Presented at Symposium on Computer
Graphics in Medicine, Pittsburgh, Pa.
March 7-10, 1972

SYMBOLIC COMPUTER GRAPHICS AND BIOLOGICAL MODELS

Mark W. Horovitz, Donald M. Austin and Harvard H. Holmes

August 1972

AEC Contract No. W-7405-eng-48

DISCLAIMER

SYMBOLIC COMPUTER GRAPHICS AND BIOLOGICAL MODELS*

Mark W. Horovitz, Donald M. Austin
and Harvard H. Holmes
Lawrence Berkeley Laboratory, University of California
Berkeley, California 94720

In the first part of my talk I will describe the operation of a symbolic graphics computer system; next, I will show how some models can be implemented by using this system, and finally I want to discuss techniques for establishing the validity of models.

The Computer Graphics System

The interactive computer graphics modeling program which I use is called PICASSO and has been described in detail elsewhere;[1] I wish to discuss it from the biologist user's point of view and will not emphasize the computing aspects.

We begin the process of graphical modeling by creating and defining symbols. We then connect these symbols into a diagrammatic model. The PICASSO program can translate the diagram into a program. The program can then be compiled and executed to produce numbers and graphs. We can compare these simulation results with experimental data and use any discrepancies to guide us in improving the model. We can then immediately return to the graphic diagram, modify it, and thus start another cycle of modeling.

The system permits a biologist or physician, who is a computer novice, to construct a model in his first session at the graphics console, thus altogether bypassing conventional programming languages. We hope this will make computer modeling more accessible to this group of users.

Graphical Structures

The building blocks of our graphic structures are called "elements." Each element has a name, a symbol, and a definition. A symbol may have labels. An element which is not composed of other elements is called a primitive element and has a text definition. The text should be a set of statements which have meaning in a programming language such as FORTRAN or MIMIC.[2] It is possible to define an element in terms of a graph—i.e., a table of X, Y value pairs.

After some primitive elements have been defined, a more complicated element can be constructed by combining the symbols for these primitives into a single new symbol, the definition of which refers back to the text definition of its components. This is called a MACRO definition. The power of MACRO definitions stems from the ability to generate a hierarchy of such MACRO definitions. Groups of primitives can be combined into first-level MACRO's, then these can be combined to form second level structures and so on up to 128 levels deep.

User Interaction

The equipment that we presently use includes a CDC Type 252 display console. It has a 19" diameter CRT, a light pen, a teletype, and a keyboard with 22 function keys. The user brings the PICASSO program into execution, using a teletype to communicate with our PTSS time-sharing system running on a CDC 6600 computer.

The illustrations do not show the complete CRT screen. At one side a menu of commands or operations is displayed; at the top of the screen, space for 5 comments is provided. Both the commands and comments are omitted from the illustrations. Typically the comments serve the following functions:

Comment 1. Specifies the name of the element being edited.
Comment 2. Specifies the next action to be taken by the user.
Comment 3. Varies greatly in function.
Comment 4. Gives instructions on how to exit from the editor currently in use, in order to return to a higher level program.
Comment 5. Gives the name of the currently operating program or editor.

Initially, the user is presented with a choice from a menu of commands which include the following:

EDIT SYMBOL
EDIT OLD DEFINITION
NEW MACRO
NEW TEXT
ERASE NAME
LOAD LIBRARY
APPEND LIBRARY
SAVE LIBRARY
ERASE LIBRARY
VIEW
ANALYZE
FINISHED

Most commands start by displaying a list; for instance, if we choose to edit a symbol we will see a list of existing symbols. We initially select the "LOAD LIBRARY" command. The program will then display a list of the names of existing symbol libraries. The user then selects one of these for loading into core memory from the data-cell storage device. The user might next choose to edit a symbol. He will then be shown a list of existing symbols from which he will select one. The picture editor is then called into execution. The picture editor permits the user to create new symbols by drawing line segments, erasing all or part of existing symbols, attaching labels, changing the display magnification, taking microfilm photos, etc. One can create or erase symbols or text, combine symbols into new structures, and so on.

