Comparison of available software for dynamic modeling

Jaap J. van Milgen, R Boston, R Kohn, J Ferguson

▶ **To cite this version:**

Jaap J. van Milgen, R Boston, R Kohn, J Ferguson. Comparison of available software for dynamic modeling. Annales de zootechnie, 1996, 45 (Suppl1), pp.257-273. hal-00889623

HAL Id: hal-00889623
https://hal.science/hal-00889623v1

Submitted on 11 May 2020

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Ann Zootech (1996) 45, Suppl, 257-273
© Elsevier/INRA

Comparison of available software for dynamic modeling

J Van Milgen[1], R Boston[2], R Kohn[2], J Ferguson[2]

[1]*Institut National de la Recherche Agronomique, Station de Recherches Porcines, 35590 Saint-Gilles, France*
[2]*University of Pennsylvania, School of Veterinary Medicine, New Bolton Center, 382 West Street Road, Kennett Square PA 19348-1692, USA*

Summary - During the last few years, mathematical modeling of nutritional problems has become increasingly popular due to rapid development and availability of both computer hardware as well as problem-oriented software packages. In this review we discuss software tools that can be used for solving systems of differential equations. Four software packages (STELLA, Scientist, SAAM, and ACSL/SimuSolv) were used to re-create a model describing a full 300-day lactation period in a dairy cow. The model included a continuous milking function (simulating calf drinking) or a discrete milking function (machine milking). No software package met all the requirements for efficient model development. Because different modelers have different modeling objectives, no specific criteria for software selection can be given. For occasional modelers, ease-of-use and integration with other packages under the same operating system can be important issues. On the other hand, performance and flexibility are likely to be key criteria if large and complex models are to be created.

Introduction

Modeling is not a new science. Ever since people have been involved in research, conceptual, qualitative models have been built. However, with the onset of the computer era, the quantification of conceptual models has become increasingly popular. Today, many nutrition models are used in research, education, and extension. In research, models are typically developed to integrate existing knowledge about a system. Putting the pieces of the puzzle together may identify sensitive parts of the system and can expose lack of knowledge in certain areas. The resulting mechanistic models are used as a guide directing future research. These mechanistic models can also be used in education to illustrate biological phenomena. The main reason to develop models for agricultural extension is the prediction of production. The consequences of different scenarios can be evaluated before management decisions are taken. In general, extension models tend to be more empirical in nature than research models.

In order to characterize a system, two different approaches can be employed. By far the most frequently used approach is the one where a (sub)system and its model is represented by a single, explicitly defined equation. The model, more or less empirical in nature, is characterized by a limited number of parameters which are obtained by fitting the model to the data using nonlinear parameter estimation procedures. Alternatively, in simulation modeling, a system is described by a (large) number of (differential) equations and parameters for which an explicit solution may not exist. Although both approaches have their niche in scientific research, we will focus here on software that is to be used in latter approach.

Dynamic model development

The changing user environment

Development of computers has made it possible to solve complex mathematical and logical operations rapidly and efficiently. The first analog computers became commercially available in the late forties. However, widespread utilization was limited due to cost and availability. A major breakthrough came in the late seventies when personal computers were developed. To illustrate the rapid development, the first Intel processor (8086) was introduced in 1978 and contained 29000 transistors whereas the first Pentium processors (1993) contained 3.1 million transistors. In 1979, an 11 MB hard drive cost almost $1800 (which is

currently the price of a 4.3 GB drive).

Not only the hardware but also the software environment has changed dramatically bringing comprehensive and effective methods to nutritional modelers. The first nutritional models were created using procedure-oriented programming languages such as FORTRAN, BASIC, and PASCAL. As a result, creation of models was restricted to those nutritionists willing to spend resources in computer programming. The advantages of procedure-oriented languages (generality, flexibility) have to be weighed against its disadvantages (all routines have to be programmed). Today, several problem-oriented computer packages exist that may help researchers in their modeling efforts. Many of these require little or no background information about computer programming. Moreover, some incorporate user-friendly interfaces and are targeted towards a large scientific audience. They range from easy-to-use packages that can be run on personal computers to sophisticated packages that need the computational power of a mini-computer or mainframe. The choice between a procedure-oriented and problem-oriented language depends largely on the modeling objective. Procedure-oriented languages are ideally suited to building models that are intended to be used rather than developed (e.g., extension models). In contrast, if the objective is model development, the use of a problem-oriented language will allow the researcher to focus on the (changing) problem rather than on programming procedures.

The portents of these changes

Where computers were once tools that required specialized personnel, today a computer is part of standard office equipment. Computers and problem-oriented simulation software are within the financial reach of most researchers. Although models can add value to biological research, it is important to realize that modeling remains a costly and time-consuming exercise. Computers can help scientists in their modeling efforts but do not substitute for skills in mathematics, statistics, and computer science. Investments in these areas have to be made in order to create quantitative nutrition models. It is important to realize that the computer is an extension of the human mind, and not a replacement for it.

Requirements of modeling software for the efficient development of models

In as much as models are developed in the nutrition area for a wide variety of quite legitimate reasons, software facilitating that development needs to address the specific nuances of the diversity of purposes. And whereas software of limited scope and flexibility can often be used by skilled investigators in sophisticated and flexible ways, users rapidly appreciate the additional power and utility of software less limited in scope and flexibility. In this section of our review we rank software features in regard to their capacity to facilitate aspects of model development and model analysis.

User interface

If modeling is to be accessible as a scientific technique to the widest appropriate assembly of nutrition researchers, the tools allied to it need to be cast in syntactical and semantic forms to which such investigators are, or can become quickly accustomed. Here we see the pre-eminence of the software user interface surfacing ... modeling software with its diversity of responsibilities needs to offer the user a consistent, reliable, and predictable environment within which research efforts may be efficiently and productively advanced. To this end software needs to address the following:

-predictable and expandable environment
-menu organization facilitating access to critical software functionality
-appropriate iconification of modeling objects
-intuitive functional structure underpinning graphical objects
-comprehensive graphical manipulation of modeling objects
-task grouping for efficient yet protective software use
-context sensitive and expert level user help
-consistent use of fonts, colours, and navigational controls
-capacity to reverse/undo and log actions
-accelerated mode of use via hot keys and command language

Data manipulation

One of the major activities associated with nutrition modeling involves the interpretation and analysis of nutrition and related data. In

order to accomplish this step using modeling software we need ways to represent and manage data. For this, software (to varying degrees) support the following:
-entry of data
-data storage and retrieval
-statistical characterization and weight assignment
-data screening, summarization, inspection, and description
-representation of different data forms (e.g., string, date, enumeration)
-transformation and auto-transformation (e.g. normalization)
-block manipulation (e.g., include/exclude, expose/hide, transpose, join)

