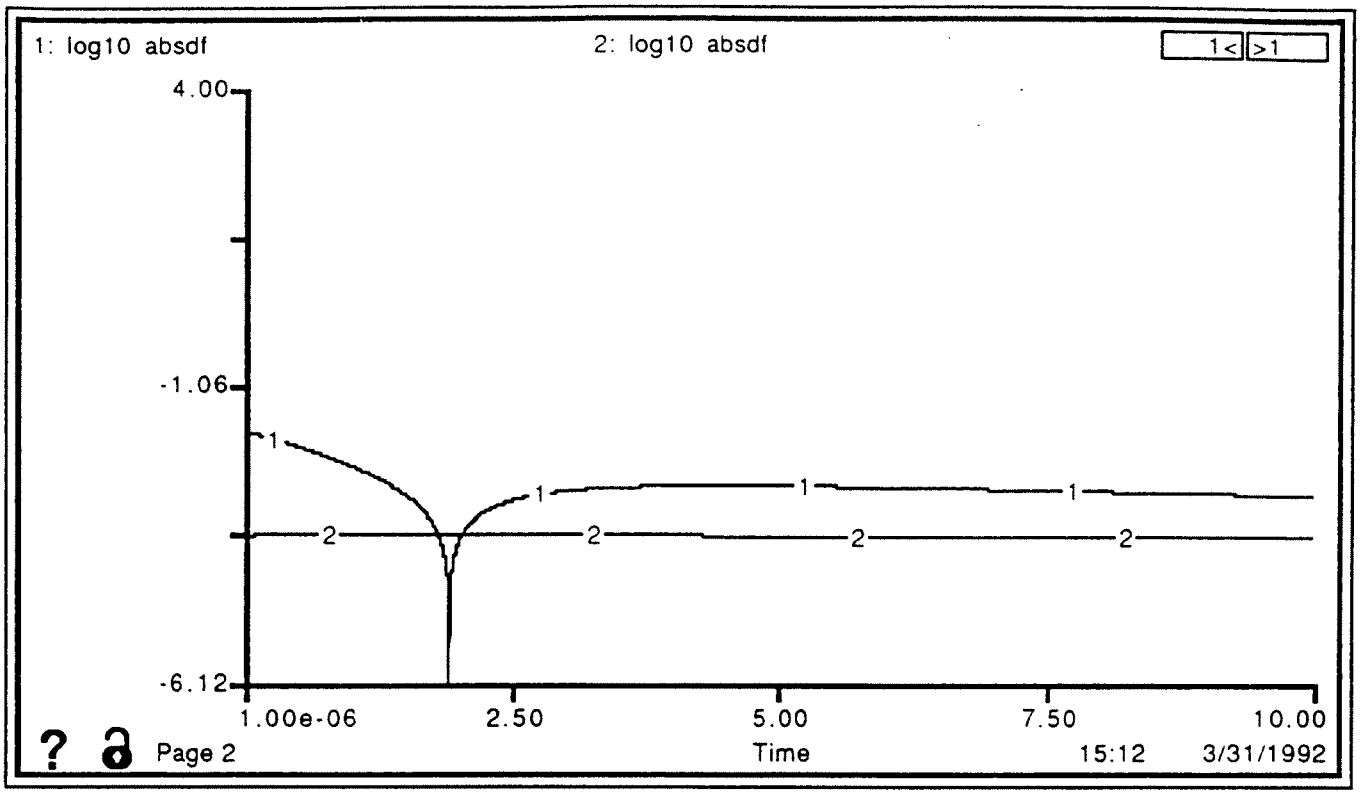

1-3: log10 time v. log10 f
1<
>1
1.00
-0.61
-2.22
log10 f
-6.00
-2.50
1.00
log10 time
Page 2
16:42
3/31/1

Figure 4


1: log10 absdf
2: log10 absdf
1< >1
4.00
-1.06
-6.12
1
2
1.00e-06
2.50
5.00
7.50
10.00
Page 2
Time
15:12
3/31/1992

Comparison 1 - ACSL

ACSL is a general purpose continuous simulation language. It models systems described by time dependent, nonlinear differential equations and/or transfer functions. Linear analysis capabilities (Bode, Nichols, root locus, eigenvalues, for example) are available at runtime.

ACSL runs on personal computers, workstations, mainframe computers, and supercomputers. Programs created on one platform can be transferred to and run on any other platform.

Program: ACSL provides a wide choice of integration algorithms, both fixed and variable. The Gear's stiff algorithm is chosen as the model default in the ALGORITHM statement. The allowable error in the integration calculation is set in the XERROR statement. The model parameters are defined in CONSTANT statements with values as given in the example definition. The rate equations are integrated with the INTEG operator to obtain r, m, and f. Runs are terminated when the logical argument (in this case a time condition) to the operator TERMT becomes true.

We would like the sample points to be exponentially spread in time; *i.e.*, more points to be clustered at smaller times to produce equal separation on a logarithmic scale. Thus, the sample points should be given by:

$$t_o,\ t_o(1+K),\ t_o(1+K)^2,\ \ldots\ t_o(1+K)^n$$

The communication interval (*cint*) is obtained by calculating a Δt of:

$$\Delta t_n = t_o(1+K)^{n+1} - t_o(1+K)^n = t_n K$$

In order to get ten samples per decade, we make:

$$(1+K)^{10} = 10 \quad \text{or} \quad K = 10^{1/10} - 1$$

Since T starts off at zero, we limit the communication to some minimum (and some maximum) value as shown in the last equation in the program.

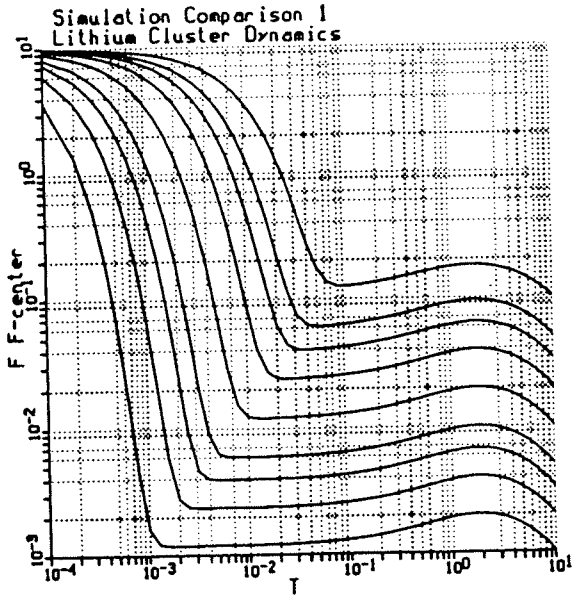
```
PROGRAM simulation comparison 1
      |------------select Gear's stiff integrator by default
      ALGORITHM    ialg = 2
DYNAMIC ; DERIVATIVE
      |------------define initial conditions
      CONSTANT     fz = 9.975      , mz = 1.674
      CONSTANT     rz = 84.99
      |------------define rate coefficients
      CONSTANT     kr = 1.0        , kf = 0.1
      CONSTANT     lf = 1000       , dr = 0.1
      CONSTANT     dm = 1.0        , pc = 0.0
      |------------integrate
      r       = INTEG(-dr*r + kr*m*f, rz)
      m       = INTEG(dr*r - dm*m + kf*f*f - kr*m*f, mz)
      f       = INTEG(dr*r + 2*dm*m - kr*m*f - 2*kf*f*f &
                - lf*f + pc, fz)
      |------------define very small absolute error; first
      | mentioned state establishes the default.
      XERROR r = 1.0e-8
      |------------define stopping condition
      CONSTANT     tstp = 10.0
      TERMT(t .GE. tstp, 'Stopped on time limit')
END ! of DERIVATIVE
      CONSTANT       cintmn = 0.0001, cintmx = 0.2
      |------------log-log plots with equal points/decade
      CONSTANT       pointsperdecade = 10
      cscale  = 10.0**(1.0/pointsperdecade) - 1.0
      cint    = BOUND(cintmn, cintmx, t*cscale)
END ! of DYNAMIC
END ! of PROGRAM
```

Results: A summary of the integration action during th run for all variable step algorithms shows the number c times each state controlled the step size, the number c Jacobian evaluations, and the number of L decompositions during the run. The cpu time required for 10 second run with lf of 1000 is determined by setting th algorithm and running the model interactively at runtime

ALGORITHM	MicroVAX	Sun 4
Adams-Moulton (variable order)	388.85	20.63
Gear's stiff (variable order)	1.99	0.15
Euler (1st order)	8.43	0.47
Runge-Kutta 2nd order	11.48	0.63
Runge-Kutta 4th order	16.70	0.85
Runge-Kutta-Fehlberg 2nd order	13.37	0.84
Runge-Kutta-Fehlberg 5th order	11.01	0.76

Parameter sweep: Next, the integration algorithm is s back to the model default (Gear's stiff) and a parame sweep of lf from 100 to 10000 is executed. The results a plotted on a log-log plot with the command:

```
ACSL> PLOT/XLOG/XLO=0.0001/XHI=tstp &
      f/LOG/TAG='F-center'
```



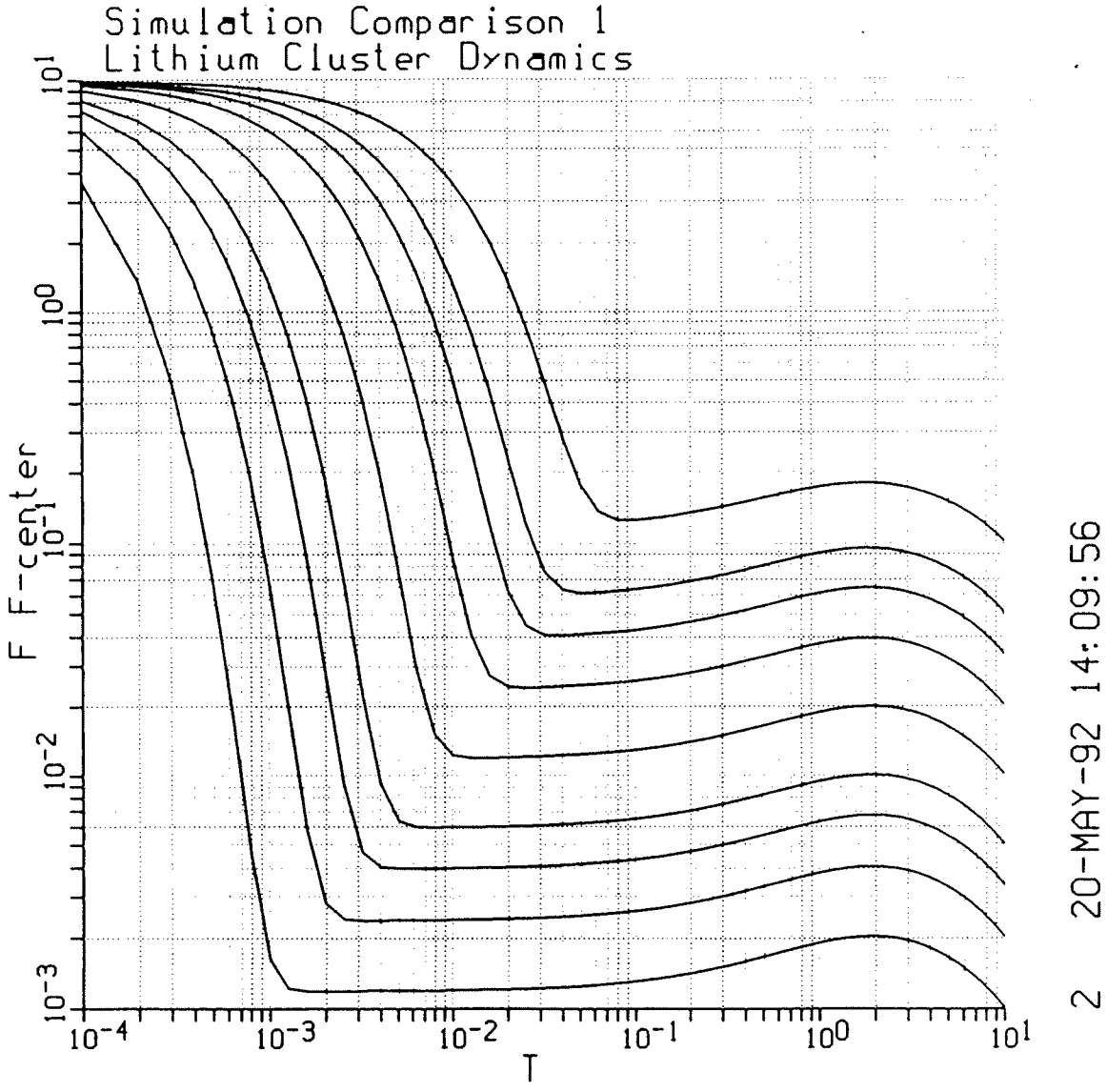
Steady state: Steady state conditions (when derivatives are zero) are evaluated in ACSL with runtime command:

```
ACSL> ANALYZE /TRIM
```

For this model, the steady state at p_c of zero bombardment) and 10000 (constant bombardment) evaluated and the values of r, m, and f are extracted wit DISPLAY command.

p_c	r	m	f
0	-1.7E-7	-1.1D-10	2.5E-12
10000	0.9995	0.1	1.0

Edward E.L. Mitchell and Marilyn B. Kloss, Mitchel Gauthier Associates, 200 Baker Avenue, Concord MA 0174


Simulation Comparison 1
Lithium Cluster Dynamics
F F-center
T
10^{1}
10^{0}
10^{-1}
10^{-2}
10^{-3}
10^{-4}
10^{-3}
10^{-2}
10^{-1}
10^{0}
10^{1}
2 20-MAY-92 14:09:56

Comparison 1 - ACSL

ACSL is a general purpose continuous simulation language. It models systems described by time dependent, nonlinear differential equations and/or transfer functions. Linear analysis capabilities (Bode, Nichols, root locus, eigenvalues, for example) are available at runtime.

