LA-UR -82-656

Conf-820433--1

Los Alamos National Laboratory is operated by the University of California for the United States Department of Energy under contract W-7405-ENG-36

LA-UR--82-656

DE82 011983

TITLE: A FAST ALGORITHM FOR TWO-DIMENSIONAL DATA TABLE USE IN HYDRODYNAMIC AND RADIATIVE-TRANSFER CODES

AUTHOR(S): W.L. Slattery
W.H. Spangenberg

MASTER

SUBMITTED TO: Cray Research, Inc. symposium "Science, Engineering and the Cray-1".

Los Alamos Los Alamos National Laboratory
Los Alamos, New Mexico 87545

A FAST ALGORITHM FOR TWO-DIMENSIONAL DATA TABLE USE IN HYDRODYNAMIC AND RADIATIVE TRANSFER CODES

by

W.L. Slattery

and

W.H. Spangenberg

ABSTRACT

We describe a fast algorithm for finding interpolated atomic data in irregular two-dimensional tables with differing materials. The algorithm is tested in a hydrodynamic/radiative transfer code and shown to be of comparable speed to interpolation in regularly spaced tables, which require no table search. The concepts presented are expected to have application in any situation with irregular vector lengths. We also describe the procedures that were rejected either because they were too slow or because they involved too much assembly coding.