Once a satisfactory diagrammatic model has been constructed, we enter the analysis phase. One program carries out a topological analysis of the diagram and then a second program is called. If our text definitions have been in FORTRAN for example, then at this stage a valid FORTRAN program is produced by constructing subroutine calls for the symbols, carrying out subroutine parameter substitutions, and similar tasks. One can then compile and execute the constructed program, produce graphs, etc. Since the user writes the text definitions, PICASSO can generate code for a wide variety of programming languages or programs. For each such language, a short template must be constructed which defines, for this language, the equivalent of FORTRAN subroutine calls, program header and end statements, and a few other quantities. At the moment PICASSO can output FORTRAN and MIMIC code and also generate input for the circuit analysis program called CORNAP (the Cornell University circuit analysis package). MIMIC is a simulation language which simulates the operation of an analog computer. It is suitable for continuous systems simulation. We use our own interactive version of the Control Data MIMIC processor. (2)

Computer System

PICASSO runs on a CDC 6600 computer under our own BKY operating system, which permits multiprogramming with up to 64 jobs simultaneously in execution. The display is a CDC type 250 system consisting of an 8K word buffer memory controller driving five 19" refreshable display scopes and a microfilm recorder.

Besides a good complement of discs and magnetic tape drives, two other mass storage devices are in the system. One is an IBM 2321 data cell with 50-100 million words of storage—this is used for program and some data storage; the graphical libraries are stored there permanently. We also have an IBM 1360 photo digital store with 5×10^9 words of slower access permanent storage.

Construction of a Model

I will use compartmental models to illustrate our procedures, since these models are conceptually very simple. To help understand the essential ideas of compartmental models, try to imagine a system of water tanks interconnected by pipes. In each pipe there is a pump, pumping water at a constant rate between the tanks. The water flow rates are adjusted so that the tanks are in a steady state—this means that the water level in each tank is constant with time. This is called a steady state compartmental system; the tanks are called compartments. Now we introduce some tracer—such as a colored dye—into one of the compartments. Each tank has stirrers which keep the water and dye in that tank thoroughly and uniformly mixed. The dye will be pumped from one compartment to another. Our problem is to compute the amount of tracer in each compartment as a function of time.

Figure 1 shows a symbol for a compartment. The element is named COMPRT. It has 8 connections and 2 internal variables, X and XZERO. $X_i(t)$ represents the amount of tracer in compartment i at time t, and XZERO represents its initial value. In Fig. 2 we see another element, called a channel, which represents a channel of flow between compartments; it has a parameter labeled A.

In Fig. 3 we see a two-compartment model with a single flow channel connection. The simulation run will go from zero time to time = TFIN. The symbol at the top of the diagram denotes that the program will request the user to supply parameter values for XZ1, R1, and XZ2 via the teletype. The "scope-like" symbol on the lower right side specifies that CRT graphs of X_1 and X_2 versus time are to be generated. Let $X_1(t)$ represent the amount of tracer in compartment 1 at time t. Then $dx_1/dt = -R_1X_1$ and $X_1(t) = XZ1*\exp(-R1*t)$. In the general n compartment case we have a set of simultaneous first-order linear ordinary differential equations, the solutions of which are sums of exponential terms. Figures 4 and 5 display the MIMIC language program automatically generated from the simple model shown in Fig.3. Figure 6 shows an 8-compartment model representing short-term plutonium tracer kinetics in the rat. The compartments are labeled bone, liver, etc. Plots are specified for the PFREE and BONE compartments. Some channel flow parameters have fixed values, others are to be varied with each simulation run. Figure 7 shows the percentage of total tracer in the BONE and PRFEE compartments as a function of time, for one particular set of parameter values. I think that the actual behavior of this model is not pertinent to this discussion. The results and conclusions of our plutonium compartmental model have been published. (3)