ACSL runs on personal computers, workstations, mainframe computers, and supercomputers. Programs created on one platform can be transferred to and run on any other platform.

Program: ACSL provides a wide choice of integration algorithms, both fixed and variable. The Gear's stiff algorithm is chosen as the model default in the ALGORITHM statement. The allowable error in the integration calculation is set in the XERROR statement.The model paramet are defined in CONSTANT statements with values as given in the example definition. The rate equations are integrated with the INTEG operator to obtain r, m, and f. Runs are terminated when the logical argument (in this case a time condition) to the operator TERMT becomes true.

We would like the sample points to be exponentially spread in time; i.e., more points to be clustered at smaller times to produce equal separation on a logarithmic scale. Thus, the sample points should be given by:

<$Et sub o, ~~ t sub o ^ (1~+~K^), ~~ t sub o ^(1~+~K^) sup 2 , ~~...~~ t sub o ^(1~+~K^) sup n>

The communication interval (cint) is obtained by calculating a <$EDELTA t> of:

<$EDELTA t sub n> = <$Et sub o ^(1~+~K^) sup (n+1) ~~~~~ t sub o ^(1~+~K^) sup n> = <$Et sub n ^ K>

In order to get ten samples per decade, we make:

<$E(1 ~~+~~K^) sup 10 ~~~=~~~ 10> or <$EK> = <$E10 sup (1 ^/^ 10) ~~~~~ 1>

Since T starts off at zero, we limit the communication to some minimum (and some maximum) value as shown in the last equation in the program.

```
PROGRAM simulation comparison 1
      !------------select Gear's stiff integrator by default
      ALGORITHM    ialg = 2
DYNAMIC ; DERIVATIVE
      !------------define initial conditions
      CONSTANT     fz = 9.975      , mz = 1.674
      CONSTANT     rz = 84.99
      !------------define rate coefficients
      CONSTANT     kr = 1.0        , kf = 0.1
      CONSTANT     lf = 1000       , dr = 0.1
      CONSTANT     dm = 1.0        , pc = 0.0
      !------------integrate
      r       = INTEG(-dr*r + kr*m*f, rz)
      m       = INTEG(dr*r - dm*m + kf*f*f - kr*m*f, mz)
      f       = INTEG(dr*r + 2*dm*m - kr*m*f - 2*kf*f*f &
                  - lf*f + pc, fz)
      !------------define very small absolute error; first
```

```
      ! mentioned state extablishes the default.
      XERROR r = 1.0e-8
      !------------define stopping condition
      CONSTANT     tstp = 10.0
      TERMT(t .GE. tstp, 'Stopped on time limit')
END ! of DERIVATIVE
      CONSTANT      cintmn = 0.0001, cintmx = 0.2
      !-------------log-log plots with equal points/decade
      CONSTANT      pointsperdecade = 10
      cscale  = 10.0**(1.0/pointsperdecade) - 1.0
      cint    = BOUND(cintmn, cintmx, t*cscale)
END ! of DYNAMIC
END ! of PROGRAM
```

Results: A summary of the integration action during the run for all variable step algorithms shows the number of times each state controlled the step size, the number of Jacobian evaluations, and the number of LU decompositions during the run. The cpu time required for a 10 second run with lf of 1000 is determined by setting the algorithm and running the model interactively at runtime.

ALGORITHM	MicroVAX	Sun 4	
Adams-Moulton (variable order)		388.85	20.63
Gear's stiff (variable order)		1.99	0.15
Euler (1st order)	8.43	0.47	
Runge-Kutta 2nd order	11.48	0.63	
Runge-Kutta 4th order	16.70	0.85	
Runge-Kutta-Fehlberg 2nd order		13.37	0.84
Runge-Kutta-Fehlberg 5th order		11.01	0.76

Parameter sweep: Next, the integration algorithm is set back to the model default (Gear's stiff) and a parameter sweep of lf from 100 to 10000 is executed. The results are plotted on a log-log plot with the command:

```
ACSL> PLOT/XLOG/XLO=0.0001/XHI=tstp &
        f/LOG/TAG='F-center'
```

Steady state: Steady state conditions (when the derivatives are zero) are evaluated in ACSL with the runtime command:

```
ACSL> ANALYZE /TRIM
```

For this model, the steady state at pc of zero (no bombardment) and 10000 (constant bombardment) are evaluated and the values of r, m, and f are extracted with the DISPLAY command.

pc	r	m	f
0	-1.7E-7	-1.1D-10	2.5E-12
10000	0.9995	0.1	1.0

Edward E.L. Mitchell and Marilyn B. Kloss, Mitchell and Gauthier Associates, 200 Baker Avenue, Concord MA 01742 USA

To Compare: 1st ACSL Solution

ACSL-Model:

```
PROGRAM EUROSIM EXAMPLE No. 1
' Language ACSL Level 9, Mitchell & Gauthier Ass., U.S.A.'
' prepared by Dr. Ingrid Bausch-Gall, January 2nd, 1991 '
CONSTANT kr=1., kf=0.1, lf=1000., dr=0.1, dm=1., p=0.
CONSTANT fnull=9.975, mnull=1.674, rnull=84.99  $ 'initial conditions'
ALGORITHM IALG=2                $  'take Gears stiff for integration '
CINTERVAL CINT=0.05             $  'store results at multiples of CINT'
CONSTANT  TEND=10.              $  'simulation time'
' --------  model equations --------  '
  r = integ(-dr*r + kr*m*f,rnull)
  m = integ(dr*r - dm*m + kf*f*f -kr*m*f,mnull)
  f = integ(dr*r + 2.*dm*m-kr*m*f-2.*kf*f*f-lf*f+p,fnull)
  TERMT(T.gt.TEND)              $  'stop at simulation time'
END
```

ACSL-Runtime-Commands:

```
     s p=1.e4, wesitg=.f., nstp=1
' a) Comparision of computer time '
     prepar t,r,m,f             $ 'store results of these variables'
     s ialg=1                   $ 'calculate with ADAMS-Moulton method'
     spare $ start $ spare      $ 'give computer time'
     s ialg=2                   $ 'choose now Gear's stiff'
     spare $ start $ spare
     s ialg=9                   $ 'one step Runge-Kutta order 4/5'
     spare $ start $ spare
' b) Parameterstudies '
     s ialg=2                   $ 'choose Gears Stiff for parameterstudies'
     s lf=1.e2
     start
     s nrwitg=.t.               $ 'write all results on one file'
     s lf=1.e3
     start
     s lf=1.e4
     start
     s title='Example EUROSIM 1, Parameterstudies '
     s title(11)='lf = 1.e2 (1), 1.e3 (2), 1.e4 (3)'
     s ftsplt=.t.,symcpl=.t.,npccpl=40
     plot f,'xhi'=10.,'char'='1'   $  plot results
' c) Calculate steady state result '
     s p=1.e4
     analyz 'list'=.t.,'trim'
     s p=0.
     analyz 'trim'
     stop
```

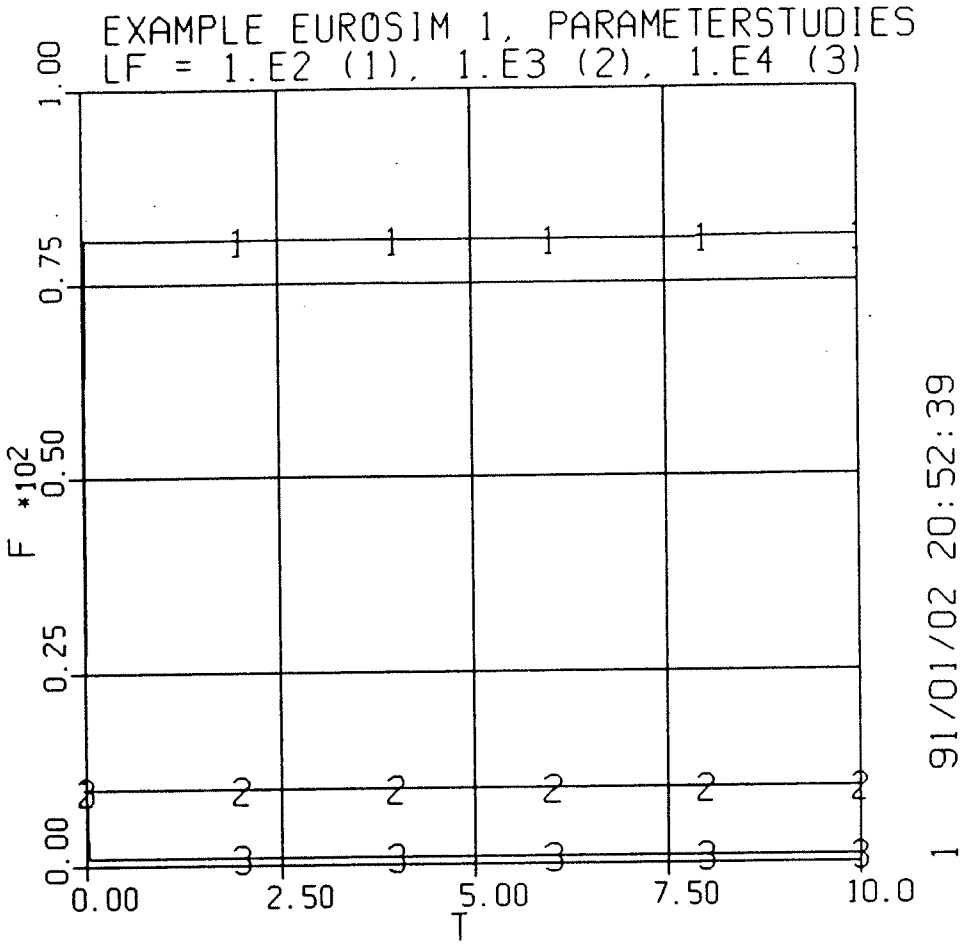
Results:

All calculations have been done on a Commodore PC-40 (AT) with 12 MHz and a 80286 numeric coprocessor.

a) Comparision of computer time

Adams-Moulton-Predictor-Corrector Method, IALG=1	155.055 sec.
Gear's Stiff, IALG=2	3.460 sec.
Runge-Kutta order 4/5 with stepsize control, IALG=9	55.035 sec.

b) Plot of Parameterstudies



c) Calculate steady state result for lf=1000.

p = 1.E4 gives as last iteration:

```
Newton step  0.24366500 Steep desc step  0.11443300 mu 0
  State vector - iteration number 11
        F 10.0000000          M 10.0000000          R 1000.00000
  Derivative vector - residual is  5.3226E-05  previous  0.02483470
                Scaled residual is  9.9485E-05  previous  0.04599450
    Z09996 5.1546E-05      Z09997 5.4854E-05      Z09998-5.3751E-05
```

p = 0. gives as last iteration:

```
Newton step  0.12913000 Steep desc step  0.06764160 mu 0
  State vector - iteration number 8
        F-1.5045E-12          M-1.5373E-09          R 1.3290E-07
  Derivative vector - residual is  1.3339E-08  previous  0.01348860
                Scaled residual is  2.5906E-08  previous  0.02502220
    Z09996 1.1720E-08      Z09997 1.4827E-08      Z09998-1.3290E-08
```

Comparison 1 - STEM

Short description of STEM

STEM, Simulation Tool for Easy Modelling, is a general purpose simulation package for MS-DOS machines. Models have to be specified in a Model Specification File, containing the model equations. This Model Specification file is translated by STEM to a Turbo Pascal program and compiled with Borland's Turbo Pascal compiler. The resulting executable file is a menu-driven interactive program with facilities for simulation, calibration, printing, graphical and numerical presentation of results. It is possible to run a model under batch-file control. External data (ASCII or Lotus 1-2-3) can be used in the simulation. For calibration of model parameters a target function must be specified, for instance the difference between simulated data and external data. A large set of standard functions is avaliable, if this should not be enough one can add self-programmed Turbo Pascal functions.

Model description

In a STEM model variables are divided in groups, each with their own properties. In this model you can find constants c[ ], states s[ ] with derivatives d[ ] and auxiliaries a[ ]. Running a model, each group is presented in a window on the screen. Comments can be displayed running the model. Graphical windows can be defined also.