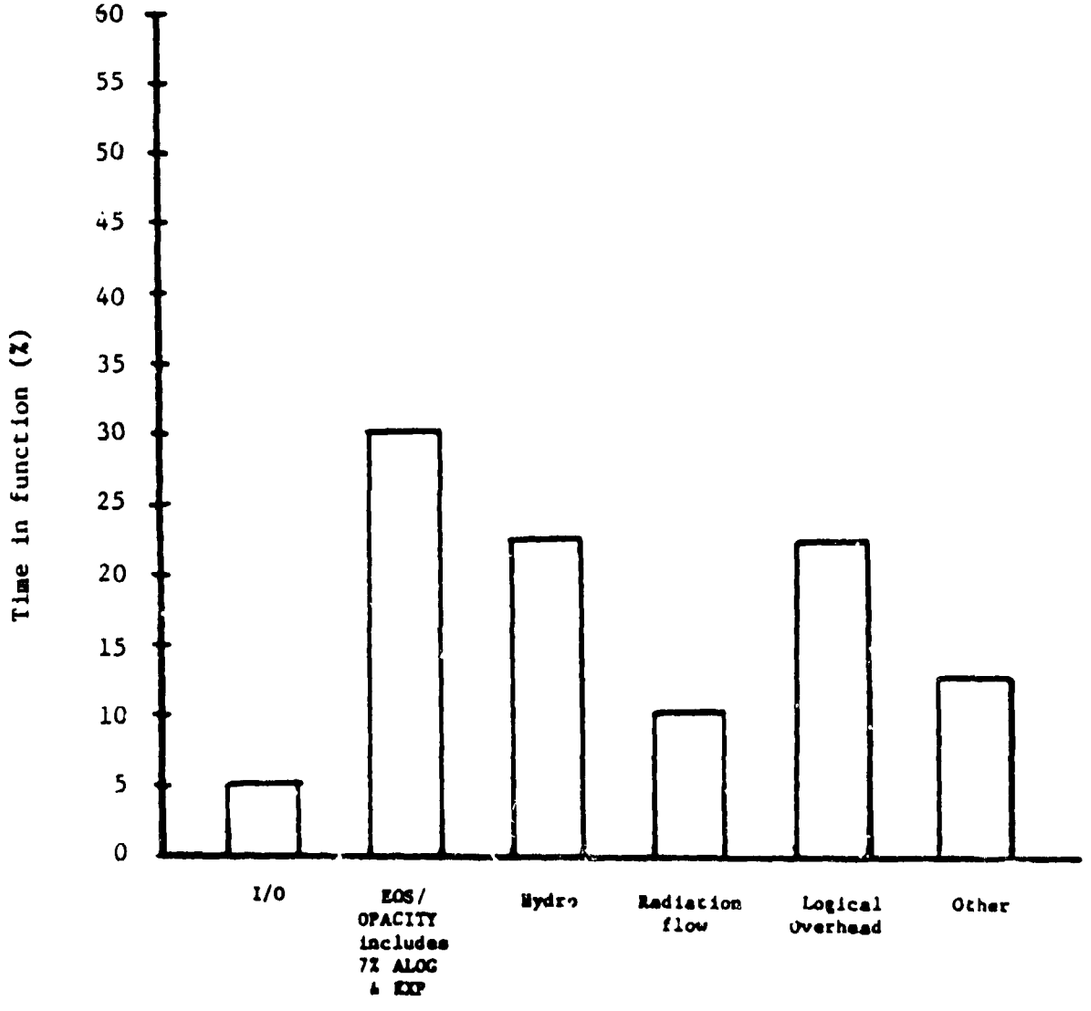
I. INTRODUCTION

Large hydrodynamic and radiation transport codes require data to represent reality. The results of detailed, but tedious, atomic physics calculations are converted into extensive two-dimensional data tables. An example is a pressure table as a function of the two independent variables, density (ρ) and temperature (T). Group T-4 has organized such data into their SESAME Library. In order to preserve accuracy, the data is given in tabular form with uneven intervals between table entries. Many points are calculated in regions of ρ-T space in which complicated atomic processes such as phase changes or ionization phenomenon occur. Other smoothly varying regimes are sparsely tabulated.

Since the table intervals are irregular, there exists no simple algorithm to find a location within the table. This constraint prevents the straightforward implementation of the SESAME data in a vector machine environment. The new SESAME data base is intended to replace the older MAPLE data base. The MAPLE data required no table search. Data for all materials in the library were tabulated on a fixed density-temperature (logarithmic) grid. Hence, the logarithm of the independent variable provided both the table index (via the characteristic) and the interpolation fraction (the mantissa).

The advantage of a data table which requires no table search is manifest in our base-line calculation. We chose a highly vectorized two-dimensional Lagrangian hydrodynamics code. The radiation transport uses the one-group diffusion approximation. The logical mesh employes 61 x 74 computational cells. Eleven different materials are defined within the mesh. A vectorized bilinear (logarithmic) interpolation scheme returns quantities (pressure, energy, and opacity) and their derivatives from the data tables as functions of cell density and temperature. 1300 computational cycles require 143 seconds on the CRAY-1. Figure 1 indicates how the calculation time is distributed. (This talley is generated by the CTSS utility SAMPLE.) Thirty percent of the calculation time is spent doing EOS and opacity interpolations; this includes 7% of the time spent in the vector ALOG and EXP routines.

The base-line problem was rerun using the SESAME data tables. A binary search routine found the table indexes for each cell. The same vectorized bilinear interpolation scheme referred to earlier provided quantities from the data tables. 1300 cycles now require 358 seconds on the CRAY! Sixty percent of the calculation time is spent in the one search algorithm, while all other code algorithms account for the remaining 40% of the time. The advantage of the flexible data representation are juxtaposed to the disadvantage of an intolerably slow table search. The purpose of our work was to devise a method in



Distribution by Function

Fig. 1. MAPLE Bench-Mark Calculation

FORTRAN whereby a table search could be done quickly, to be followed by an interpolation in the table, on a vector machine (the CRAY-1 in particular).

In Section II we describe the problem in more detail, and we give our solution to it in Section III. In Section IV we report approaches to the problem that did not work and discuss the reason that they did not work in our application. The Appendix sketches the look-ahead algorithm and indicates where a general code (FORTRAN) is available.