Model specification

To efficiently build and manipulate models calls for an array of modeling constructs. The richer, or more extensive, this array the greater the diversity of model forms that can be represented. The more symbolic and functionally intuitive the modeling constructs are the greater ease we will experience in assembling the model. Nutrition models in particular have demonstrated a need for a wide variety of model components, in part because of the need to represent processes from the sub-cellular level to the whole animal level, and in part because of the need to represent processes themselves ranging from growth, to transport, from production to excretion. To build and manipulate models, software usually supports the following model management steps and tasks:
-entry
-inspection and modification
-retention and retrieval
-automated diagnostic analysis
-compilation and allied integrity analysis
-depiction and printing
-functional linkage to parameters
-functional association with variables
-internal documentation
-redundancy checking

Model solving

In order to gain a sense of confidence in a model, its prediction ought to be consistent with data at hand, or its profile should at least demonstrate the features characteristic of the system for which it is being developed. Either way, the investigator needs access to model solutions and in this regard the following features are often available in modeling software:
-flexible and intuitive array of «solvers»
  ·explicit linear equations
  ·implicit linear systems
  ·explicit nonlinear equations
  ·implicit nonlinear systems
  ·ordinary differential systems
  ·stiff differential systems
  ·differential algebraic systems
  ·partial differential equations
-(automated) selection of appropriate solver
-(automated) solver parameter refinement
-capacity to handle discrete events
-retention and retrieval of annotated solutions

Experimental protocol

Much time and money can be wasted if inadequate preparation is undertaken prior to conducting experiments. One of the most powerful uses of modeling methodology is linked to the planning of experiments. Here through simulation analysis injection sites, observation points, sampling times etc. can be evaluated in regard to their capacity to expose some «unique» feature of a system. To help evaluate experimental protocol, modeling software supports the following tasks:
-setting initial conditions
-representing observations
-solving the system at sampling points or to steady state
-representing experimental perturbations
-automatically managing experimental/model object units
-graphically portraying experimental protocol
-simulating experimental error
-predicting data information levels
-calculation of experimental design points

Model fitting

Through its constructs a model may, for example, encapsulate metabolic pools, metabolic pathways, and reaction mechanisms as a reflection of our understanding of the science surrounding our domain of investigation. A key ingredient of the modeling exercise is in judging the degree to which data reinforces these ideas. Without compromising our scientific ideas, it is not unreasonable, under certain circumstances, that small (or

possibly even large) variations to the magnitude of model parameters may be found to achieve acceptable consistency between our model and the data. This step is known as model fitting and in conjunction with it modeling software supports the following:
-linear and nonlinear parameter fitting
-automated separation of linear and nonlinear adjustable parameters
-incorporation of known parameter information into fitting scheme
-accommodation of adjustment boundaries on parameters
-gradient and non-gradient search techniques
-include and exclude capability for adjustable parameters
-control over the objective function
-capacity to deal with under-identified systems
-clear association between model and data set for model fitting
-lossless interruption capability during model fitting
-automated fitting procedure parameter refinement
-control over parameter updating following model fitting
-automated selection of appropriate fitting algorithm
-graceful termination if fitting problems encountered
-intelligent reuse of calculated values to save time
-detailed reporting of fitting convergence information

Statistical analysis

In order to ascertain whether, within the bounds of experimental error, a treatment has caused certain significant quantitative shifts in aspects of a model we require statistical information in regard to the original model and the newly refined model. This type of concern often arises in the analysis of nutrition models and for it, and other statistical issues, modeling software offers the following to assist with statistical analysis:
-estimates of parameter values and their uncertainties
-statistical description of observations
-model prediction uncertainties
-propagated error analysis for dependent functions
-post-fitting residuals analysis:
  *Durbin-Watson determination and test
  *R^2, and R^2 determination
  *serial plots for residuals
  *outlier analysis
  *heteroscedasticity analysis
  *Kruskal-Wallis normality test
-robust estimates of parameter confidence intervals
-simultaneous analysis of data sets from independent experiments including parameter comparisons
-model selection analysis (Akaike Information Criterion; Akaike, 1973)
-leverage analysis (x and y variables; Hoaglin and Welsch, 1978)
-robust estimation support
-support for model (re)parameterization

Graphics and printing

Considerable time and expense associated with scientific analysis and reporting can be saved if (publication quality) outputs of critical aspects of an investigation can be obtained directly as the investigation advances. In recognition of this, we find the following array of utilities emerging in modeling software:
-model structure and model output
-publication-quality graphics and graphic printout
-control over display (print) features:
  *functions and observations displayed on a plot
  *form of display of each plot item
  *ordinate and coordinate scale and value ranges
  *ordinate and coordinate forms; linear, logarithmic, normal
  *plot item annotation
-preconfigured model fitting diagnostic plots:
  *normality, heteroscedasticity, leverage
-format to journal specifications
-extensive printer support
-(dynamic) linkage graphs and data to word processors, spreadsheets, etc.
-sensible defaulting of plot specifications

General features

Finally, if software, for any intended purpose, is to be adopted by a user group it needs to specifically target that group and commit itself (via its development team efforts) responsively in a variety of ways to that group. Successful

products usually address the following issues in this regard:
-user group
-on-line friendly user help
-quality documentation including:
  *reference manual
  *user guide
  *expert assistance sections
  *tutorials covering diversity of product use
-competitive pricing
-graded level of software use (user protection)
-user meetings and workshops
-serialized publications updating users on:
  *meetings, new features, new applications, program updates
  *changes in support structure, new platforms supported
-support for a wide variety of platforms
-robust, efficient, and reliable product
-company responsive to user requirements
-local representatives
-variety of means for support (phone, fax, e-mail, World Wide Web)

Evaluation of four model development environments

Evaluation environments

There is currently a wide selection of software available for simulation modeling. Table I lists the names of some software product and vendors. The simulation software reviewed here included STELLA II (version 3.06), Scientist (Windows version 2.0), SAAM (version 31), and ACSL/SimuSolv (version 11 and 3, respectively). Selection of the software was based on familiarity of the authors with these packages. The software was evaluated on a personal computer (P5-75) with 16 MB of memory (STELLA II), a Gateway 2000 P5-90 personal computer (32 MB of memory) (Scientist and SAAM), and a Sun SPARCstation 10 model 30 (128 MB of memory) using the Solaris operating system (ACSL and SimuSolv). The packages were evaluated by using a lactation model published

Table I. Names and addresses of some vendors of modeling software.