Evaluation of the System

How easy is it to construct models by using the PICASSO program? Skill is required to choose and define the primitive elements so that they yield neat, natural building blocks for a class of models. Once the primitives have been defined, model structures can easily be built. The library storage facilities for graphic models are very convenient. To illustrate some of the features of the system, let us suppose that we want to give a user an introduction to compartmental models. We can take a one-compartment model out of the library, analyze it—examine the equations produced in the analysis phase, execute a simulation run—examine the results, change the parameters, and run it again. Then we can pick a two-compartment model and go through the same cycle. Next we could build a model of real interest to the user or look at more complete stored models—all in one session at the console. Starting out in this way, the new user does not have to spend a great deal of time learning about the system before being able to tackle problems of interest to him.

Extensions

It is easy to accommodate analysis of other languages, and we expect this aspect to proliferate. It seems to me that this should be encouraged, provided a processor for the language is available on our machine. If one is aiming for ease of use, then a new PICASSO user with some past experience of modeling, using for example GPSS or DYNAMO, should be encouraged to continue by generating PICASSO models which are executed via GPSS or DYNAMO.

In the present system, if I wish to construct a model with the same functions and analyze it via either MIMIC or FORTRAN, I have to generate two sets of definitions and names. In other words, I may have one visual representation for the model, but I need two sets of names. Example: for an adder I could have ADDERF with a FORTRAN definition, and ADDERM with a MIMIC definition. A later version of PICASSO will permit multiple definition of symbols. It is also hoped to add features so that animation of diagrams will be made easy.

The Process of Model Building

I now will outline the stages one usually has to pass through in the process of building a model.

Assume we start with some experimental data relating two or more variables. Often we start by describing this relationship by some appropriate mathematical function. We might choose a power series, Fourier series, or a sum of exponential terms. Having chosen an appropriate set of functions we might then try to find those parameter values of the functions that best fit the data in the least-squares sense. For example we might fit our data with one exponential term, then successively try fitting 2, 3, and 4 terms. This process will be expedited by the availability of interactive computing facilities. Existing programs, such as MODELAIDE,[4] MLAB, and others[5] are suitable for this kind of operation. Such preliminary curve-fiting may suggest a suitable category of models; for example, if sums of exponentials fit the data well this might suggest a compartmental model.

The next stage would then be to construct and test such a model. We can use PICASSO for this simulation phase. Such a simulation permits us to explore the behavior of the model for different sets of parameter values and initial conditions. If we are lucky this process will lead to a model that fits the data well. The final phase of the modeling process should be to find model parameters that give a least-squares solution. For this we need a program that will find the minimum of a function of many variables. Programs such as MODELAIDE include such facilities. In interactive computer modeling I think one needs to be able to go back and forth easily between a fitting mode and a simulation mode of operation such as PICASSO provides. We intend to extend our system to permit this.

Models with many components are plagued by the possibility of many different solutions that fit the data about equally well. With PICASSO it is quite easy to generate a complex model. To keep things in balance, we should now provide improved methods of testing models. Breaking a large model up into sub-models, each of which is then subjected to a least-squares fitting process, is one necessary approach.

There are other techniques which are applicable to problems of uniqueness and equivalence of models. For example, consider two models, each with the same five compartments but with different connections. Each of these may be equally compatible with the available, incomplete information. Mapping procedures exist which permit one to transform one of these models into the other. Such procedures have been discussed and implemented by Berman[8, 9] and others and should be conveniently available in computer modeling systems.

A related technique, which can be applied to any linear dynamic system, is the theory of flow graphs. It is useful in dealing with a complex network of flows where not all the nodes are accessible to measurement. From the model's topology we construct a flow graph, and from the system equations we find transfer functions between nodes. From such a graph, we can obtain an equivalent flow graph of minimal complexity, by following a straightforward set of rules for graph reduction.[6, 7] This minimally complex graph, called the essential graph of the system, is unique and that is why it is interesting for this kind of work. In our computer modeling system I plan to provide facilities for automatic graph reduction.