II. DATA USE WITHIN A LARGE HYDRODYNAMIC OR RADIATIVE TRANSFER CODE

We shall illustrate tabular data use in a typical hydro code by considering equations-of-state (pressure as a function of temperature and density such as the SESAME 301. type tables). Within such a code the equation-of-state (EOS) for several materials needs to be stored. In Fig. 2 we show hypothetical EOS for two materials. Typically, the hydrodynamic sections of a code feed the EOS search and interpolation routines large vectors (linear arrays) of (1) cell temperatures, (2) cell densities and (3) cell material identifiers. (A cell is a discrete two-dimensional spatial section of the problem.) The temperature, density, and material identifier then constitutes an ordered triple for each cell. In turn, the cells are organized in a logical mesh and are assigned indexes (K,L) for computational purposes. As an example, suppose an ordered triple for cell (K1, L1) has temperature, T(K1,L1)=8.5, density, RHO(K1,L1)=10.4 and a material index MATNDX(K1,L1)=1. Using the Material 1 table in Fig. 2 an EOS routine finds the pressure corresponding to these values as follows: First a search is made in the T table for Material 1 to find the T(index) such that T(index) $\leq$T(K1,L1) but T(index+1) >T(K1,L1). In this case,

Figure 2. Hypothetical EOS tables for two materials. The abbreviations are the following: T=temperature, RHO=density, and P=pressure.

Material 1

INDEX	T	P_1 (RHO,T)				
6	13	50	50	54	60	
5	12	46	48	52	54	
4	9	40	41	43	50	
3	8	35	37	40	47	
2	6	30	32	38	45	
1	3	28	30	40	45	
		0.3	16	17	20	RHO
INDEX		1	2	3	4	

Material 2

INDEX	T	P_2 (RHO,T)					
4	15	20	24	30	32	35	
3	10	18	22	20	22	25	
2	6	17	20	17	20	24	
1	0	15	16	17	19	20	
		1	2	4	8	25	RHO
INDEX		1	2	3	4	5	

T(index)=3. Likewise a search is made in the RHO table; since RHO=10.4, the RHO index is 1. Second, now that the indexes are known, the tabular values are gathered based on the value of the indexes. The number of values gathered depends upon the type of interpolation used to find the pressure. For example, if the interpolation is bilinear, then the temperature values 8 and 9, the density values 0.3 and 16, and the pressure values 35, 40, 37, and 41 are gathered. Third, the interpolation is performed using the gathered values to give us an approximate pressure at T=8.5 and RHO=10.4. This process is repeated for the next temperature, density, and material index triple as we proceed through the logical mesh.

We are now in a position to state the problem more precisely. For a given material, the algorithm has three distinct components:

(1) the search through the independent variable tables to obtain indexes

(2) the gathers of the dependent variables (i.e., the tabulated data) based upon the appropriate indexes from 1), and

(3) the interpolation of a cell-related quantity using the values from 2).

We have T(K1,L1, MATNDX) and RHO(K1,L1, MATNDX) for each cell. We need P(K1,L1, MATNDX). But the tabulated pressures are not functions of the cell indexes. Rather they are functions of the cell temperature and density, which are, in turn, functions of the cell indexes. Our task was to devise a method to vectorize this indirect procedure as much as possible.

III. LOOK-AHEAD AND VECTOR SEARCH

Large hydro and radiative transfer codes almost always exhibit some regularity in the arrangement of materials within adjacent cells, i.e., it is extremely rare to have materials

alternate from cell to cell. Consequently, if we could know the number of adjacent cells that have the same material index, we could use that number as our vector length and vectorize some operations. This idea led to the concept of look-ahead. Consider the first element of the material index array. If this element is subtracted from the remaining elements, the first IVL elements of the difference array will be zero, thus indicating that IVL cells have the same material index. Hence we could perform the necessary operations over a vector of length of IVL.

Which of our operations can be vectorized on the CRAY? First, there is a CALMATH library routine written by Tom Jordon (C-3) called SRCH64, which performs a vector search in a table. Given an ordered table, SRCH64 locates a vector of real values in the table and returns a vector of index locations. SRCH64 is described in the Los Alamos Program Library write-up M127. Second, the interpolations can be vectorized if there is no concern about boundaries of the table, e.g., bilinear interpolation requires no special action for extrapolation. We will discuss extrapolations in more detail later. Unfortunately, gathers are not vectorized on the CRAY-1.