Name	Vendor	Address	phone/fax
ACSL and SimuSolv	Mitchell and Gauthier Associates, Inc	200 Baker Avenue Concord, MA 01742 USA	+1 508 369 5115 +1 508 369 0013
Powersim	MicroWorlds	47 third Street Suite 200 Cambridge, MA 02141 USA	+1 617 225 0025 +1 617 225 0028
SAAM	US National Intitutes of Health	Laboratory of Mathematical Biology BLDG 10, RM 6B13, NIH Bethesda, MD 20892, USA	+1 301 496 8914
Scientist	MicroMath Scientific Software	2469 E Fort Union Blvd Ste 200 PO Box 21550 Salt Lake City, UT 84121-0550, USA	+1 801 943 0290 +1 801 943 0299
SIMULINK	The Math Works, Inc	24 Prime Park Way Natick, MA 01760-1500 USA	+1 508 653 1415 +1 508 653 6284
SLAMSYSTEM and FACTOR/AIM	Pritsker Corporation	Suite 500 8910 Purdue Road Indianapolis, IN 4628-1170, USA	+1 317 879 1011 +1 317 471 6525
STELLA II	High Performance Systems, Inc	45, Lyme Road Hanover, NH 03755, USA	+1 603 643 9636 +1 603 643 9502
Vensim	Ventana Systems, Inc	149 Waverley Street Belmont MA, USA	+1 617 489 5249 +1 617 489 5316

by Neal and Thornley (1983). The model predicts milk production in a cow milked twice (or three times) daily or continuously (simulating calf drinking) over the entire 300 days lactation. Model parameters were those as listed in table II of the manuscript by Neal and Thornley. The model is based on hormone concentration (H), number of secretory cells in the mammary gland (C_s), quantity of milk in animal (M), and average quantity of milk in animal ($\bar{M}$). The model contains the following particularities:

-The model uses two integration step-sizes for machine milking. A short (0.001 d) integration step-size is used during milking and a longer one (0.1 d) between milkings. The model requires even a considerably smaller step-size during milking if solutions are to 1) precisely hit milking points, and 2) allow for both milk removal and gland refilling moderately smoothly during each milking episode. Furthermore, to realistically represent this situation required uncoupling integration and solution steps since it is not minute by minute based calculations we require but rather daily or weekly (see figures in Neal and Thornley's paper).

-To achieve a sigmoidicity in the affect of milk accumulation on death rate of secretary cells, the latter contains the term ($\bar{M}/M_h$) which is raised to the power 10 in the article. This may cause floating point problems in evaluation of this term for small M.

-The differential equations for differentiated secretory cells (C_S) and milk in animal (M) are highly nonlinear, particularly, as Neal and Thornley observe, for a milk removal constant (K_M) of about 5 kg.

-Neal and Thorley use case-sensitive naming of parameters (e.g., K_M for milk removal constant and k_M for milk secretion constant).

STELLA II

Technical data

STELLA II is an graphical-oriented modeling package developed by High Performance Systems Inc. (USA). Originally developed for the Macintosh, a Microsoft Windows compatible version has been available since 1994. The minimum configuration requires 4 Mb of RAM (8 Mb recommended), at least 5 Mb of free disk space, and Microsoft Windows 3.1 (Windows version) or System 6.0.4 (Macintosh version). The core package costs $715 but significant discount is available for universities and students. To support model distribution, an authoring module (for creation of specific user-interfaces) and run-time versions of STELLA are available.

Help and User support

The software comes with extensive documentation including manuals for getting started, technical aspects, and the methods and tools used in STELLA II. Although there is no on-line help, the input syntax of functions is displayed when entered incorrectly. High Performance Systems, through their foreign representatives, offer workshops for training. To promote exchange between users, a subscription to six-monthly published list of STELLA II users is available ($10 per year).

Functional supports for

Model specification/manipulation

The model specification in STELLA is based on two main levels of information, the *map layer* and the *model construction layer*. At the map layer, the general structure of the model is depicted in sectors. Each sector, which consists of a series of related model elements, can be individually run, documented and illustrated with text, pictures or movies (Macintosh version). Relations between sectors are indicated by bundled flows which are based on flows defined in the model construction layer.

The actual model specification is done in the model construction layer. State variables (*stocks*), rate variables (*flows*), and auxiliary variables and constants (*converters*) are represented as graphical building blocks. During model construction, building blocks are picked from an icon bar and placed in a diagram of the model construction layer (Fig 1). Each building block will carry a unique name (long names are possible but names are case-insensitive). A question mark appearing in the building block indicates that information is missing. Double-clicking on the building block will open it and will allow entry of specific information (e.g., initial values for stocks). Each building block can be individually documented. Functional relations between building blocks are performed by connecting these with an arrow (*connector*) after which the relation can be specified. Ghost images (i.e., copies) of building blocks can be used in the diagram

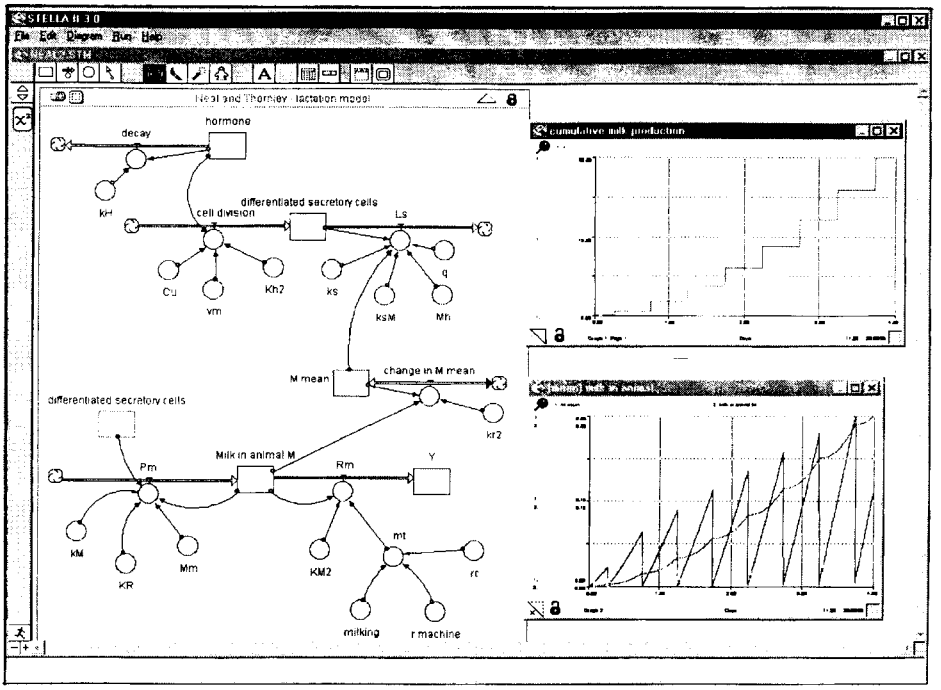

Figure 1. Model construction layer in STELLA II of the Neal and Thornley lactation model (left window), cumulative milk production (Y, upper right window) and (mean) quantity of milk in animal during the first four days of lactation (M and M; lower right window).

serving as anchors for connectors thereby avoiding lengthy connections between (spatial) distant elements. For complex models, the search utility is helpful in relocating model elements.