Your might be sceptical about the utility of this approach, since large complex models are usually nonlinear systems. However, one can often construct linearized approximations that are good over a small range of the variables. For example, tracer experiments on chemical or cellular systems often satisfy these conditions. If one has data for several different such linearized regions, then one can expect the paths of flow (called channels in my PICASSO examples) in the linearized model, to correspond to those in the original nonlinear model, even though the dynamical behavior of the two models may be quite different.

Of course the imbalance between the effort required to construct the model versus the work required to establish its validity already existed before graphical modeling programs like PICASSO came along. Simulation languages such as MIMIC or DYNAMO make it fairly easy to construct dynamic models. PICASSO accentuates this imbalance and makes it more urgent to integrate tools for model testing, mapping, and simplification into computer-aided modeling systems.

In this discussion I have described the work of many of my colleagues--my contribution is really quite small. Don Austin and Harvard Holmes constructed the PICASSO program; William Benson and Andy Tannenbaum wrote the time-sharing system; many of my ideas on modeling came from Nooney,[10] Berman,[8, 9] and Rescigno.[6, 7]

Footnote and References

*Work done under the auspices of the U. S. Atomic Energy Commission.

1. Donald M. Austin and Harvard H. Holmes, PICASSO, A general interactive graphics modelling program, Lawrence Berkeley Laboratory Report LBL-580, Jan. 1972.
2. CDC MIMIC, Control Data Corporation Publication #44610400, April 1968.
3. Patricia W. Durbin, Mark W. Horovitz and Elon R. Close, Plutonium Kinetics in the Rat, Health Physics 22, 731-741 (1972).
4. Richard I. Shrager, Modelaide, A computer graphics program for the evaluation of mathematical models, N. I. H. DCRT Technical Report #5, Oct. 1970.
5. Griffith Hamlin Jr., A Multi-compartment Exponential Data Interactive Computer System (thesis), University of North Carolina Department of Computer and Information Science, 1970.
6. Aldo Rescigno, Flow Diagrams of Multi-Compartment System, Ann. N. Y. Acad. Sci. 108, 204-216 (1963).
7. A. Rescigno and G. Segre, Analysis of Multi-Compartmented Biological Systems, J. Theoret. Biol. 3, 149-163 (1962).
8. Mones Berman, Ezra Shahn and Marjory F. Weiss, The Routine Fitting of Kinetic Data to Models, Biophys. J. 2, 275-287 (1962).
9. Mones Berman, Marjory F. Weiss and Ezra Shahn, Some Formal Approaches to the Analysis of Kinetic Data in Terms of Linear Compartmental Systems, Biophys. J. 2, 289-316 (1962).
10. Grove C. Nooney, Iron Kinetics and Erythron Development, Biophys. J. 5, 755-765 (1965).