In summary, our solution to the problem of data table "look up" over an irregular definition of materials, each represented by irregularly spaced tables, is the following:

1. Look-ahead on materials to find vector lengths, IVL.

2. Search on independent variable's tables (e.g., temperature and density) using SRCH64.

3. Gather dependent and independent table values on indexes returned from SRCH64.

4. Interpolate using gathered values

5. And finally, repeat steps 1-4 as necessary to sweep through the logical mesh.

We schematically illustrate the FORTRAN coding for these steps in the Appendix.

Initially we wrote a stand-alone code to develop the look-ahead concept. Figure 3 displays the results of our study. Each example performed calculations equivalent to the search- gather- interpolation sequence for 64 x 10**5 cells. The random length vector case, with an average vector length (IVL) = 9, takes 31 seconds. This approaches the optimum length case (IVL = 64) at 21 seconds. The short case with IVL=1 is the scalar case. The "overhead" associated with the look-ahead concept (categories "setup" and "look-ahead") is trivial. The three major functions are well-defined: search, base address calculation-gather, and interpolation.

The look-ahead algorithm was implemented in the two-dimensional Lagrangian hydro/radiation transport code. The base-line problem now runs 1300 cycles using SESAME data in 160 seconds. This is a major improvement over the initial SESAME algorithm that took 358 seconds. Figure 4 indicates the distribution of computational time. The 35% of the time spent doing EOS and opacity "look-up" includes 12% of the time spent within the CALMATH routine SRCHDF (M128 written by Tom Jordan) doing the table search. SRCHDF returns table spacings and interpolation fractions that eliminate later gathers. It replaces SRCH64. The advantage of the flexible data table represention compensates for the slightly longer computational times required by the SESAME-related algorithms as compared to the old MAPLE schemes.

The hydro code test-bed has a TTS option (temporary triangular subzoning). Lagrangian quadrilaterals are subdivided into four triangles. This deters mesh tangling. But the number

of cells effectively increases by four also. On the other hand, the average vector lengths, IVL, in the look-ahead scheme are increased. The TTS calculation on the base-line problem using MAPLE data requires 397 seconds to run 1300 cycles. The look-ahead algorithm with SESAME data runs 1300 cycles in 373 seconds. Figure 5 displays the timing distribution.

The look-ahead concept is extremely general with apparent applicability to a wide range of computational-logical organizations. Lagrangian formulations fix the material identification of each cell. Rezoning, either periodic or continuous, may change the logical mesh-material index correspondence from one computational cycle to the next. But it remains straightforword to look-ahead in a logical mesh that may change with each cycle. By its very nature, the Eulerian formulation exhibits a dynamic association of materials and logical mesh. But we may expect a relatively small number of mixed Eulerian cells within any mesh. Let us reiterate that we may expect some regularity of material arrangement within both a Lagrangian and an Eulerian logical mesh. Any given material region is usually defined by several cells. The necessities of computational accuracy and resolution demand it. Furthermore, the look-ahead concept may be applied to algorithm sweeps through the logical mesh in any direction and in any order. Subcycling of algorithms within a given computational proceedure causes no concerns. Iterative procedures are handled straightforwardly. The look-ahead procedure may be applied to three-dimensional and higher order data tables as well.

IV. PROCEDURES THAT DID NOT WORK

In the process of arriving at the scheme described in the last section, we tried a number of other schemes, which either were too cumbersome or were slower. We shall delineate these now.