Fig 1 shows the model construction layer of the Neal model. Through its use of stocks and flows, the STELLA architecture ensures that no material is inadvertently lost. In the Neal model, the cumulative milk production is the cumulation of all the milk taken from the animal. The use of a flow ensures that the outflow from M (milk in animal) always equals the input in Y (cumulative milk production). Unit conversions between two stocks (e.g., from mole to g) can be indicated in a flow. The milking function was modeled using a combination of the STELLA modulo function and logical commands (i.e.: mt = if ((mod(time,1)>0.25 and mod(time,1)<0.255) or (mod(time,1)>0.75 and mod(time,1)<0.755)) then rm else 0).

Stocks can be operated as conveyors, queues or ovens to delay flow of material. A conveyor has a defined capacity and potential for leakage (in nutrition, the conveyor could represent the gastro-intestinal tract where leakage is absorption). Queues can be used in processes that require a first in - first out type of operation whereas an oven may be suitable for modeling batch processes. In addition to commonly found functions, STELLA II can report a series of statistics related to the time spent «in process» based on a time-stamp introduced at (the beginning of) a flow chain.

Complex models can be managed more easily by the creation of sub-models. Sub-models are represented at the model construction layer as special icons, but are actually comprised of a series of building blocks which can be displayed or hidden on request.

Apart from the built-in functions and constants, STELLA II also allows the use of

tabular data as model input (up to 1500 data points per series). Incorporation of data from external files can be performed using the Macintosh version, but not under the Windows version of the program (copy and paste of a series is supported). In addition, if no detailed data are available, a graphical relation of variables can be drawn and used as model input.

Model compilation
No specially compiled version of the model can be constructed. Model construction and model solving are done in the same environment.

Model solving
Three fixed step-size integration methods are available in STELLA (Euler (default), 2nd and 4th-order Runge-Kutta) and only one method can be used during model solving. Moreover, the integration step-size cannot be changed during solving (as required by the Neal model). The model will not run if simulation run is long compared to the chosen integration step size. Running the Neal model (300 d) with the short integration step-size (0.001 d) resulted in an error message which can be resolved by shortening the simulation run or increasing the step-size. The maximum simulation run for the short integration step-size was 32 days of lactation, which required 3 minutes and 45 seconds to solve on a P5-75 with 16 MB of memory. Models are run for a predefined time (a pause function is available) but cannot be terminated depending on model conditions.

Parameter values can be changed between runs, but cannot be saved separately from the model definition. Discrete events are generally handled correctly but care must be taken if they are scheduled at times that do not coincide with sampling points. For example, the modulo function described above will not be executed correctly if the integration step is greater than 0.05 (the equivalent STELLA **PULSE** command will work correctly but ignores the dynamics during milking).

Model fitting
STELLA II does not include a module for model fitting.

Sensitivity analysis
The sensitivity analysis capabilities of STELLA II are rather limited. The value of the parameter under study is changed (either to conform to an incremental series, a distribution, or user-specified) and its effect on a single model variable will be displayed graphically or tabularly. STELLA II will not calculate statistics related to the sensitivity analysis.

Graphics, printout, and links with other packages
STELLA II can display and print different types of information related to the model structure and model output. At the model specification level, the map layer (overview of the model) and model construction layer (detailed model structure) can be printed. In addition, all underlying equations, including the documentation of model elements are available as output. As a Windows (or Macintosh) program, program output can be printed on any device for which a driver is available. Graphical output can be defined as a time-series (up to five variables per graph) or scatter plot (one X-variable and one Y-variable). Previous plots can be conserved in order to compare output between simulation runs. Few options exist to modify the lay-out of the graph (e.g., color, line styles, fonts, supplemental information) and the standard graph is unlikely to be conform a journal's standards. Therefore, the user will have to export data to a third-party graphics package. STELLA II does support direct links to external packages for import or export of data but only in the Macintosh version.

Conclusion
STELLA II is highly intuitive package that will allow users to create a model within an hour of installation. It's main strength is the graphical interface (layering concept and building blocks) and printed documentation. However, drawbacks include lack of on-line help, limited integration methods, no model fitting capabilities, and lack of integration with other packages (Windows version). Although it may serve better as a tool for demonstration and education than for scientific model development, its graphical interface can be useful during the initial phases of model development.

Scientist

Technical data
Scientist is a Microsoft Windows program

which brings together the entire power and functionality of a suite of scientific software developed and refined for the last nine years at the MicroMath corporation. The target application of the package is, and has always been, the scientist's work bench.
Scientist is only available for the PC and runs under all Intel processor chips upward from the 80386/80387 combination. The software requires at least 4 Mb of RAM (though 8 Mb is recommended) and also requires at least 5 Mb of secondary storage.

Help and User support
The Scientist program comes with one instructive (500 pages) manual which combines reference material, tutorial guides, and «manual style» information. Windows format on-line help is available as well as context-based information in the software status bar. For particularly difficult problems, the user can contact MicroMath for direct assistance.

Functional supports for

Model specification/manipulation
The functionality of Scientist is accessed via a series of windows; a data or spreadsheet window, a plotting window, a model window, a worksheet window, and a general text window, and the functions available are automatically configured to match the needs of the current (uppermost) window. In the model window, models are created using a pseudo-mathematical syntax in which persistent identifiers, in various classes, need to be declared (see example below) ... temporary variables need not be declared.

Whilst at first entry into this window a model template is available, to facilitate initial model development, there is in fact no assignment of identifiers, by name or form, to any subsequent role (in contrast to SAAM's modeling syntax). Identifiers may be of any length but are not case-sensitive.

Scientist supports around sixty common scientific and programming-oriented functions making it possible to create quite sophisticated, and discontinuous models extremely easily.

The software has a number of innovative and helpful features to expedite investigations, e.g. it is capable of solving implicit nonlinear equations; there is considerable flexibility regarding parameter value, integration range, and other solution information entry; and there is virtually no ordinal significance connected with the information entry into the model window.

Many of the features of model specification using Scientist can be seen in our representation of the Neal model in Fig 2. We note that the 4 state variables, H, C_S, $\bar{M}$, and M, are identified as «dependent variables» and are defined in their respective state equations (denotes derivative). They are initialized as needed. The milking function uses a similar approach as for SAAM in which a periodic function is cropped at the appropriate level. We have set the integration range from t=0 to t=300 and our abstraction of «t» units is day.