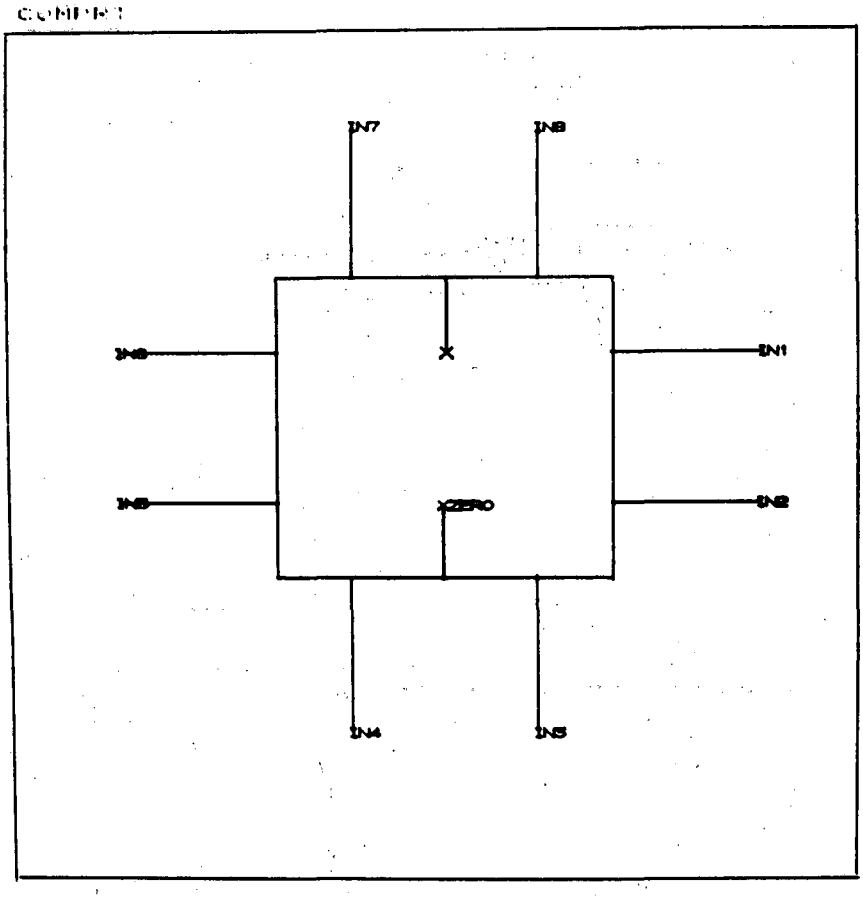

Fig. 1.

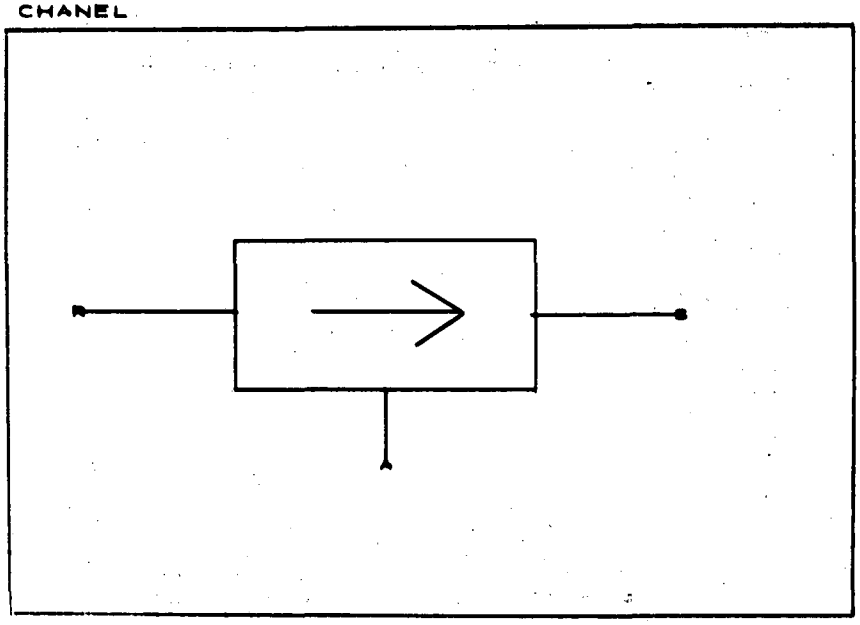

Fig. 2.