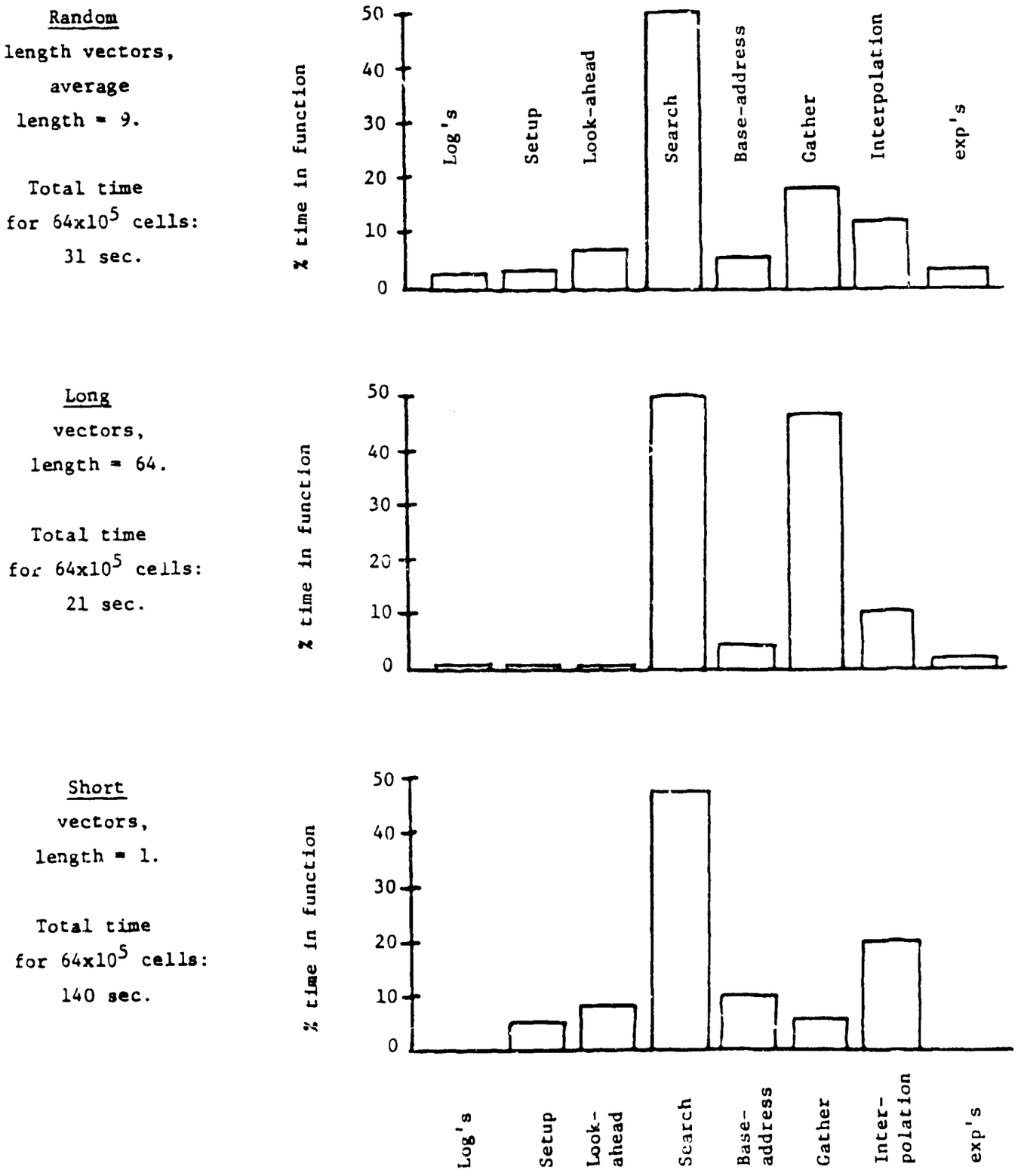


Fig. 3. The look-ahead algorithm in the stand-alone mode for bilinear interpolation. Distribution of calculational time for various vector lengths.