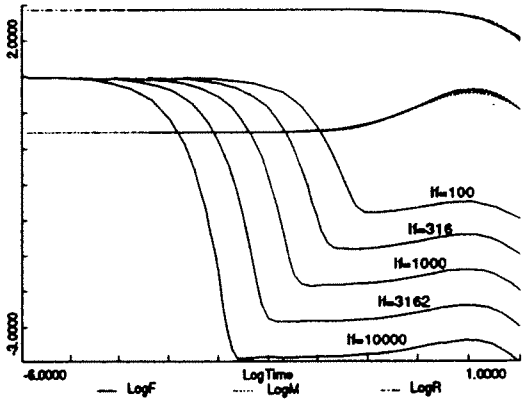
```
Environment
  BegValue    = 0      (* initial value of independent variable *)
  EndValue    = 10     (* end value of independent variable *)

Declaration
  Measurement    (* no external data *)
  Constants      (* constants used in program *)
    c[R0]   = 84.99  ! starting value for s[R]
    c[M0]   = 1.674  ! starting value for s[M]
    c[F0]   = 9.975  ! starting value for s[F]
    c[Dr]   = .1   ! rate for decay of R-center into M-center and F-center
    c[Dm]   = 1      ! rate for decay of M-center into two F-centers
    c[Lf]   = 1000   ! loss of F-centers at surface
    c[Kr]   = 1 ! formation rate of R-center out of M-center and F-center
    c[Kf]   = .1     ! rate for formation of M-center out of two F-centers
    c[P]    = 0      ! electron bombardment

  Zerostate     (* initial conditions *)
    s[Time] = BegValue ! independent variable
    s[R]    = c[R0]  ! concentration of aggragates with three F-centers
    s[M]    = c[M0]  ! concentration of aggragates with two F-centers
    s[F]    = c[F0]  ! concentration of F-centers

Model         (* the model-equations *)
  a[dRdT]   = c[Kr]*s[M]*s[F] - c[Dr]*s[R]  ! net formation of R
  a[dMdT]   = c[Kf]*sqr(s[F]) - c[Dm]*s[M] ! net formation of M from F
  d[R]      = a[dRdT]
  d[M]      = a[dMdT] - a[dRdT]
  d[F]      = c[P] - a[dRdT] - 2*a[dMdT] - c[Lf]*s[F]

Output        (* output-variables *)
  a[LogTime] = Conditional(s[Time]>0,log10(s[Time]),-MaxFloat)
  a[LogR]   = Conditional(s[R]>0,log10(s[R]),-MaxFloat)
  a[LogM]   = Conditional(s[M]>0,log10(s[M]),-MaxFloat)
  a[LogF]   = Conditional(s[F]>0,log10(s[F]),-MaxFloat)

Minimization  (* no calibration-criteria *)
UserDefined   (* no userdefined functions *)
```

Results

a) Comparison of integration algorithms. The system was simulated over a period of 10 seconds using nine different integration algorithms available in STEM. Computation times for a 20 MHz 80386 system with 387 coprocessor are presented in the table below. Simulation is carried out with an absolute error of 0.001 and a relative error of 1E-6. All integration methods use variable step size, Gear and Adams also variable order. Writing of results to screen and disk is minimized. Times are calculated using a Pascal function in the Userdefined block (not presented above).

algorithm	computation time (seconds)
Gear's stiff, variable order	0.50
Adams-Bashforth-Moulton, variable order	41.03
Runge-Kutta-Fehlberg, order 1(2)	18.84
Runge-Kutta-Fehlberg, order 2(3)	11.54
Runge-Kutta-Fehlberg, order 3(4)	10.27
Runge-Kutta-Fehlberg, order 4(5)	10.82
Dormand-Prince, order 5(4)	13.45
Runge-Kutta-Fehlberg, order 5(6)	13.30
Runge-Kutta-Fehlberg, order 7(8)	20.98

b) Parameter sweep. This task, changing constant c[Lf] may be performed manually running the model, or in a STEM-batch file. STEM produces the following figure varying Lf from 100 to 10000. The (logarithmic) values of F, M and R-centres are displayed against (log) Time.



c) Steady state calculation. STEM can solve the states for all derivatives equal to zero. With lf = 1000, the results are:

p	R	M	F
10000	1000	10	10
0	0	0	0

More information about STEM and a demonstration disk with this model is available with:

Diederik Waardenburg, ReMeDy Systems Modelling P.O.Box 11019, 7502 LA Enschede, The Netherlands E-Mail: REMEDY@UTWENTE.NL.

Comparision 1 - TUTSIM

Description of TUTSIM

TUTSIM is a blockoriented simulation system with some equation oriented aspects. It supports a wide range of analog and discrete blocks for system modelling and control. In addition there are blocks for Bondgraph models in this simulation system. Some Studies in the frequency domain may be made by the TUTFFT task. TUTSIM was developed at the Twente University of Technology in The Netherlands and is now supported and distributed by:

Meerman Automation, Postbus 15, 7160 AC Neede,
The Netherlands, Tel. (0031)5450-93901

and for North America and Canada:

TUTSIM Products, 200 California Avenue # 212,
Palo Alto, CA 94306, USA

TUTSIM runs on IBM-PC/XT/AT and PS/2 compatibles. The mathematic coprocessor 80x87 is supported, but not necessary. Suppoted graphic bords are Hercules, IBM CGA, IBM EGA, IBM VGA and SVGA.

Model description

The model was set up by TUTSIM's own interactive editor TUTEDIT, which automatically starts at each simulation session, except you have a predefined model on disk. All defined symbols (left hand side of the equations below) may be accessed by TUTCALC, the simulation part of TUTSIM, which follows after TUTEDIT.

```
F=PLOT[f]
      PLOT number          : 1.00000
      Minimum              : 0.000000
      Maximum              : 2.50000E-2
dmrdt=1[(1.00000E-1*f*f)-m]              ;dm/dt + dr/dt
drdt=1[(m*f)-(1.00000E-1*r)]             ;dr/dt
f=INT[-drdt-(2.00000*dmrdt)-             ;f(t)
      (1.00000E+3*f)]
      Initial value        : 9.97500
m=INT[dmrdt-drdt]                        ;m(t)
      Initial value        : 1.67400
r=INT[drdt]                              ;r(t)
      Initial value        : 8.49900E+1
t=TIME[]
      Time step DELTA      : 5.00000E-4
      End time             : 1.00000E+1
```

Results

All simulation runs were made on an 16 MHz 386-SX-AT with a Cyrix-Coprocessor, which is compatible to the Intel 80387-SX.

a) Computing time depending on the two different integration algorithms available on TUTSIM

TUTSIM has two different integration algoritms with fixed stepsize:

- Adams-Bashfort second order (INT)
- Euler (EUL)

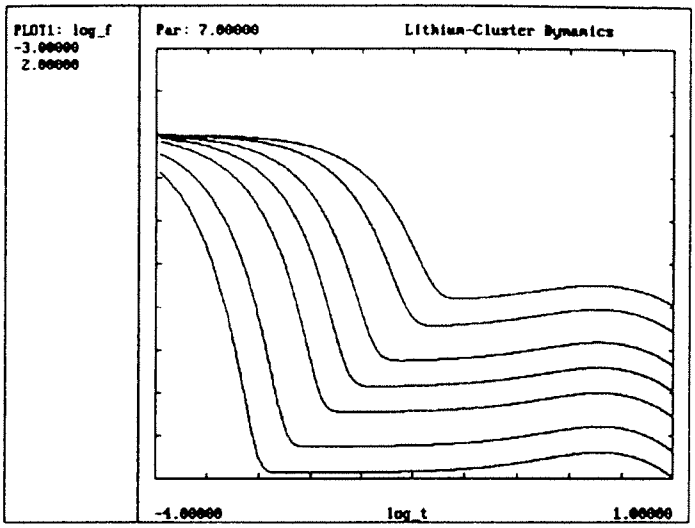
The algoritmus is selected within TUTEDIT by selecting the block for the integration (INT or EUL).
The simulation time was measured with linear spacing of t-axis und f(t)-axis. During simulation run a VGA-plot was drawed with 500 simulation points at the screen.

algorithmus	maximum-step-size	simulation time
INT	5E-4	49 sec.
EUL	5E-4	44 sec.

b) Paramater variation of l_f

For the parameter variation l_f was defined as a function table [100, 200, 500, 1000, 2000, 5000, 10000] with a variable as input. TUTCALC can vary the parameter value via this input during an automatic multirun.

To get a logarithmic spaced plot, two LOG-blocks were added.



c) Steady states

For calculation of steady states, the derivations of the differential equations have to be set to zero. The result are 3 algebraic equations whith the state variables at the left hand side:

$$r = k_r mf/d_r$$
$$m = (d_r r + k_f f^2 - k_r mf)/d_m$$
$$f = (d_r r + 2d_m m - k_r mf - 2k_f f^2 + p)\,l_f$$

To avoid algebraic loops, m and f are defined by ADL (Algebraic delay) blocks. The table below shows the results for p=0 during 5 iteration steps:

n	m	r	f
0.000000	1.67400	1.66982E+2	9.97500
1.00000	9.95006	-1.64695	-1.65521E-2
2.00000	2.73973E-5	5.45208E-6	1.99001E-2
3.00000	3.96013E-5	-9.66588E-12	-2.44080E-8
4.00000	5.95750E-17	4.71849E-23	7.92026E-8
5.00000	6.27305E-16	-7.12279E-33	-1.13546E-18

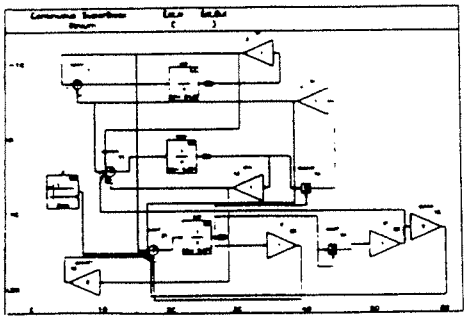
and for p=10000:

n	m	r	f
0.000000	1.67400	1.66982E+2	9.97500
1.00000	9.95006	9.93359E+2	9.98345
2.00000	9.96692	9.96689E+2	9.99997
3.00000	9.99993	9.99987E+2	9.99993
4.00000	9.99987	9.99987E+2	1.00000E+1
5.00000	1.00000E+1	1.00000E+3	1.00000E+1

Bernd Lange, Fachhochschule Ulm, Fachbereich Automatisierungstechnik, Parkstraße 4, D-W-7340 Geislingen, Tel. +49-(0)7331 22526, Fax +49-(0)7331 40898

Comparison 1 - MATRIXx

MATRIXx is a comprehensive linear system analysis tool. It is an interactive matrix manipulation environment which combines powerful numerical tools of LINPACK and EISPACK with an easy to use interface, comprehensive graphics facility and an expandable function library. In MATRIXx nonlinear sytems have to be described by block diagrams, Fig. 1. Leaving the graphical model editor (System Build) by the command analyze, the simulation is carried out in the MATRIXx core. To compare the complete capabilities of the different integration algorithms, the simulations have been carried out for two time 'vectors': with 77 non equidistant points and with 10 000 equidistantly spaced points of 1 msec.



For the non equidistant time vector the command sequence is

```
sim('ialg'); 6
v = [1.2,1.5,2,3,4,5,6,7,8,9,10];
t = [1e-6 1e-5 1e-4 1e-3 1e-2 0.1 1];
t = t*v
clock('cpu'); yss=sim(t); time =clock('cpu');
```

For equidistantly spaced points row number 3,4 and 5 are replaced by `t = [0.001:0.001:10]';`

Results: PC 486, 33 Mz

Integration algorithm	10 000 equidistant time points	77 not equidistant time points
Implicit Stiff System Solver	117.0 sec	3.02 sec
Variable Kutta-Merson	261.0 sec	71.0 sec
Fixed Kutta-Merson	255.7 sec	
4th order Runge Kutta	217.7 sec	
RK2 (Modified Euler)	132.6 sec	
Euler	90.3 sec	

Results: Workstation Sun 4, 40 MHz

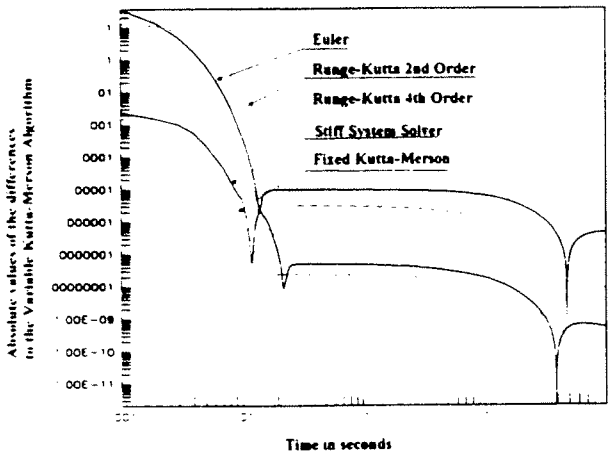
Quicksim Solver	8.2 sec	failed
Variable Adams-Moulton	11.62 sec	1.78 sec
Stiff System Solver	15.21 sec	0.43 sec
Variable Kutta-Merson	24.83 sec	6.65 sec
Fixed Kutta-Merson	23.31 sec	
4th order Runge Kutta	19.02 sec	
RK2 (Modified Euler)	11.81 sec	
Euler	8.19 sec	

For the Parameter simulation the 'Stiff System Solver' was used and the command sequence is: (compiled in a so called Execute File):

```
kr = 1; kf=0.1,1f = 1000; dr = 0 1, dem = 1; p = 0;
v = [1.2,1.5,2,3,4,5,6,7,8,9,10];
t1 = [1e-6,1e-5,1e-4,1e-3,1e-2,0 1,1];
t = (t1.*v)';
lfp = [100,200,500,800,1000,2000,5000,8000,1e4];
y3 = 0*t;
clock('cpu');
for i=1:9;...
lf = lfp(i);...
y = sim(t);...
y3=[y3,y(:,3)];...
end;
plot(t,y3(:,2:20),'logx,logy');...
time=clock('cpu')
```

PC-Simulation: 29.0 sec Workstation Simulation: 3.56 sec

To compare the results of the different integration algorithms the Variable Kutta-Merson algorithm is considered as a reference (deviations see figure).