Model compilation
Model compilation can be independently invoked (from most windows) and serves largely to verify model syntax and to isolate linear parameters. A compiled model cannot be used separately from the package

Model solving
Scientist supports six integrators, two fixed step-size (Euler and Runge-Kutta) and four variable step-size (Bullirsch-Stoer, error-controlled Runge-Kutta, Adams, and Episode's stiff integrator). Selection of an integrator automatically exposes the appropriate integrator parameter set for which assignment may be made ... though each procedure is sensibly defaulted and restoration of defaults is quite simple. Some dimming errors and apparently conflicting assignment options obscured initial attempts to invoke fixed step integration.

To solve the Neal milk production equations, the fixed step Runge-Kutta integrator was used with a step size of 0.001 d. Approximately 19 min of processing time was required for the 300 day lactation solution, and in Fig 2 we present plots of Neal's «Milk in Animal» and «Average Milk in Animal» functions.

Model fitting
Scientist has comprehensive weaponry to facilitate model fitting and includes versions of Steepest Descent, Levenberg-Marquardt, Powell, and Simplex minimization algorithms. In addition, and in conjunction with model fitting, Scientist can automatically identify linear parameters and it takes advantage of this as it attempts nonlinear estimation.

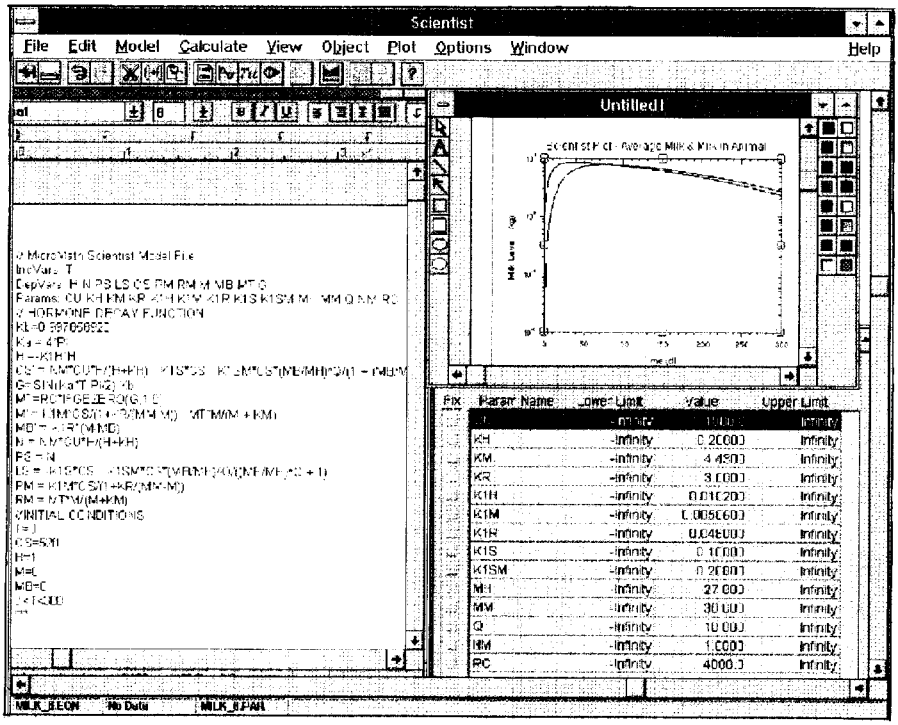

Figure 2. Listing of the Scientist model definition (left window), parameter values (lower right window), and a plot of the quantity of (mean) milk in animal ($\bar{M}$ and M, respectively; upper right window).

Following model fitting, Scientist is able to calculate many critical statistical details concerning the fitting process. Indeed its capacity to provide residuals analysis together with optimal data transformation for data/residual homogeneity is quite unique.

Sensitivity analysis

Although not explicitly supported, the mathematical/numerical tools available within Scientist, including its capacity to differentiate and integrate functions provide most of the computational machinery necessary to perform sensitivity analysis.

Graphics, printout, and links with other packages

Scientist has a very handsome and quite manageable graphics window permitting, for example, displays in 2D rectangular, 2D polar, 3D surface, 3D curve, and 3D bar modes. Although, in our estimation, a little too sticky, plot templates can be easily created, saved, and re-invoked to avoid rebuilding complex plot descriptors each time re-plotting is needed. Graphs can be output to meet a wide assortment of drivers.

With the addition of an equation editor, the capacity of Scientist to manage and output text and related information is such that the user need never leave a (modeling) session between initial data entry and putting the final touches on a paper ... oh if life were so simple.

In recognition that the Scientist user may perhaps be more comfortable with data entry/manipulation using one of the variety of spreadsheets and database systems currently around, the developers have thoughtfully

incorporated translation linkages between their spreadsheet and at least six other popular ones.

Conclusion

Scientist is a creative program which equips the serious investigator with all the tools needed to rapidly and effectively advance data analysis and model fitting. The Neal and Thornley model was quickly formulated in Scientist's modeling language and solved with the required precision for the entire lactation cycle.

Some minor shortcomings of Scientist were its lack of case sensitivity, its lack of integration across certain of its operational windows, and its somewhat excessive processing time (approximately 19 min) to solve the Neal model on a Gateway 2000 P5-90 computer.

SAAM

Technical data

SAAM is a kinetic modeling program suite that has been developed and refined for over thirty years. It has been used in conjunction with the analysis and publication of many thousands of kinetic investigations in areas embracing digestion, metabolism, pharmacokinetics, and radiation dosimetry.

The SAAM software is offered in two versions, a batch version (SAAM 31) and a command-line oriented interactive version (CONSAM). They are developed, and distributed for Microsoft Windows. To run the programs requires approximately 5 Mb of memory, a 80386/80387 or better Intel processor and at least 5 Mb of secondary storage. The programs are distributed freely on two disks from the Laboratory of Mathematical Biology at the US National Institutes of Health.

Help and User support

There are three manuals that accompany SAAM and CONSAM: The SAAM Manual, The CONSAM User's Manual, and The CONSAM User's Guide. Two of the manuals come on the distribution disks. In addition, there is available a well documented instructional tutorial set embracing many features of the SAAM vocabulary and the CONSAM instruction set. Furthermore, interactive help is available while using CONSAM to assist with syntactic and semantic problems.

SAAM user meetings are held up to three times a year and are either woven into the agenda of scientific meetings or held under industry auspices for select participation and specific problem issues. Visitors are welcome at NIH or at the University of Pennsylvania where they may receive personalized guidance in SAAM and CONSAM use.