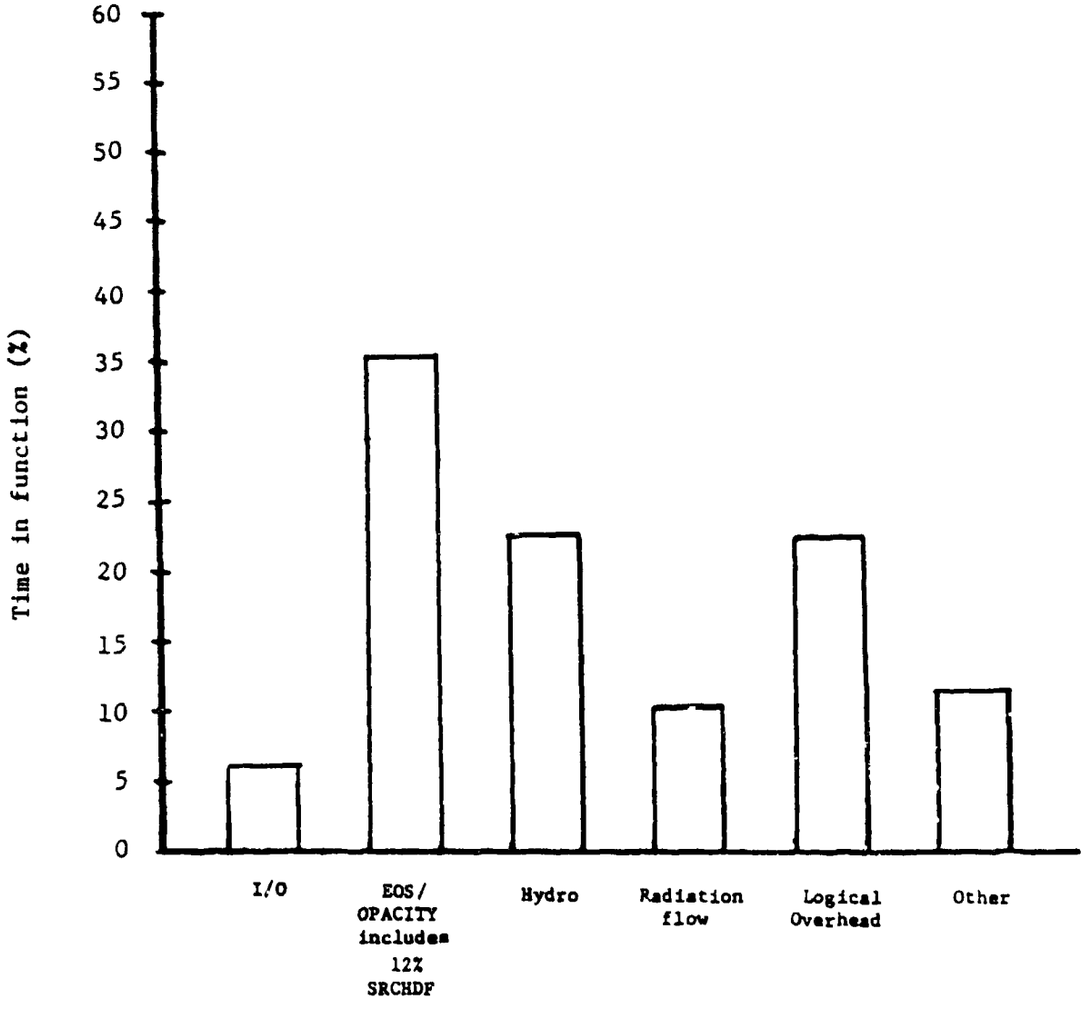


Fig. 4. SESAME with Look-Ahead Calculation

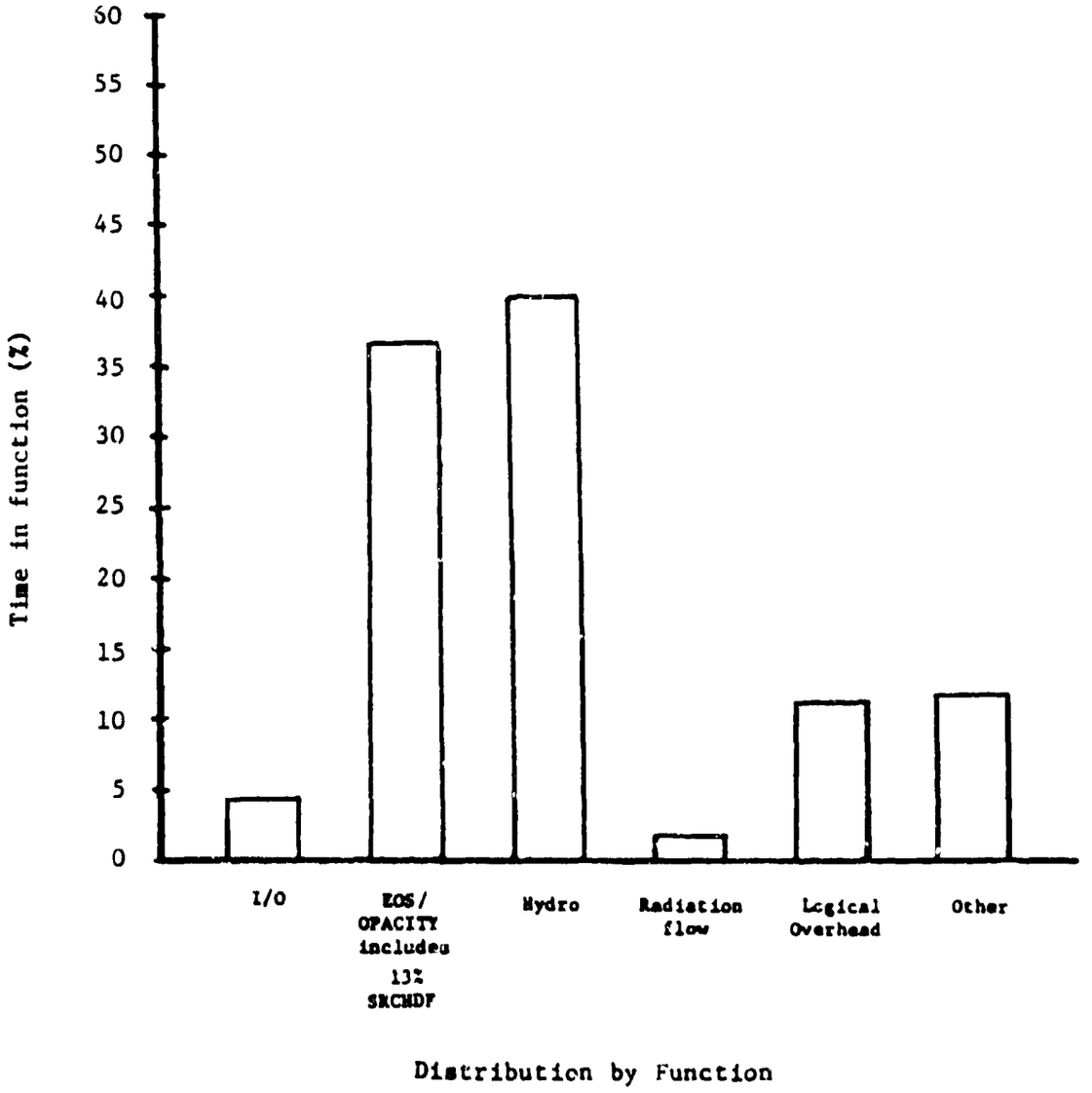


Fig. 5. SESAME with Look-Ahead and TTS Calculation

At first we included the interpolation inside the look-ahead loop, but it was discovered that if we pulled it outside, the routine would run more quickly. The time decrease was substantial for short vectors and higher order interpolation schemes like the rational function. In pursuing this line of thought one must be careful not to go too far. If the gathers are also pulled outside of the look-ahead loop, the routine runs more slowly (but not by much). Of course the table searches cannot be pulled out because of the irregular nature of the tables.

A more general procedure for doing all of the table lookup was suggested by Tom Jordon. Jordon proposed to sort all incoming data by materials, thus producing very long vectors for each material. Once sorted, SRCHRV (a variation of SRCH64) could be used for the table searches. Next gathers would be done, followed by interpolations. It would then be necessary to unsort the data to put it with the correct cells. Jordon estimated that he would be able to decrease the time required for the EOS routines by a factor of 1.5 if he wrote the whole thing in CAL. We decided that the disadvantage of CAL coding, coupled with the increased storage required, was too great a price to pay.

To replace SRCH64, we tried the CALMATH routine LINSRCH (by Tom Jordon). LINSRCH takes as input a table and a single number for which one wants the index. This routine is fast when one wants only one index (the IVL=1 case), but it is slower than SRCH64 when one needs the table indexes for many associated numbers.