For the calculation of the steady state the trim command causes a linearization of the system under consideration with all the known problems. An iteration of the procedure can improve the result. Two iteration steps have been carried out. The command for the calculation of the steady state is

```
[xt,ut,yt] = trim(0,1,[0,0,0],[0,0,0],x0)
```

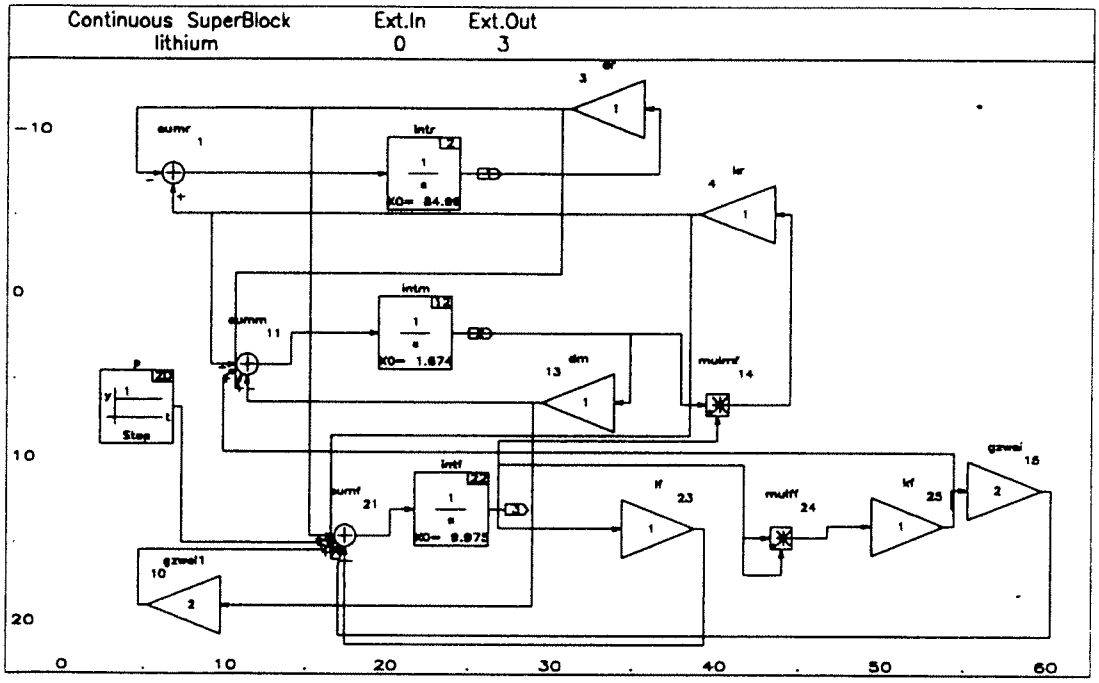
The trimmed variables are: state vector xt, the input ut, and the output yt. The first parameter of the trim command is the input value u and the second indicates that this value should be frozen. The next two vectors concern the nominal output vector where the second means that the output is not frozen. x0 indicates the initial condition. In the second iteration step x0 is replaced by xt from the foregoing step. The result is shown in the following table:

p	r	m	f
0	-3.4e-7	-1.1e-9	-3.0e-11
10 000	1002.6	10	10

Rudolf H. Kern, Fachbereich Feinwerktechnik Fachhochschule Heilbronn, Max-Planck-Str. 39, D-74081 Heilbronn

MATRIXx is a comprehensive linear system analysis tool. It is an interactive matrix manipulation environment which combines powerful numerical tools of LINPACK and EISPACK with an easy to use interface, comprehensive graphics facility and an expandable function library. MATRIXx includes comprehensive tools for system analysis and control design (system idenitifcation, opitmization, signal processing, robust control) as well as nonlinear simulation, block diagram system modeling, and in the workstation version also automatic real-time code generation and implementation.

In MATRIXx nonlinear sytems have to be described by block diagrams, Fig. 1. Leaving the graphical model editor (System Build) by the command analyze, the simulation is carried out in the MATRXx core. To compare the complete capabilties of the different integration algorithms, the simulations have been carried out for two time 'vectors': the first with 77 non equidistnat points and the second with 10 000 equidistant spaced points of 1 msec.



For the non equidistant time vector the command sequence is

```
sim('ialg')                                  // menu for selecting integration allgorithm
6                                            // number of integration allgorithm
v = [1.2,1.5,2,3,4,5,6,7,8,9,10];            // points within a decade
t = [1e-6 1e-5 1e-4 1e-3 1e-2 0.1 1];        // decades for time vector
t = t*v                                      // generating the time vector t

clock('cpu'); yss=sim(t); time =clock('cpu'); // start clock, simulate, stop clock and read out
```

For equidistant spaced points row number 3,4and 5 is replaced by

```
t = [0.001:0.001:10]';                       // generating the time vector
```

Results: PC 486, 33 Mz

Integration algorithm	10 000 equidistant time points	77 not equidistant time points
Implicit Stiff System Solver	117.0 sec	3.02 sec
Variable Kutta-Merson	261.0 sec	71.0 sec
Fixed Kutta-Merson	255.7 sec	
4th Order Runge Kutta	217.7 sec	
RK2 (Modified Euler)	132-6 sec	
Euler	90.3 sec	

Results: Workstation Sun 4, 40 MHz

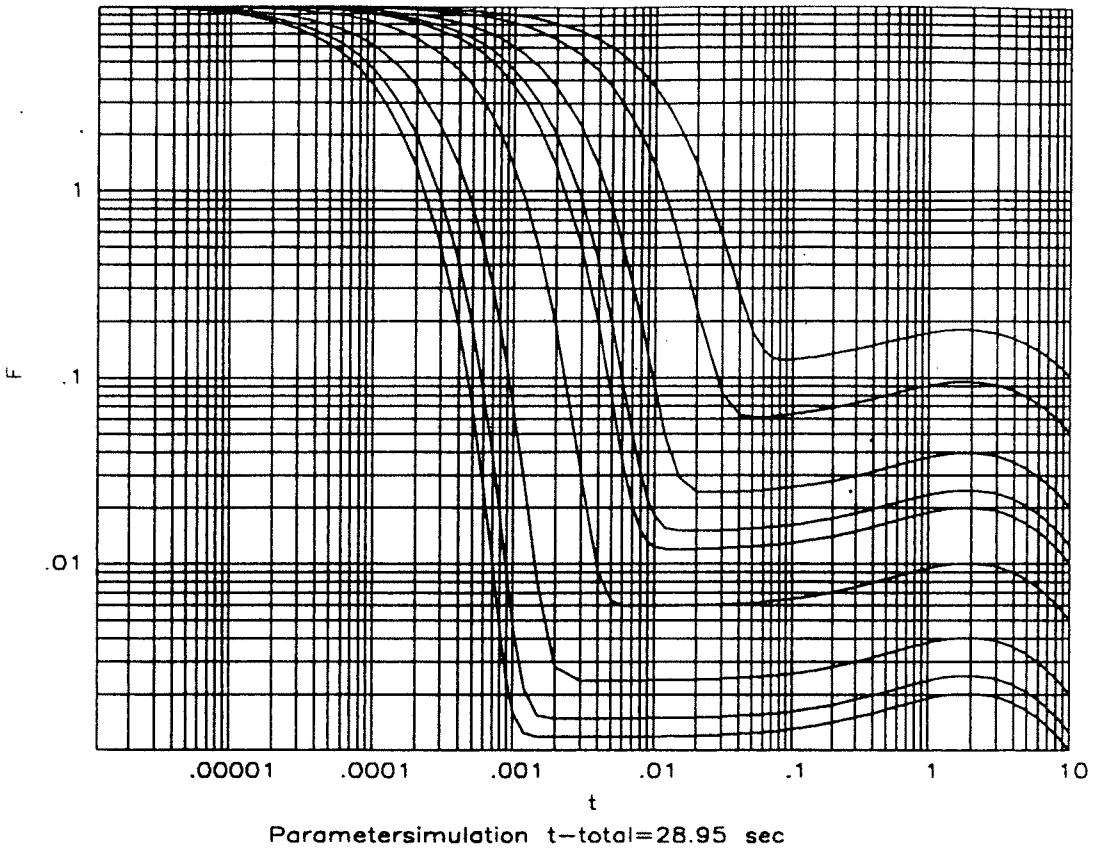
Intgration algorithm	10000 equidistant time points	77 not equidistant time points
Quicksim Solver	8.2 sec	failed
Variable Adams-Moulton	11.62 sec	1.78 sec
Stiff System Solver	15.21 sec	0.43 sec
Variable Kutta-Merson	24.83 sec	6.65 sec
Fixed Kutta-Merson	23.31 sec	
4th Order Runge-Kutta	19.02 sec	
RK2 (Modified Euler)	11.81 sec	
Euler	8.19 sec	

For the Parameter simulation the 'Stiff System Solver' was used and the command sequence is: (compiled in a so called Execute File):

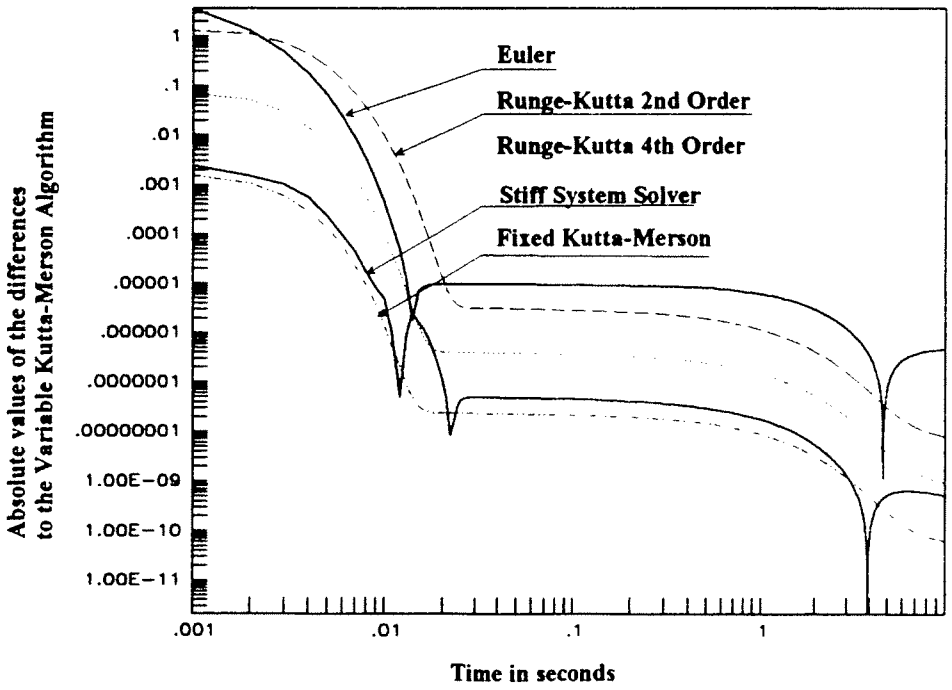
```
kr = 1; kf=0.1;lf = 1000; dr = 0.1; dem = 1; p = 0;   // system parameter
v = [1.2,1.5,2,3,4,5,6,7,8,9,10];                     //timepoints within a decade
t1 = [1e-6,1e-5,1e-4,1e-3,1e-2,0.1,1];               //decades
t = (t1.*.v)';                                        //timepoints
lfp = [100,200,500,800,1000,2000,5000,8000,1e4];      // simulation parameter
y3 = 0*t;                                             //
clock('cpu');                                         // clock start
for i=1:9;...                                         // loop start
lf = lfp(i);...                                       // current parameter
y = sim(t);...                                        // simulation for current parameter
y3=[y3,y(:,3)];...                                    // storing of the results in a matrix
end;                                                  // loop end
plot(t,y3(:,2:20),'logx,logy');...                    // display the results
time=clock('cpu')                                     // stop clock, read simulation time
```

PC-Simulation: 29.0 sec Workstation Simulation: 3.56 sec

The result is shown in the follwoing Fig.