Functional supports for

Model specification/manipulation

CONSAM incorporates a command line interactive shell through which models are created, manipulated and explored. In essence, the CONSAM commands can be classified as editing commands, processing commands, and i/o commands. Using the editing commands, the user creates and modifies models in the SAAM syntax. Here a set of pre-defined modeling constructs and operational units with their specific functional underpinnings are lexically manipulated by the user to enable the rapid creation of quite complex (differential) models.

The modeling constructs available include fractional turnover parameters (L(I,J)): which define the existence of compartments, the connections between compartments, and the nature and rate of material exchange between compartments; delay parameters (DT(J), DN(J)): which define transit delays for the transport of material between compartments; user-definable linear and nonlinear parameters and functions (K(J), S(I,J), P(J), and G(J), respectively. Users exploring linear systems need only specify L parameters ... the SAAM interpretative machinery automatically creates the differential system implied. Indeed it is 1) its terseness, and 2) its abstraction of routine mathematics that has led to the popularity of this software.

The operational unit set extend the user's flexibility in two distinct directions. Firstly, and most obviously, it allows the user to emulate experiments on the system represented by the model. Here using initial conditions, input functions, and sampling switches simulated consequences of assaults to the system can be explored. Secondly, using forcing functions, and other (exotic) tools, we can manipulate models to synthesize responses possibly characterizing data originating from quite diverse situations.

The listing of the SAAM representation of

```
C
Compt. 1 = Hormone level,     Compt. 2 = Secretory cells number
Compt. 3 = Milk in animal,    Compt. 4 = Average Milk in animal
C
L(0,1)=P(4)
L(0,2) 1                                              *G 2
L(0,3)=P(15)                                          8G 3
L(0,4)=P(7)
H DAT
C G(1) = Rate of production of secretory cells
C G(2) = Milk induced loss of secretory cells
C G(3) = Modified milking function
C G(4) = Milk secretion rate
C
X G(1)=P(13)*P(1)*F(1)/(P(2)+F(1))
X G(5)=(F(4)/P(10))**P(12)
X G(2)=P(8)+P(9)*G(5)/(1+G(5))
X G(4)=P(5)*F(2)*(P(11)-F(3))/(P(11)-F(3)+P(6))
X G(3)=(SIN(P(20)*T-P(21))-P(22))/(P(3)+F(3))
X UF(2)=G(1)
X UF(3)=G(4)
X UF(4)=P(7)*F(3)
```

Figure 3. Listing of the SAAM representation of the Neal Model.

the Neal model (fig 3) highlights a variety of modeling constructs and operational units.

There are 4 compartments (1 to 4) or state variables and 2 of these have linear state equations (H and $\bar{M}$) and 2 have nonlinear state equations (C_S and M). The nonlinear equations have been built, in part, using function dependent transfer modifiers, and, in part, using nonlinear input rates (UF(2) and UF(3)). The «8» on the L(0,3) parameter definition line causes the parameter to take non-zero values only when the modifier, G(3), is greater than zero ... indeed it is through this very switching process that we obtain the Neal milking machine demand function.

The flexibility of the software is achieved from this capacity to very easily build and modify quite realistic nonlinear and discontinuous mechanisms.

Model compilation

The second set of CONSAM commands, the model processing commands, comprise signals to the SAAM processing machinery to advance the state of the model. For instance the DECK processing command compiles the model (builds equations and encodes data), performs some syntactic analysis, and, if possible solves the dependency equations. Whilst the compiled version of a model can be saved, the software is still required to continue to use it.

Model solving

After compiling, the model its state equations can be solved for the nominated solution points using CONSAM's processing command SOLVE. Whilst SAAM will automatically select amongst an array of, implicit, explicit, linear, nonlinear, and quasi-symbolic integrators, one which is best suited to the problem, the user can override this choice. Specific integrators available include Runge-Kutta 4, Runge-Kutta 4/5, Newton-Raphson exponential solver, Convolution solver, Gear, and Adams-Bashforth. Tuned parameter values for the respective integrators are refined by SAAM in conjunction with the problem details ... again though these can be altered by the user.

Using the above model we attempted to solve the milking machine demand model, for a twice day milking program, over the entire lactation (300 days) cycle. Unfortunately, because this system required fixed step integration, with step size around 0.001 d and because SAAM neither supports a nonlinear

fixed step integrator nor accommodates more than 1000 solution points we were unable to solve the system for more than 2 days.

Model fitting
After solving the CONSAM user can invoke function fitting using SAAM's highly refined nonlinear least squares algorithm. Some features of this facility includes:
-separation of linear and nonlinear estimation steps
-generalized least squares objective function
-Marquardt style modification to the Gauss-Newton algorithm
-known parameter variances admitted into the fitting process
-boundaries on parameter adjustment ranges are honored
-step-size scaling and convergence criteria automatically managed by software
-singular value analysis invoked when non-uniqueness would otherwise cause inversion errors
-incorporation of steady state data and steady state parameters into the model fitting scheme

At the conclusion of model fitting a critical set of convergence guides is displayed.

Sensitivity analysis
Following model solving and model fitting the third class of CONSAM commands, the i/o commands are usually invoked, for example, to help the investigator to assess the model adequacy. Here two forms of sensitivity are available 1) model sensitivity; and 2) data sensitivity. The model sensitivity, a scaled version of the partial matrix (each column representing a particular adjustable parameter) portrays the temporal sensitivity of the model to a parameter around its current value.

The data sensitivity, in essence the variance of a datum divided by the variance of the entire data, reflects the relative role of each datum in conjunction with model fitting. Using this latter information measure investigators may be assisted in locating observation points to assure optimal model identifiability.

Graphics, printout, and links with other packages
Using CONSAM's i/o commands, most of the model solution results generated by SAAM can be displayed graphically, displayed lexicographically, or sent to graphics (HPGL) or ASCII text files. All the user's interaction with CONSAM can be logged to a file though currently the inputs and outputs are logged to separate files. CONSAM lacks the capacity to communicate with other software.

Conclusion
CONSAM and its processing kernel SAAM comprise an extremely efficient and highly integrated kinetic modeling environment. It is very fast and, from the model formulation perspective, quite amenable for the investigation of models such as that of Neal.
The greatest shortcomings of CONSAM relate to 1) its 'older style' architecture .... being command-line oriented makes its difficult to use; 2) the fact that it is not able to communicate through or with other packages; and 3) its limitation of being only able to handle systems involving no more than 75 state equations, somewhat limit its utility.
This investigation has demonstrated the clear need within SAAM of at least one integrator for which the integration and solution steps can be independently set. In addition, the importance of a more flexible method for naming objects became apparent.