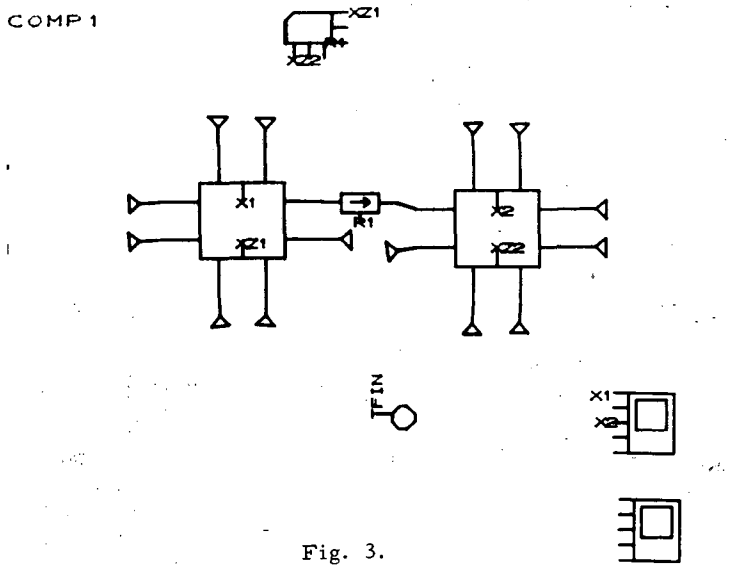

Fig. 3.

COMP 1

```
 1  $IAM      MIMIC RUN OF COMP1 PRODUCED BY MIMVERT
 2            XZ2       XZ1
 3            G001      0.0
 4            G002      0.0
 5            G003      0.0
 6            G004      0.0
 7            G005      0.0
 8            G006      0.0
 9            FG008     X1
10            FG009     X1
11            FG006     X1
12            FG003     X1
13            FG001     X1
14            FG002     X1
15            FG004     X1
16            FG005     X1
17            X1        INT((G008+G009+G006+G003+G001+G002+G004+G005),XZ1)
18            G010      FG008*R1
19            G008      -G010
20            G009      0.0
21                      PAR(XZ2,XZ1)
22                      FIN(T,TFIN)
23            G011      0.0
24            G012      0.0
25            G013      0.0
26            G014      0.0
27            G015      0.0
28            FG016     X2
29            FG017     X2
30            FG015     X2
31            FG013     X2
32            FG011     X2
33            FG010     X2
34            FG012     X2
35            FG014     X2
```

Fig. 4.

COMP 1

```
36            X2        INT((G016+G017+G015+G013+G011+G010+G012+G014),XZ2)
37            G017      0.0
38            G016      0.0
39                      PLO(T)
40                      PLO(T)
41                      END
42
```

Fig. 5.

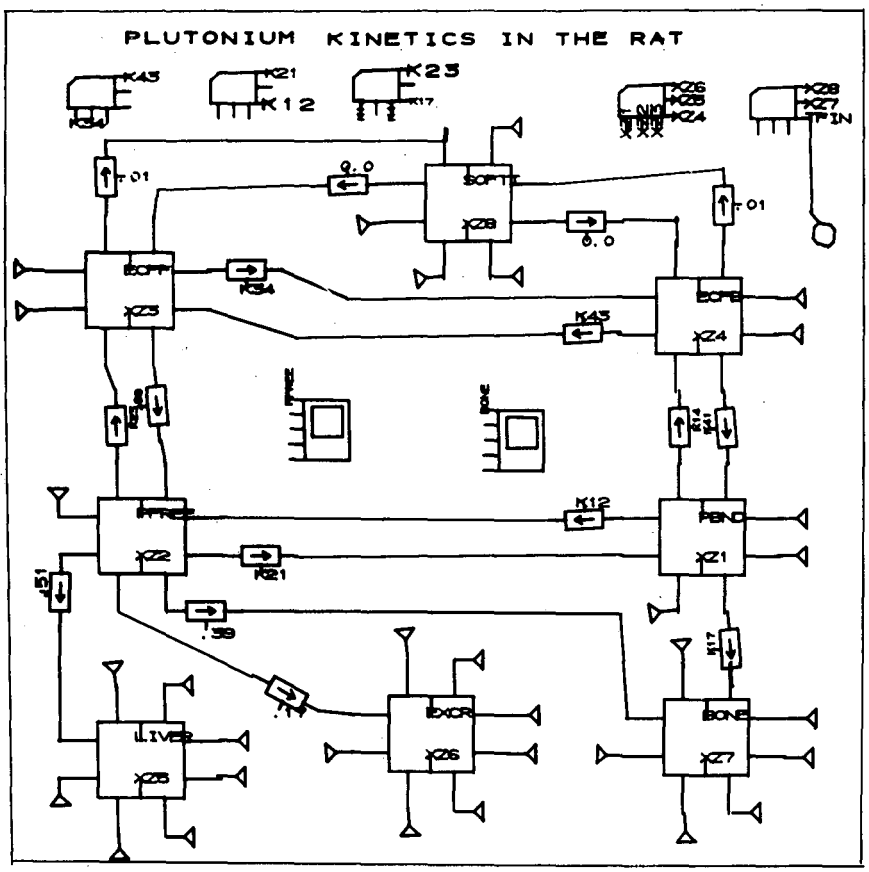
PLUTONIUM KINETICS IN THE RAT
K43
K21
K12
K23
K17
K34
K26
K25
K24
K28
K27
TFIN
.01
0.0
SOFTT
X28
.01
SOFT
X23
K34
K43
SOFE
X24
FFREE
BONE
K12
FFREE
X22
K21
PENC
X21
.39
K17
LIVER
X25
BNOR
X26
BONE
X27