To compare the results of the different integration algorithms the Variable Kutta-Merson algorithm is considered as a reference. The deviations hereof are plotted in the next Fig.



For the calculation of the steady state the trim command causes a linearization of the system under consideration with all the known problems. An iteration of the procedure can improve the result. Two iteration steps have been carried out. The command for the calculation of the steady state is

[xt,ut,yt] = trim(0,1,[0,0,0],[0,0,0],x0)

The trimmed variables are: state vector xt, the input ut, and the output yt. The first parameter of the trim command is the input value u and the second indicates that this value should be frozen. The next two vectors concern the nominal output vector where the second means that the output is not frozsen. x0 indicates the initial condition. In the second iteration step x0 is replaced by xt from the foregoing step. The result is shown in the following table:

p	r	m	f
0	-3.4e-7	-1.1e-9	-3.0e-11
10 000	1002.6	10	10

Rudolf H. Kern
Fachbereich Feinwerktechnik
Fachhochschule Heilbronn
Max-Planck-Str. 39
D-74081 Heilbronn

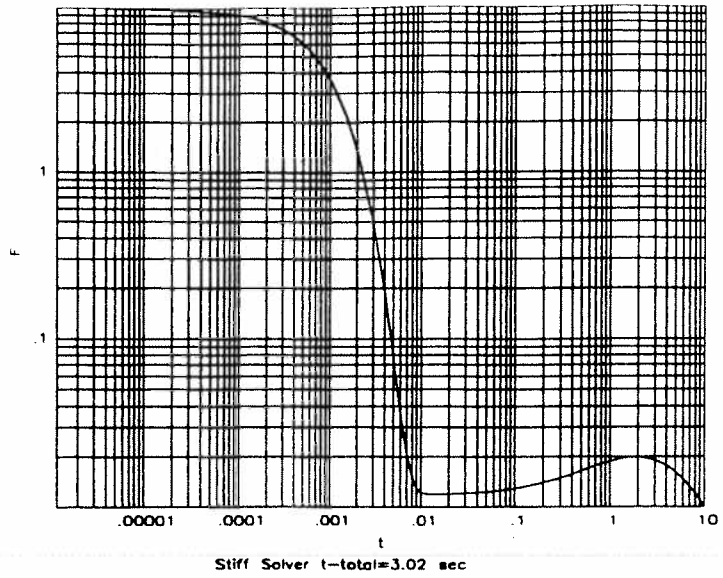


Bild 2: Simulationsergebnis mit 'Stiff Solver' (PC-Version)

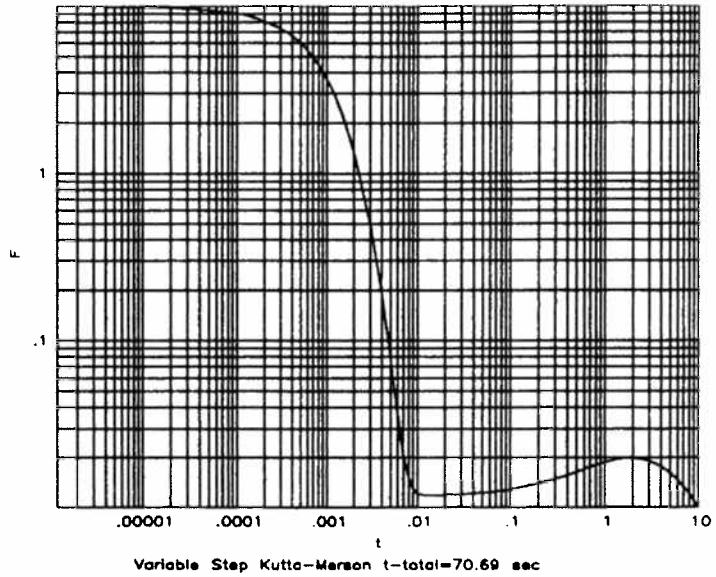


Bild 3: Simulationsergebnis mit Kutta-Merson Verfahren variabler Schrittweite

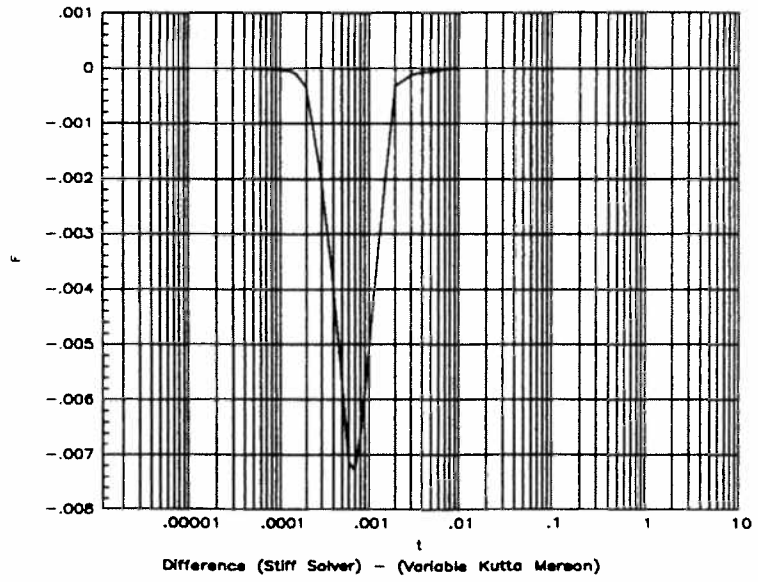
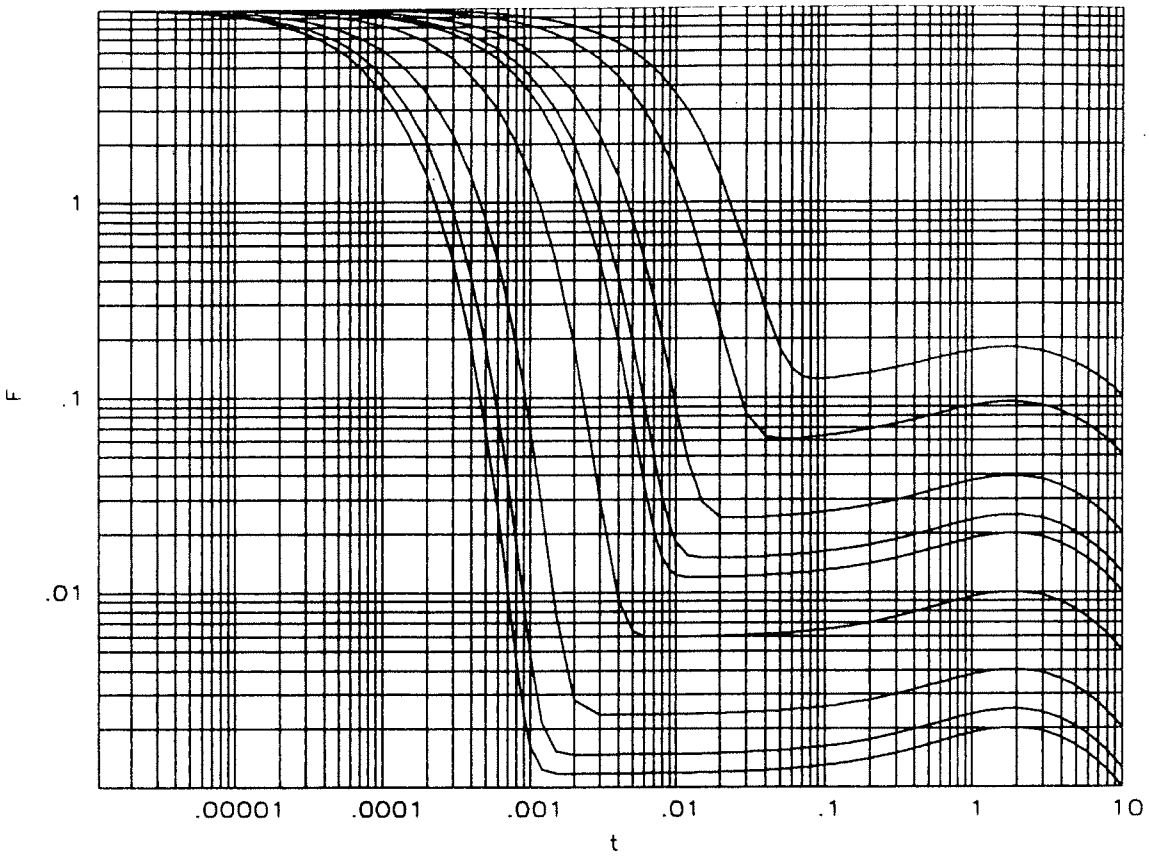


Bild 4:: Differenz der Simulationsergebnisse



Parametersimulation t-total=28.95 sec

Comparison 1 - SABER

Description of SABER

SABER is a well known Simulator for analog electronic systems, but is also useful for simulating analog or analog/digital systems of non-electrical or mixed type.

SABER is a product of Analogy Inc. and was published first in February 1987. The last release 3.2 was introduced in September 1993. The MAST modeling language is a de-facto standard for Analog HDL.

Model Description with MAST:

```
# EUROSIM Comparison 1
# Lithium-Cluster Dynamics under
# Electron Bombardment
#-----------------------------------------
# Language MAST (R), MAST is a registered
# Trademark of Analogy Inc.
# ----------------------------------------
# prepared by Rainer Mayer,
# Robert Bosch GmbH, Stuttgart, 25.4.94
# ----------------------------------------
#
number  kr = 1.0,
        kf = 0.1,
        lf = 1000,
        dr = 0.1,
        dm = 1.0,
        p  = 0
var nu  r, m, f
equations {
 r: d_by_dt(r) = -dr*r + kr*m*f
 m: d_by_dt(m) = dr*r - dm*m + kf*f*f -
                 kr*m*f
 f: d_by_dt(f) = dr*r + 2*dm*m - kr*m*f -
                 2*kf*f*f -lf*f + p
}
```

Task a) Comparison of integration algorithms:

All calculations have been done on a Sun SPARCstation 10 Model 402. SABER can be used in Graphical or in Command Mode. First an operating point (t=0) has to be defined, followed by a transient analysis (example with GEAR-algorithm):

```
dc (hold f 9.975 m 1.674 r 84.99

tr (te 10, ts 1m, terr 0.0001, terrn 6,
      steps VAR, meth gear, ord 2
```

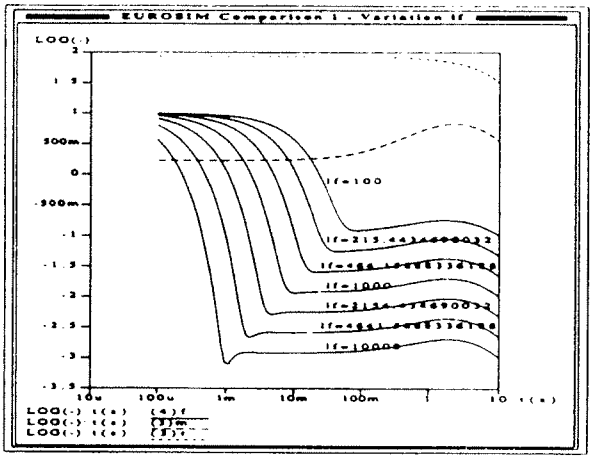
The CPU-times for different integration algorithms are:

Algorithm	tstep	CPU
Gear 2nd Order	var	0.33 sec
Gear 1st Order	var	0.75 sec
Trapez	var	0.75 sec
Gear 2nd Order	0.0005	47.30 sec
Gear 2nd Order	0.0010	21.20 sec

Task b) Variation of lf:

SABER offers a loop command for parameter variation, logarithmic scales are generated by postprocessing:

```
vary lf from 100 to 10000 log 7
        tr (tend 10, ts 1m, terr 0.0001
        end

extract / (pfile xlog, dfile tr, xs
     from 0.0001 to 10 log 300
```



Task c) Calculation of steady states:

Steady state was calculated by a second DC-Analysis with the operating point as a start value.

```
dc (dcip dc, dcep ep
display ep
alter p=10000
tr (dcip dc, dcep ep
display ep
```

The results are:

p=0:	f=0,	m=0,	r=0
p=10000:	f=10,	m=10,	r=1000

Dipl.-Ing. Rainer Mayer, Robert Bosch GmbH, D-70442 Stuttgart

EUROSIM Comparison 1
SABER-Implementation

26.4.199

Description of SABER

SABER is a well known Simulator for analog electronic systems, but is also useful for simu lating analog or analog/digital systems of non-electrical or mixed type.

SABER is a product of Analogy Inc. and was publicated first in February 1987. The last re lease 3.2 was introduced in September 1993. The MAST modeling language is a de fact standard for Analog HDL.