ACSL/SimuSolv

Technical data
ACSL (pronounced axel) is an acronym for Advanced Continuous Simulation Language and has been developed by Mitchell and Gauthier Associates Inc (USA). SimuSolv, which uses the ACSL simulation language, has been developed since 1983 by The Dow Chemical Company. Its major purpose is to solve optimization problems or to estimate model parameters given an ACSL model and experimental data. Both packages are marketed in Europe by Rapid Data Ltd (UK). Since January 1996, the DOW Chemical Company has stopped development of SimuSolv for use outside the company. Mitchell and Gauthier Associates Inc have announced a new module for ACSL (ACSL /Optimization) that will have at least the same functionality as SimuSolv.

ACSL is available for a variety of mini-computers and mainframes (multi-user license) as well as for PC and Macintosh (single-user license). In contrast, SimuSolv is available only for mini-computers and mainframes. Both ACSL and SimuSolv require a FORTRAN compiler (not included). A single-user ACSL

license costs $1500 (educational price), whereas a multi-user license costs $3500 for the first year plus a $1000 annual license fee (educational price). A multi-user SimuSolv educational license is $2000 for the first year, and $500 thereafter.

Help and User support

The ACSL documentation consists of a single technical manual and several brochures for installation and testing. Technical documentation containing reported problems and suggested solutions are distributed on a quarterly basis. The SimuSolv documentation, which covers both ACSL as well as specific SimuSolv commands, is more extensive and includes many examples. A separate *Introductory Guide* discusses the main features of SimuSolv. Both Mitchell and Gauthier Associates Inc (ACSL) and The Dow Chemical Company (SimuSolv) publish free newsletters. In addition, training sessions are being organized on a regular basis. Developers and their European representative can be reached by e-mail.

Functional supports for

Model specification/manipulation

An ACSL model consists of two parts: the *model definition file* and the *run-time commands*. The model definition involves establishing mathematical equations for the system to be modeled. Model conditions are executed through run-time commands or a runtime command file. The advantage of this design is that the internal model structure can be kept constant, whereas the model conditions (e.g., the values of model parameters) can be varied for different simulations.

The model definition (an ASCII file) of an ACSL model consists of three main sections: INITIAL, DYNAMIC, and TERMINAL. The INITIAL-section is executed only once at the beginning of a simulation run. In contrast, statements in the DYNAMIC-section are executed numerous times during the simulation run. It may contain differential equations that describe the evolution of state variables, as well as discrete events (i.e., events that only occur at a specified time or under specified conditions). For this purpose, DERIVATIVE-blocks and DISCRETE-blocks can be defined within the DYNAMIC-section. Finally, a TERMINAL section may be defined for those events that need to be calculated only at the end of a simulation run.

ACSL makes use of a large library of internal functions and operators. In addition, user-defined functions based on FORTRAN can be included in the models. Constants may be defined as scalars or multi-dimensional arrays. Interpolation between external data points (up to 10000) is possible through linearization or smoothing methods. Missing data are permitted and questionable data can be marked and will be ignored in statistical analyses.

ACSL and SimuSolv allow the use of macros and FORTRAN procedures, which can be used to create forms for data entry and requests for data analysis. Being a FORTRAN-based program, ACSL and SimuSolv impose some restrictions on the format of the model definition file. Names of variables and constants cannot exceed six characters in length (31 in ACSL), and a maximum of 72 characters per line can be used. Model documentation can be included in the model definition as quoted text and is ignored during compilation. The model definition file in ACSL is case-insensitive; as a result the naming in the following ACSL model definition file deviates somewhat from the naming convention used by Neal and Thornley (Fig 4).

The DYNAMIC section contains one DERIVATIVE-section (which contains the differential equations and integration instructions) and two DISCRETE-sections (to turn machine milking on and off and to change the integration step-size). The model run can be terminated (TERMT) based on the value of the run-time variable (TIME) or any other model condition. Parameter values are declared in the model definition file, but can be changed on the command line.

Model compilation

The ACSL model definition and command files are ASCII files and can therefore be used under any operating system. In order to be run, the model definition file needs to be compiled under FORTRAN. Once compiled, run-time commands can be entered on-line or read from the command file. Any change in the model structure (i.e., the model definition file) requires re-compilation. The ACSL software does not impose a limit on the program size or number of state variables.

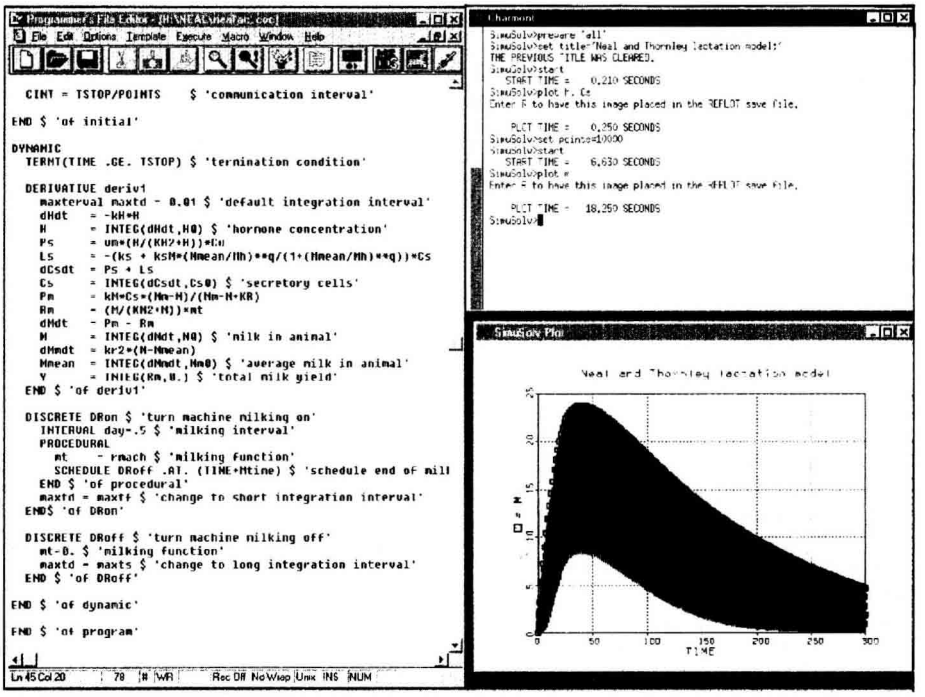

Figure 4. Partial listing of the ACSL/SimuSolv model definition file (left window), the command line window (upper right window), and a plot of the quantity of milk in animal, using a solution interval of 1.44 h (M; lower right window).

on line-printers or plotted on various output devices (Tektronix, HP, postscript, X-windows, Sunview, Microsoft Windows compatible devices). To assess the effect of changing model conditions (e.g., changing parameter values), outputs from successive simulation runs can be displayed in a single graph. Parameter estimation behaviour (e.g., confidence regions) can be studied by three-dimensional plotting of the objective function versus two parameters.