In an attempt to speed up the gathers for higher order interpolations (explicitly we used G. Kerley's (X-7) rational function interpolation), we tried the CALMATH routine GATHR16. This routine worked well for long vectors (length 64), made no change in what we considered average vectors (length 9), but it severely slowed the short vectors (length 1). GATHR16 was dropped because of this severe slowdown and because of the added complication of yet another library routine.

Another attempt to save gather time was made by Kerley. Quite frequently, during a given cycle, adjacent cells in a hydro problem have roughly the same temperature and density. This means that many of the tabular values gathered are duplicates. By avoiding the regathering of tabular values, substantial time was saved in higher order interpolation schemes (factor of 2). Nevertheless, the extra storage required was prohibitive (of the order of 100,000 words decimal) and the coding was substantially more complicated. In addition such a strategy can be defeated by finely divided tables or finely zoned problems. The sorting becomes staggering during rezoning phases or with Eulerian formulations.

Extrapolations that use explicit IF statements are very expensive. Originally, the rational function interpolation had four explicit IF statements to determine if indexes were on edges of tables. If there was an extrapolation, a subroutine was called to handle an interpolation algorithm modification. Timing studies determined that up to 1/3 of the EOS routine's time was spent doing only the explicit IF statements! (This time excluded the actual extrapolation subroutines, which would only have increased the time.) Consequently, it is imperative that the tables be padded at problem definition (or setup) time to totally avoid checking for extrapolations. Alternatively, interpolation algorithms must be constructed to provide smooth and consistent extrapolations without introducing special cases.

V. CONCLUSION

Given the problem of calculating interpolated atomic data for differing materials defined by irregularly spaced two-dimensional tables, we have described a fast numerical solution to the problem. For those who would like to pursue this problem further, we have also discussed solutions that did not work. The look-ahead aspect of our algorithm, which determines vector lengths, should have application anywhere that irregular vector lengths are encountered.

VI. ACKNOWLEDGEMENTS

We wish to thank Tom Jordon (C-3) for providing invaluable assistance in assembly coding some vital subroutines. We had many useful discussions with Joe Abdallah (T-4), Dennis Brockway (X-7) and Gerry Kerley (X-7) who helped define the problem.

APPENDIX. LOOK-AHEAD AND SEARCH IN FORTRAN (SCHEMATIC)

```
      c     NUMBER = Number of cells to calculate EOS this cycle
      c     MATNDX = Material index array
      c     T      = Temperature array
      c     RHO    = Density array
      c     LD     = Difference array
      c     IVL    = Vector length in current material
      c
      c     MINO and LEADZ are CFTLIB routines
      c     MASKVN (F147) and SRCH64 (M127) are CALMATH routines
      c
      c     MASKVN provides a one-word mask (bit-vector) of
      c     conditional (X(I).NE.0) on a vector of integers or
      c     reals, N.LE.64.
      c     LEADZ counts the number of zeros before a non-zero
      c     bit is encountered
       .
       .
      IC = 1
20    IVL = MINO( NUMBER-IC,63 )

      DO 40 I=1,IVL
         LD(I) = MATNDX(IC) - MATNDX(IC+1)
40    CONTINUE


      IVL = MINO( LEADZ( MASKVN(LD,IVL,1) ),IVL ) + 1

      CALL SRCH64( IVL,T,...,IT,...)
      CALL SRCH64( IVL,RHO,...,IRO,...)

      DO 50 I=1,IVL
         Gather T, RHO and EOS table values on indexes IT and IRHO
50    CONTINUE

      IC = IC + IVL
      IF( IC .LE. NUMBER ) GO TO 20

      DO 60 I=1,NUMBER
         Interpolate EOS and/or derivatives for cells
60    CONTINUE
      .
      .
```

A general FORTRAN code (CTSS) to fetch the SESAME tables and to compute EOS and grey opacity values using the look-ahead concept with several options for interpolation has been written by C. Cranfill (X-7). It is available for general use upon request.