Model Discription:

```
# EUROSIM Comparison 1
# Lithium-Cluster Dynamics under Electron Bombardment
#-----------------------------------------------------
# Language MAST (R)
# MAST is a registered Trademark of Analogy Inc.
# ------------------------------------------------
# prepared by Rainer Mayer, Robert Bosch GmbH, Stuttgart
# 25.4.94
# ------------------------------------------------------
#
number  kr = 1.0,
        kf = 0.1,
        lf = 1000,
        dr = 0.1,
        dm = 1.0,
        p  = 0

var nu  r, m, f

equations {

      r: d_by_dt(r) = -dr*r + kr*m*f
      m: d_by_dt(m) = dr*r - dm*m + kf*f*f - kr*m*f
      f: d_by_dt(f) = dr*r + 2*dm*m - kr*m*f - 2*kf*f*f -lf*f + p
}
```

EUROSIM Comparison 1
SABER-Implementation

26.4.1994

SABER Runtime Commands:

SABER can be used in Graphical or in Command Mode.

Operating Point (t=0)

```
dc (hold f 9.975 m 1.674 r 84.99
```

Transient Analysis (Example)

```
tr (te 10, ts 1m, terr 0.0001, terrn 6, steps VAR, meth gear, ord 2
```

Postprocessing to generate log. x-Axis

```
extract / (pfile xlog, dfile tr, xs from 0.0001 to 10 log 300
```

Variation of lf

```
vary lf from 100 to 10000 log 7
        tr (tend 10, ts 1m, terr 0.0001
        end
```

Steady state for lf=1000

SABER offers no steady state analysis. Results are recieved by transient analysis with tend = 2000.

```
tr (te 2000, ts 1m
di tr
alter p=10000
tr (te 2000, ts 1m
di tr
```

Results:

All calculations have been done on a Sun SPARCstation 10 Model 402.

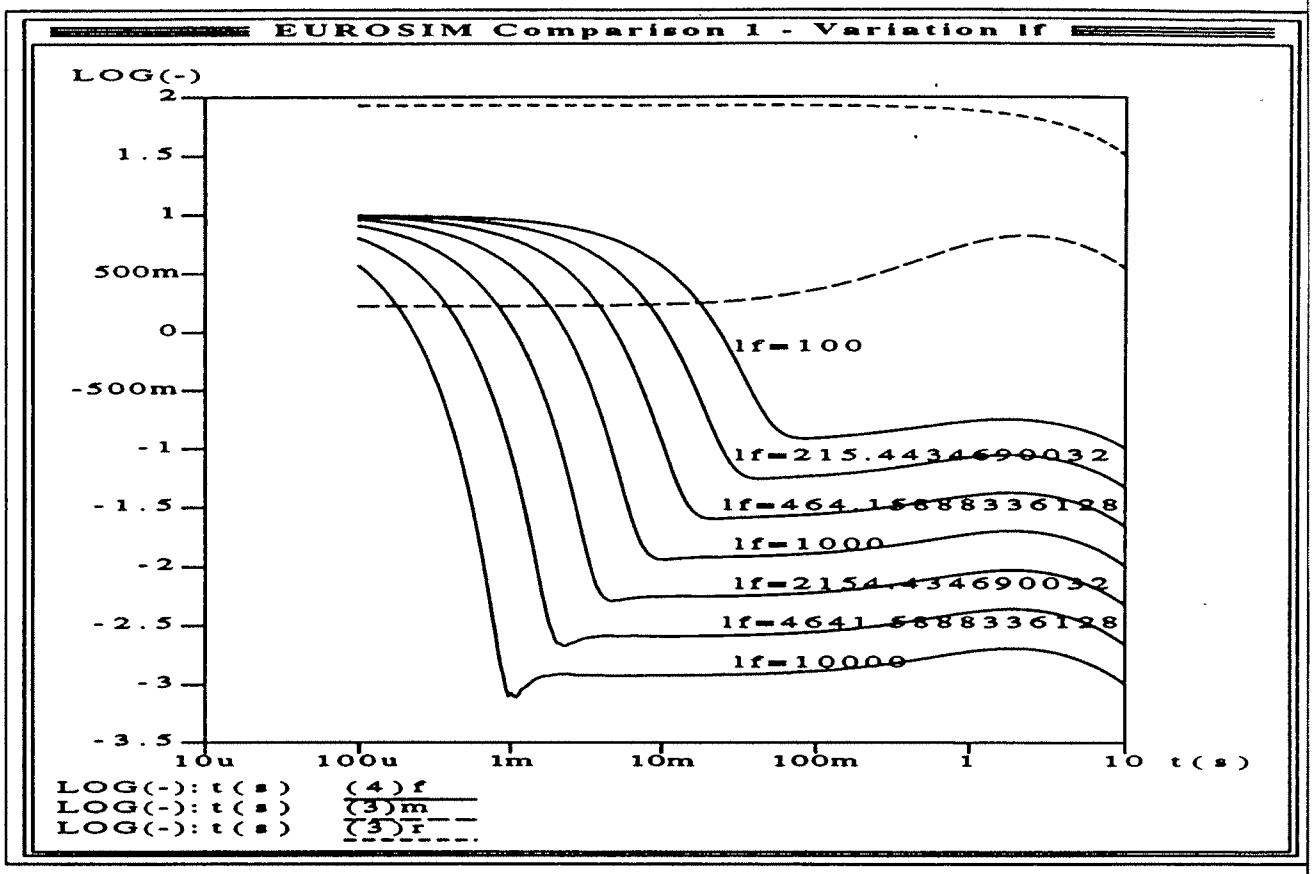
Comparison of CPU-Time

Algorithm	tstep	CPU
Gear 2nd Order	var	0.33 sec
Gear 1st Order	var	0.75 sec
Trapez	var	0.75 sec
Gear 2nd Order	0.0005	47.30 sec
Gear 2nd Order	0.0010	21.20 sec

EUROSIM Comparison 1
SABER-Implementation

26.4.19

Variation of lf



Steady State

p=0: f=0, m=0, r=0

p=10000: f=10, m=10, r=1000

Comparison 1 - SIMNON

SIMNON is a simulation software for both continuous and discrete systems, which translates programs, very quickly, directly into memory (available for UNIX,VMS and PC [DOS and Windows 3.1]). Additionally, SIMNON provides "connecting systems" to establish interconnections between various subsystems, which makes it quite easy, when dealing with larger systems, to decompose them into subsystems. There exists a real-time version where subsystems may be hardware-in-the-loop modules.

The concept of SIMNON also includes the possibility to handle model parameters and terminal values of model variables within a relatively powerful experiment language with built-in macro functions.

In order to overcome problems with the stiff system and to obtain a logarithmic scale, a transformation like in [1] has been made.

Model description:

SIMNON uses an equatio oriented model description, where state variables and derivative variables have to be defined explicitely:

```
CONTINUOUS SYSTEM MOL
* Lithium Cluster Dynamics - EUROSIM Comparison 1
* States, derivates and time:
STATE R M F
DER derR derM derF
TIME tau
* Equations:
"Test for stationarity:
test = ((abs(derR)<eps) AND (abs(derM)<eps))
st = IF sttest THEN CTERM((abs(derF)<eps) AND test ) ELSE 0
ln10 = ln(10)
const = ln10/10^tau0
derR = (-dr*R + kr*M*F)*tt
derM = (dr*R - dm*M + kf*F*F - kr*M*F)*tt
derF = (dr*R + 2*dm*M - kr*M*F -2*kf*F*F - lf*F + p)*tt
tt = IF sttest THEN 1 ELSE const*10^tau
lgR = (ln(R)/ln10)
lgM = (ln(M)/ln10)
lgF = (ln(F)/ln10)
* Parameter values:
kr: 1
kf: 0.1
lf: 1000
dr: 0.1
dm: 1
tau0: 3
p: 0
sttest: 0
eps: 1e-3
"Initial values:
F: 9.975
M: 1.674
R: 84.99
END
```

Task a) Comparison of integration algorithms:

SIMNON has only four integration algorithms, there exists in particular no implicit algorithm, which is of course a disadvantage in the case of a stiff system like the one discussed here. For time measurements the program above, which contains the logarithmic transformation, was used. The following table shows the results.

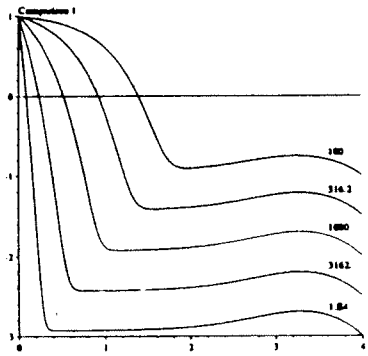
algorithm [time(min:sec)]	'286 (16 Mhz)	'386/7 (40 MhZ)	'486 (66 MZz)
RKF45	4:46,9	0:8,2	0:2,7
RKF23	6:26,3	0:12,1	0:4,2
DOPRI45R	6:39,2	0:12,0	0:3,9
EULER	--	0:31,0	0:9,8

RKF23/RKF45: Runge-Kutta-Fehlberg algorithm of orders 2/3 and 4/5
DOPRI45R: Runge-Kutta algorithm due to Dormand and Prince
(all with automatic stepsize adjustment)
EULER: Euler-algorithm with fixed stepsize

Task b) Variation of lf:

SIMNON offers parameter variation and programming with experiment variables at runtime level. The following commands load the model (SYST), change accuracy parameters, draw titles and axes and perform the parameter variation in a loop (FOR lflog ... NEXT lflog) where SIMU starts a simulation run:

```
SYST mol
ERROR 1e-7
PLOT lgf
AXES H 0 4 V -3 1
FOR lflog=2. TO 4. STEP 0.5
LET lf2 = 10^lflog
WRITE lf2
PAR lf: lf2
SIMU 0 4
MARK :lf2
NEXT lflog
```



Task c) Calculation of steady states:

Although there is no built-in steady state finder in SIMNON, it is nevertheless possible to "simulate" a steady-state finder using a combination of infinite simulation (SIMU INF) and conditional termination (CTERM), which produces acceptable results:

P	r	m	f
10000	998.93	9.9903	10.
0	9.9973E-3	1.1108E-3	3.2105E-6

The commands (for P=10000) are:

```
PAR lf: 1000
PAR p: 10000
PAR sttest: 1
SIMU 0 INF              "Infinite simulation
DISP F M R tau
```

Reference: [1] G.A. and T.M. Korn, Comparison 1 - DESIRE, EUROSIM SNE, No 4 March 1992, P. 30

M.Bracke, S.Schnitter, A.Schreiber, Insitut für Informatik, TU Clausthal, Erzstr.1, D-38678 Clausthal-Zellerfeld

Comparison 1 - SIMNON

SIMNON is a simulation software for both, continuous and discrete systems, which translates programs, very quickly, directly into memory (available for UNIX ,VMS and PC [DOS and Windows 3.1]).
Additionally, SIMNON provides "connecting systems" to establish interconnections between various subsystems, what makes it quite easy, when dealing with larger systems, to decompose them into subsystems.
The concept of SIMNON also includes the possibility to handle model-parameters and terminal values of model variables within a relatively powerful experiment language with built-in macro functions.
infinte simulation and the conditional termination of a simulation.

In order to overcome problems with the stiff system and to obtain a logarithmic scale, a transformation like in [1] has been made.