Recently, a graphical user-interface for ACSL (Graphic Modeller) has been developed which allows users to visually program, execute, and analyze simulation models rather than using a text-based program and command-line execution. ACSL vision is a product that allows animations to be included with an ACSL model whereas ACSLrt is a module for real-time simulation. Also other, external products can be linked with ACSL (e.g., SimuSolv, MATLAB). The Microsoft windows version makes use of DDE, allowing exchange of data between ACSL and, for example, graphics packages and spreadsheets.

Conclusion

The combination of ACSL and SimuSolv comprise a very powerful tool for simulation modeling. The use of a program definition file in ASCII ensures portability between different operating systems and platforms. Depending of the computational power required, a modeller can choose platforms ranging from personal computers to supercomputers. A disadvantage is the fact that, compared to a product like STELLA, ACSL requires considerable investments, both in time and finance. Larger models may require considerable thoughts on

Model solving

ACSL and SimuSolv not only provide several integration algorithms (Euler, Runge-Kutta 2 and 4, Runge-Kutta-Fehlberg 3 and 5, Adams-Moulton, and Gears's BDF) but also allows users to write their own integration algorithms in FORTRAN. More than one integration method can be used in the model and the integration step-size can be changed during the simulation run (as required by the Neal model). The definition of DISCRETE sections ensures that scheduled events will coincide with solution points. On a SPARCstation 10, the system for a 300-day lactation period with machine milking was solved in 0.25 seconds (0.7 seconds on a P5-133). To solve the system with a single integration step-size of 0.001 day required 7 seconds.

The use of discontinuities such as machine milking lead to specific characteristics in model output. For example, the removal of milk from the animal (daily milk yield; $R_M(t)$) is not a continuous function as suggested in fig 6 by Neal and Thornley, but is a series of (quasi)-spikes that coincide with the milking interval. In fact, all state variables but the hormone concentration will be affected by these discontinuous and are therefore discontinuous functions (although some state variables are more affected than others). The false appearance of continuity is due only to the selection of the solution points. If one solution point per day is chosen, we will obtain the dynamics of a state variable only at that point in time. If this solution point happens to be within a milking interval, we will obtain dynamics of a state variable at a different level then when the solution point is just before milking. For example, in Fig 4 we plotted the quantity of milk in animal (M) displaying solutions every 1.44 h. If we would have used one solution point per day, we could have obtained a false continuity of M with a maximum of 8 kg or 24 kg only depending on the selection of the solution point within the day (i.e., during milking or just before milking).

Model fitting

SimuSolv employs the method of maximum likelihood in order to obtain parameter estimates. It allows estimation of parameters given experimental data for one or more dependent variables. For example, in the model described before, data may have been collected for total cumulative milk yield (Y). The SimuSolv commands to find the best set of estimates for k_M (milk secretion constant; KM2 in ACSL model definition file) and K_R (milk secretion rate) are:

```
VARY KM2, KR
FIT Y
OPTIMIZE
```

As with any non-linear estimation procedure, reasonable initial values for the parameters are required in order for the algorithm to converge. Parameter bounds can be used to restrict the values of the estimators. A general error-model can be used in SimuSolv to account for non-homogeneous variances (heteroscedasticity) within a data set. A typical SimuSolv listing includes parameter estimates with standard deviations, observed and predicted values for all dependent variables, initial and final values of the maximized log-likelihood function (overall and per dependent variable), weighted sum of squares, percentage of variation explained, the heteroscedasticity parameter, the parameter correlation matrix, and the variance-covariance matrix.

The OPTIMIZE command in the previous example cannot only be used to maximize the log-likelihood function but also to optimize any other objective function (e.g., maximization of a state variable). Hypothesis testing cannot be performed directly with SimuSolv although the manual contains excellent (illustrated) instructions to use the SimuSolv output listing for comparison of both nested and non-nested models.

Sensitivity analysis

Dependencies between the model responses and input parameters are studied by calculating sensitivity coefficients (∂(model responses)/∂(model parameters)). To reduce computational effort, calculation of sensitivity coefficients is decoupled from solving the differential equations. Normalized (to eliminate bias due to differences in magnitude of parameter values) sensitivity coefficients are displayed graphically as a function of the run-time value allowing the study of both the magnitude and the dynamics of model sensitivity.

Graphics, printout, and links with other packages

ACSL and SimuSolv provide a extensive set of tools for producing output, which can be listed

model description, not so much for the ACSL language interpreter, but more to maintain an overview of the model. SimuSolv integrates well with ACSL language, but is based on a slightly older version and is not available for personal computers. In contrast to ACSL, no graphical interface is available for SimuSolv.

Conclusion

In this review we have seen a series of different modelings environments that can be used for nutritional modeling. The environments differ from easy-to-use graphical environments such as STELLA II to text-based programs such as ACSL and SAAM. We have listed requirements for efficient model development; none of the packages reviewed here fulfilled all of the requirements.

There are several important issues to address before purchasing a specific simulation package. The most important issue for a software product is that it should fit the user's needs. Although this may seem obvious, it is clear that «full-time» modelers have different demands (functionality, power) than those who create models on a Friday afternoon (ease-of-use). Because user needs may change over time, it is desirable that the product evolves with the user, either through the use of a «light» version of an extensive product, or through offering of different levels of access at software functionality. Given the variety of users, it is important that software vendors pay considerable attention to the documentation of the product. This should include not only a user's guide but also technical background information for experienced users as well as for the non-mathematically inclined.

An efficient simulation product on the one hand should integrate well within the operating system (e.g., provide links with other packages) and on the other hand should be available on a variety of platforms (i.e., one should be able to change platform if additional computational power is required). These horizontal (within operating systems) and vertical (between operating systems) compatibility issues have to be addressed by software developers.

The Neal and Thornley model illustrated the large differences in computational power of simulation products. Solving the model for a full lactation with an integration step-size of 0.001 day would require more than 30 minutes for STELLA compared to 7 seconds for ACSL (using different platforms). Although the Neal and Thornley model might be rather specific (requiring 300 000 evaluations for each state variable) it is rather simple in structure. However, numerous more complicated models exist that require considerable computational power. For efficient model development, the human mind should be the limiting factor and not computational power of a software product.

Literature cited

Akaike H (1973) A new look at statistical model identification. *EEE Trans Automat Contr* 19, 716-723

Hoaglin DC, Welsch RE (1978) The hat matrix in regression and ANOVA. *Amer Stat* 32, 17-22

Neal HDStC, Thornley JHM (1983) The lactation curve in cattle: a mathematical model of the mammary gland. *J Agric Sci Camb* 101, 389-400