Fig. 6.

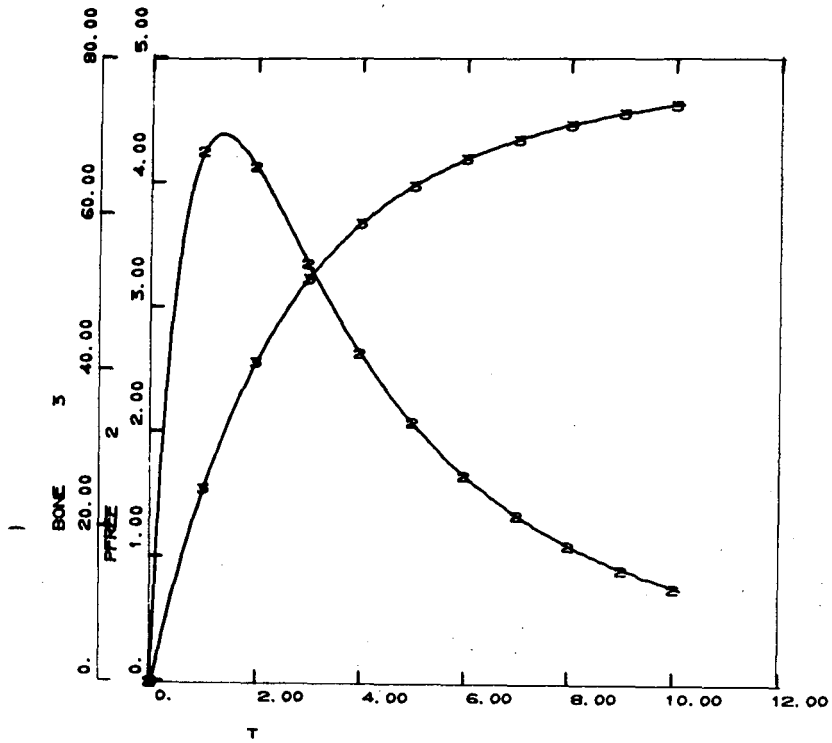
80.00
60.00
40.00
20.00
0.
5.00
4.00
3.00
2.00
1.00
0.
BONE
FFREE
0.
2.00
4.00
6.00
8.00
10.00
12.00
T

Fig. 7.

TECHNICAL INFORMATION DIVISION
LAWRENCE BERKELEY LABORATORY
UNIVERSITY OF CALIFORNIA
BERKELEY, CALIFORNIA 94720