Model description:

```
CONTINUOUS SYSTEM MOL
" Abstract:     Comparison 1
" Description:  Lithium-Cluster Dynamics
"               under Electron Bombardment
  Author:       M.Bracke, S.Schnitter, A.Schreiber

" States, derivates and time:
STATE R M F
DER derR derM derF
TIME tau
" Equations:
"Test for stationarity:
test = ((abs(derR)<eps) AND (abs(derM)<eps))
st = IF sttest THEN CTERM((abs(derF)<eps) AND test ) ELSE 0

ln10 = ln(10)
const = ln10/10^tau0
derR = (-dr*R + kr*M*F)*tt
derM = (dr*R - dm*M + kf*F*F - kr*M*F)*tt
derF = (dr*R + 2*dm*M - kr*M*F -2*kf*F*F - lf*F + p)*tt
tt = IF sttest THEN 1 ELSE const*10^tau
lgR = (ln(R)/ln10)
lgM = (ln(M)/ln10)
lgF = (ln(F)/ln10)

" Parameter values:
  : 1
um: 1
kf: 0.1
dr: 0.1
lf: 1000
tau0: 3
F: 9.975
M: 1.674
R: 84.99
p: 0
sttest: 0
eps: 1e-3
END
```

Task a) Comparison of integration algorithms:

SIMNON has only four integration algorithms, there exists in particular no implicit algorithm, which is of course a disadvantage in the case of a stiff system like the one discussed here. For time measurements the program above, which contains the logarithmic transformation, was used:

algorithm [time(min:sec)]	`286 (16 Mhz)	`386/7 (40 Mhz)	`486 (66 Mhz)
RKF45	4:46,9	0:8,2	0:2,7
RKF23	6:26,3	0:12,1	0:4,2
DOPRI45R	6:39,2	0:12,0	0:3,9
EULER	--	0:31,0	0:9,8

RKF23/RKF45 : Runge-Kutta-Fehlberg algorithm of orders 2/3 and 4/5
DOPRI45R : Runge-Kutta algorithm due to Dormand and Prince
(all with automatic stepsize adjustment)

b) Variation of lf:
SIMNON offers paramater variation and programming with experiment variables at runtime level. The following commands load the model (SYST), change accuracy parameters, draw titles and axes and perform the parameter variation in a loop (FOR lflog ..... NEXT lflog). Results are shown in fig.1.

```
SYST mol
ERROR 1e-7
NEWPLOT
PLOT lgf
AXES H 0 4 V -3 1
TEXT 'Comparison 1'
FOR lflog=2. TO 4. STEP 0.5
LET lf2 = 10^lflog
WRITE lf2
PAR lf: lf2
  4U 0 4
GIN
MARK A xs. ys.
MARK :lf2
NEXT lflog
PLOT
MSGBOX 'Ready to find steady state...'
PAR lf: 1000
PAR P: 0
PAR sttest: 1
SIMU 0 INF           "Infinite simulation
DISP F
DISP M
DISP R
DISP tau

END
```

plot.eps

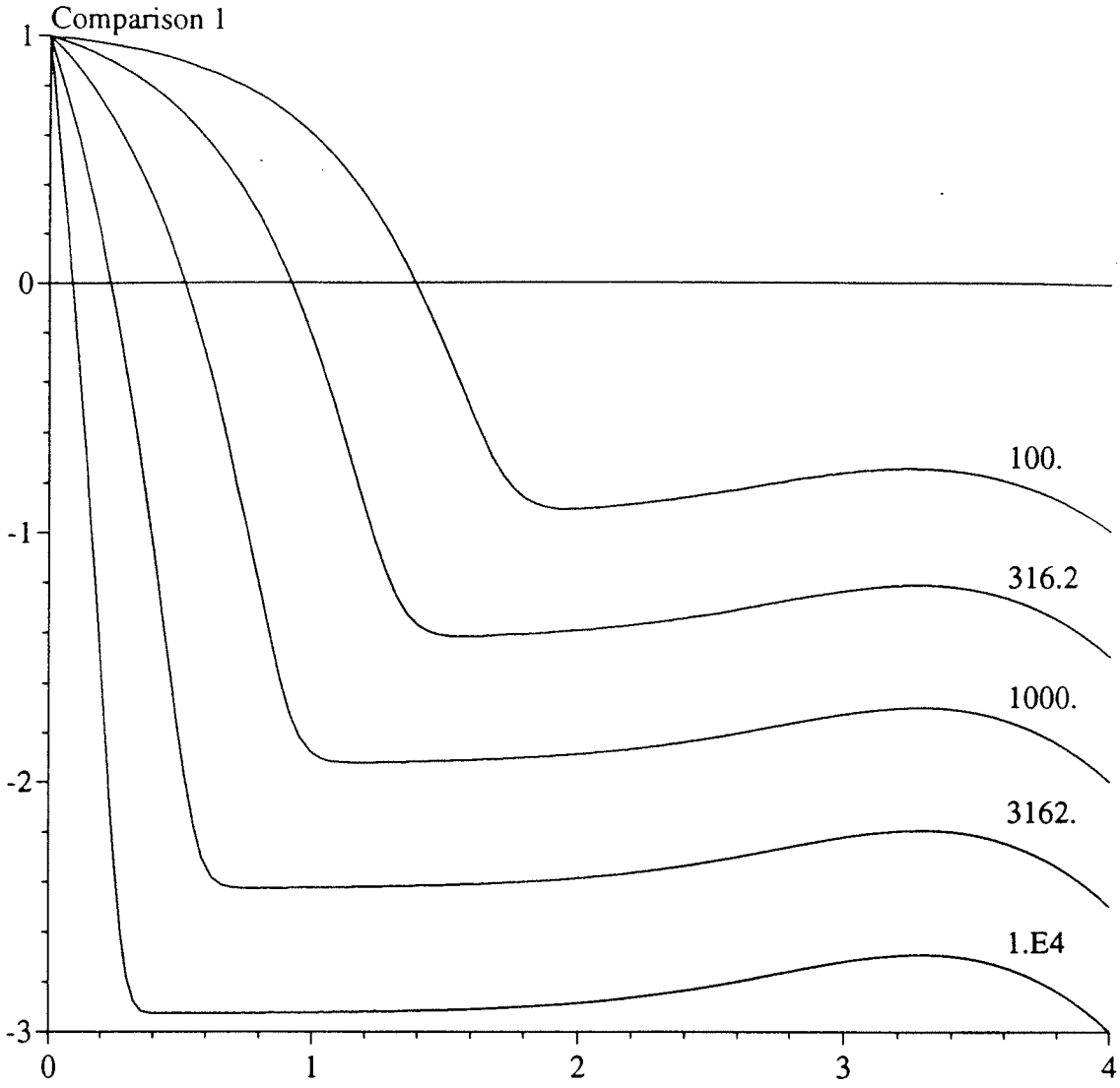
c) Calculation of steady states:

Although there is no built-in steady state finder in SIMNON, it is nevertheless possible to "simulate" a steady-state finder using a combination of infinite simulation (SIMU INF) and conditional termination (CTERM) (see program / macro), which produces acceptable results:

P	r	m	f
10000	998.93	9.9903	10.
0	9.9973E-3	1.1108E-3	3.2105E-6

The commands (for P=1000) are:

```
PAR lf: 1000
PAR P: 0
PAR sttest: 1
SIMU 0 INF           "Infinite simulation
DISP F
DISP M
DISP R
DISP tau
```


Comparison 1
1
0
-1
-2
-3
0
1
2
3
4
100.
316.2
1000.
3162.
1.E4

Comparison 1 - **mosis**

mosis (modular simulation system) is an experimental CSSL simulation language (equation-oriented) designed for modular simulation development with features for parallelization on MIMD-systems with distributed memory (see Parallel Comparison in SNE 11). mosis (developed at the Dept. of Simulation Technique, TU Vienna) is a general purpose compiling simulation language of CSSL-type on a C basis, not only for parallel programming techniques.

The simulation kernel provides several integration algorithms, a state event finder and a time event queue (all calculations in double precision number format). The runtime system also contains a powerful interpreter language where even complex algorithms can be programmed, furthermore graphical output and some routines for frequency domain analysis.

At runtime several instances of models can be connected and simulated as one big model. These instances can be created on the same processor ("serial simulation") or at different processors ("parallel simulation"), where communication is performed automatically .

mosis can be freely copied and used for non-commercial purposes (the complete and unlimited version can be obtained from the simulation server `simserv.tuwien.ac.at` at the TU Vienna by "anonymous ftp"; commercial use on request). It has been implemented on PCs, UNIX-workstations under PVM and X Window and the Cogent XTM transputer system.

Model description: The model `lithium` is defined in the file "lithium.m", translated, compiled and linked to the runtime-system; state variables and parameters must be defined explicitly:

```
model lithium() {
state r,m,f;
param kr=1.0,kf=0.1,lf=1000.,dr=0.1,dm=1.;
param f0=9.975, m0=1.674, r0=84.99, p=0.0;
preinitial {ialg=3;tend=10.;dt=cint=0.001;}
 derivative {
 r'=-dr*r+kr*m*f                    ,r0;
 m'=dr*r-dm*m+kf*f*f-kr*m*f         ,m0;
 f'=dr*r+2*dm*m-kr*m*f-2*kf*f*f-lf*f+p,f0;}}
```

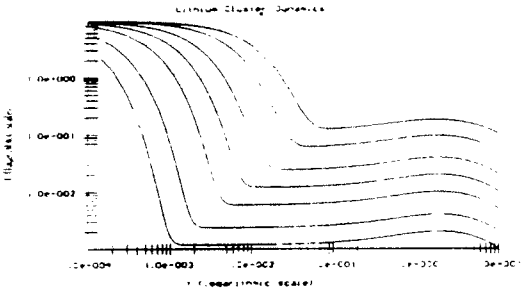
The following runtime commands instance the model `lithium` once (on an arbitrary processor, indicated by "-1"), identifying the instance with the handle `lit`, choose the integration algorithm, and simulate the model (`run`) with storing the state f (`watch`):

```
int lit; lit=instance("lithium",0);
l.ialg=8; // stiff integration algorithm
watch(l.f); run(l);
```

Task a) Integration algorithms: mosis offers various integration algorithms. The simulation results for these algorithms are summarized in the following table (* ... no stepsize control, ** semi-implicit extrapolation method by Bader and Deuflhard); results computed on a 486/33 processor, 8MB, 32-Bit version.

Algorithm	Stepsize	max.abs. error	Time
Euler	1.0E-3	*	2.3 sec
RK2	1.0E-3	*	4.5 sec
RK4	1.0E-3	*	4.1 sec
RK4	1.0E-4	*	41.3 sec
Adams-M.	1.0E-4	1.0E-8	2.58 sec
RKF	1.0E-4	1.0E-8	2.52 sec
Stiff Alg.**	1.0E-4	1.0E-8	0.089 sec
Stiff Alg.**	1.0E-4	1.0E-6	0.058 sec

Task b) Parameter study: A parameter study is performed by a C-like loop command, where an array stores the different values for the parameter `lf`. Seven runs are stored and then plotted:



```
double x[7]= { 100, 200, 500, 1000, 2000,
               5000, 10000 };
watch(lit.f);$scalex=$scaley=1;   log.scales
for(i=0;i<7;i++) { lit.lf=x[i]; run(lit);}
drawcurve(lit.f);
```

This parameter loop could be done in parallel, if a multiprocessor system is available (with nearly linear speed up). The model `lithium` has to be instanced seven times on different processors (no. 0 - 6) and run in parallel:

```
int lita[7]; double x[7]= { 100, 200, 500,
                1000, 2000, 5000, 10000 };
for(i=0;i<7;i++) {lita[i]=
    instance("lithium",i); watch(lita[i].f);}
for(i=0;i<7;i++)    {lita[i].lf=x[i];
                     run(lita[i],0);}
for(i=0;i<7;i++) drawcurve(lita[i].f);
```

Task c) Steady state calculation: mosis offers a trim command (with various parameters for accuracy, etc.) The commands `lit.p=0; trim(lit); lit.p=10000; trim(lit);` give results summarized in the following table.

	f	m	r
p = 0	2.720E-17	1.533E-11	-1.734E-10
p = 10000	10	10	1000

G. Schuster, F. Breitenecker, ARGE Simulation News, c/o Dept. Simulation Techniques, TU Vienna, Wiedner Hauptstr. 8-10, A-1040 Vienna, Austria, Email: argesim@simserv.tuwien.ac.at.

Comparison 1 - SIMNON

SIMNON is an easy to handle simulation tool. Models are described as continuous or discrete systems in the Editor-Window. There is no matter about sorting statements; this is done by SIMNON when the system is activated, that means translated into machine-code. If there are errors in the program, SIMNON will stop the translation and shows the line where the error occurs for the first time. Models can also be built up by connecting discrete and/or continuous subsystems, that means a very easy to survey structure. SIMNON is also capable of real-time-simulation, e.g. to control a physical process. The simulation is started either by mouse control or with a command in the command-dialog window. There you can also change parameters, select integration algorithms and also give the commands for plotting graphics in a plot window. For this Comparison we used SIMNON/PCW, Version 1.1 for MS Windows 3.1.

Model description: The following model was built up by using the predefined program mask. It would also be possible to write all the equations, parameter- and initial-values without sorting.

```
CONTINUOUS SYSTEM LICLU
" States and derivates:
STATE r m f
DER rdot mdot fdot
" Initializations:
r:64.99
m:1.674
f:9.975
" Equations:
rdot=-dr*r+kr*m*f
m+kf*f*f-kr*m*f
fdot=dr*r+2.0*dm*m-kr*m*f-2.0*kf*f*f-lf*f+p
lf=10^lfp
" Parameter values:
kr:1.
kf:0.1
dr:0.1
dm:1.
lfp:2.
p:0
END
```

a) **Comparison of integration algorithms:** SIMNON offers four integration algorithms: two of Runge-Kutta type (RKF23 and RKF45) a Dormand-Prince-algorithm (DOPRI45R) and the Euler algorithm (EULER). All of them are working with automatic step size, only Euler works with fixed step size. The stiff system was simulated with a 386DX-25MHz-PC with 387 coprocessor with all of the four algorithms. The results for a period of 10s with constants lf=1000, p=0 and error tolerance 1e-3 are presented in the table below:

algorithm	max length of a step	time
Euler	0.001	23s
RKF23	auto	21s
RKF45	auto	fpe
RKF45	0.01	15s
DOPRI45R	auto	fpe
DOPRI45R	0.01	26

fpe=floating point error

Since there is no special algorithm for stiff systems it was necessary to make experiments by varying the error tolerance and stepsize.

b) **Parameter variation:** This can be done interactively in the command-dialog window by formulating an assignment loop:

In order to plot the F-centre concentration (f) scaled logarithmic as a function of time (also scaled logarithmically) we had to supply the following lines to the program:

```
TIME t
lgt=log(t)
lgf=log(f)
```

After simulating with the Runge-Kutte-23 algorithm with automatic stepsize from 0.001 to 10 seconds and error tolerance 0.001 and the parameters lfp=2, 2.5, 3, 3.5, and 4 we could plot the following diagram.

lfp	computation time
2	4s
2.5	10s
3	30s
3.5	93s
4	291s

F-center-concentration as a function of time.

c) **Steady state calculation:** SIMNON has no special algorithm for steady-state finding. So we had to simulate the system over a long period and to terminate for instance with CTERM (Conditional **Termination**). We defined the condition with abs(fdot^2+rdot^2+mdot^2) < 0.001 and started the experiment with the same integration parameters as in b) and lf=1000. For p=0 the program stopped at t=56.2481 with

rdot	mdot	fdot
-0.031428	-0.00349104	-0.0000101248
r	m	f
0.314315	0.034919	0.000101276

For p=10000 we stopped the program after a computation time of about 8 hours at t=1269 with.

rdot	mdot	fdot
-1.58117	1.28698	161.542
r	m	f
997.708	9.97798	9.84063

Conclusion: Although SIMNON is a valuable simulation tool, in this example the lack of a Gear algorithm and of logarithmic plots is evident.

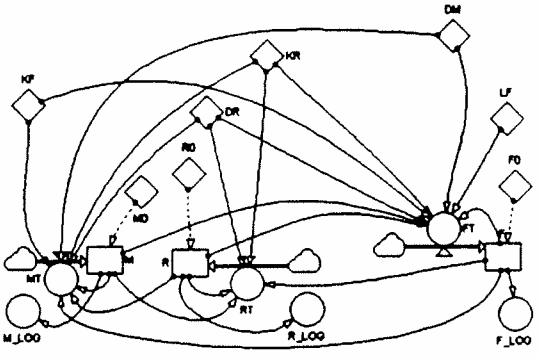
J. Plank, Strudlhofg. 5, 1090 Vienna.

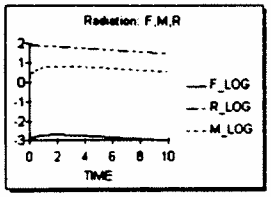
Comparison 1 - POWERSIM

POWERSIM is a Windows based simulation program for modelling and simulation of dynamic systems. A mouse and menu driven input facility allows to construct block diagram models, to control the experiments, and to process output data.

POWERSIMs modelling philosophy is based on the System Dynamics Approach. Main element in designing models is the "Level"-element, whose value is incrementally changed during a simulation. A Level is an "accumulator" (integrator), receiving flows of input or delivering flows for output (rates) from timestep to timestep. The causal connections between levels and rates are realized by links which show the direction of flow of data. The results of simulation can be presented by charts and tables, also within the modelling layout.

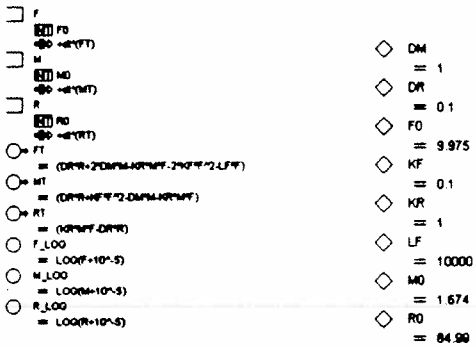
Model Description: The following "worksheet" shows the model definition and results of the problem under investigation. In the modelling layout rectangles define the levels (the state variables f, m and r), circles define auxiliary variables (internally defined by a user-defined formula and acting as rate, if fixed to a flow arrow; in this case the nonlinear terms of the equations), and squares define parameters; intial values for the levels (the state variables) are defined constants fixed to the levels by dashed lines. Results may be displayed as graphs or as tables:





TIME	F	FT	M	MT
0.0	9.98	-99 774.75	1.67	0.0799
0.1	0.0013	0.00113	2.32	6.09
0.2	0.00141	0.001	2.88	5.43
0.3	0.00151	0.000884	3.41	4.84

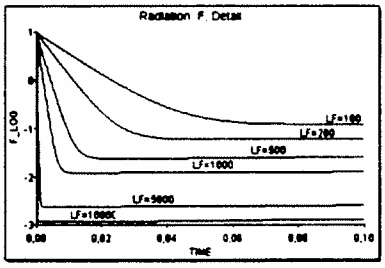
In addition to the models graphical definition the corresponding (automatically generated) equations can be viewed:



Results: Task a) The table shows the computing times using a 486 DX2/66 PC; POWERSIM doesn't support special integration algorithms for stiff systems (fixed stepsize 0.001):

Integration Algorithm	Comp.Time
Euler	32 s
Runge Kutta 2nd order	34 s
Runge Kutta 3rd order	36 s
Runge Kutta 4th order (fixed stepsize)	38 s
Runge Kutta 4th order (variable stepsize)	40 s

Task b) One feature of POWERSIM is the use of co-models, which can be synchronized with a main model. Automatic parameter variations may be defined in such co-models as a loop over the model under investigation, making it easy to collect data of multiple runs and display them together (the parameter lf was varied by values 100, 200, 500, 1000, 5000, 10000):



Task c) The calculation of steady states can only be done using long-term simulations. The following table shows the results at time t_1=500 and time t_2=1000, given in terms of the order of the error $O(10^p)$ for the solution f=r=m=0 in case of p_1=0 and given as absolute values for the solution f=m=10 and r=1000 in case of $p_2=10^4$.

State	p_1, t_1	p_1, t_2	p_2, t_1	p_2, t_2
f	$O(10^{-9})$	$O(10^{-18})$	9,9997	10,0
m	$O(10^{-9})$	$O(10^{-18})$	9,9046	9,9989
r	$O(10^{-9})$	$O(10^{-18})$	990,28	999,88

K. Scheidenberger, K. Schleiss, F. Breitenecker, Dept. Simulation Techniques, TU Vienna, Wiedner Hauptstraße 8-10, A-1040 Vienna, Austria.

Comparison 1 - IDAS / SIMPLORER

Description of IDAS

IDAS 3.01 for WINDOWS is a powerful software package mainly designed for the simulation of electronic circuits and control problems with a physical background.

Modelling may be carried out in three different ways:

- by dialog in WINDOWS-technique (easy and comfortable)
- textually in IDL (Idas Description Language)
- graphically with an additional program (e.g. ORCAD, PROTEL,...)

IDAS also provides a data analysis program called DAY, where the results can be evaluated mathematically and plotted in different ways.

Recently IDAS was extended and given the name SIMPLORER. SIMPLORER consists of
- a circuit simulator
- a signal flow graph simulator
- a state graph simulator

Some new features have been added, e.g.
- FUZZY - Control Module
- C-Programming interface
- Optimizer for automatic parameter-variation according to a predefined system behaviour
- Frequency response module etc.

The simulation was still carried out by IDAS on a Pentium 60mHz under Windows 3.11 for Workgroups.

Model description

For the simulation in IDAS a block diagram (signal flow, graph) of the given equations must be worked out. IDAS itself does not provide any possibility to show the block diagram graphically. The model was implemented by dialog in windows-technique.

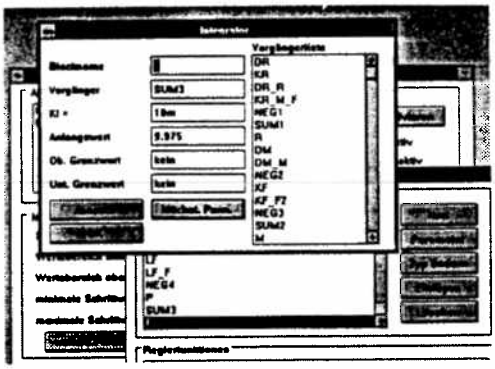
Results

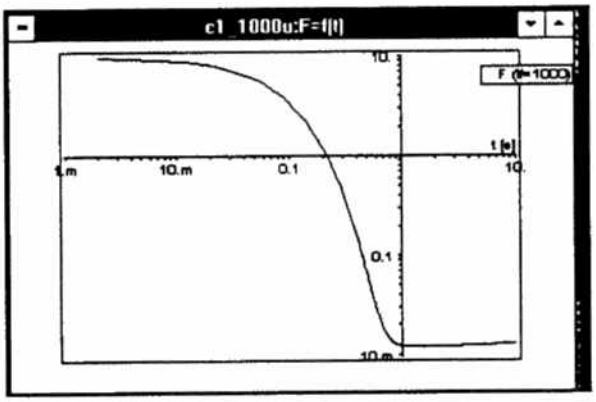
a)*Comparison of integration algorithms:*

IDAS provides two different algorithms: Euler and Trapezoidal. With a minimum step size of 0.002 and a maximum step size of 0.01 the results were nearly the same: The simulation run (including compilation and graphic output) needed approx. 8 seconds with both algorithms. Changing the step size did not show a significant influence on the output.

b)*Variation of parameter l_f:*

The system was simulated over 10 seconds with values of l_f equal to 100, 1000 and 5000 and plotted with the data analysis program DAY, with logarithmic scales as required. Unfortunately the results for l_f =10000 proved to be numerically unstable.





The last solution vectors (at t=10) were:

	r	m	f
l_f=100	84.327	2.1962	0.12481
l_f=1000	84.167	2.3069	1.2989E-2
l_f=5000	84.15	2.3184	1.3048E-3

c)*Calculation of steady states:*

As IDAS does not provide any instrument to calculate steady states the differential equations were solved in the interval $0 < t < 10000$ for p=0 and $0 < t < 30000$ for p=1E4. The last solution vectors were:

	r	m	f
p=0	4.9281E-3	5.4756E-4	1.5895E-5
p=1E4	937.51	9.4326	9.9983

Gerhard Stefan, TU Vienna, Dept. Simulation Techniques

EUROSIM Comparisons

Publication of Solutions

July 1995

	C1	C2	C3	C4	C5	C6	C7	CP
SNE 0	Def							
SNE 1	5	Def						
SNE 2	4	4	Def					
SNE 3	4	3	3	Def				
SNE 4	1	5	5	3	Def			
SNE 5	4	-	1	1	2			
SNE 6	-	2	-	2	1	Def		
SNE 7	1	2	1	2	-	1	Def	
SNE 8	-	1	-	-	-	1	3	
SNE 9	-	-	-	-	-	2	3	
SNE 10	1	2	-	-	-	2	2	Def / 1
SNE 11	2	2	1	-	1	-	-	2
SNE 12	1	-	1	-	-	-	2	3
SNE 13	-	-	-	-	-	-	3	1
SNE 14	3	-	1	-	-	-	2	-
Total	**26**	**21**	**13**	**8**	**4**	**6**	**15**	**7**

*A sophisticated parallel High-Performance CSSL-Simulation Language at **ABSOLUTELY NO COST!***

the modular simulation system

Now introducing Level 2:

- ✓ *Parallel CSSL-Simulation system on compiler-basis guarantees extremely high simulation speed, parallelization (or multitasking on single-processor systems) by modular model development.*
- ✓ *Multiple operating system environments: UNIX / PVM (possibility to build clusters) plus XWindow system (opt.), DOS, DOS/32, Windows (very fast; uses the Watcom C/C++ compiler); Windows 95, Win NT will follow soon.*
- ✓ *Easy-to-learn, clear and powerful model description language on a "C"-basis including a comfortable macro expander (superset of "C"-preprocessor).*
- ✓ *Level 2 includes object-oriented model description (Inheritance) and handling of DAEs.*
- ✓ *Very fast translator and a comfortable make-utility (Windows: Integrated Development Environment) grant short development cycles*
- ✓ *Powerful run-time interpreter language (C-like) with many experimentation commands, easy user-extendability. Background calculation enables concurrent simulation of several models.*
- ✓ *Graphical Modelling Environment under Windows is also able to produce code for other common simulation systems; many other developments available or will follow soon.*
- ✓ *Qualified support for commercial users including model development, user extensions, hardware implementations, Real-Time-Simulation etc. available from Advanced Technical Software GmbH*
- ✓ mosis *is distributed as freeware; it can be freely copied and used without any cost. The complete package can be obtained from: <URL: ftp://simserv.tuwien.ac.at> or the WWW-server: <URL: http://eurosim.tuwien.ac.at>*

For general questions regarding mosis, contact:

DI Dr. Günter Schuster
ARGESIM, c/o Dept. Simulation Techniques,
attn. F.Breitenecker, Technical University Vienna
Wiedner Hauptstraße 8-10, A-1040 Wien, AUSTRIA
Tel.: +43-1-58801-5374,..-5386,..-5484, Fax: +43-1-5874211
E-Mail: argesim@simserv.tuwien.ac.at
guenter@osiris.tuwien.ac.at

ARGESIM

For questions regarding the commercial use of mosis, please contact:

Advanced Technical Software GmbH
Flurschützstraße 16/10
1120 Wien - AUSTRIA
Tel.: +43-1-8156675
Fax: +43-1-8156676

ATS

See the Demonstration at the EUROSIM´95 congress!