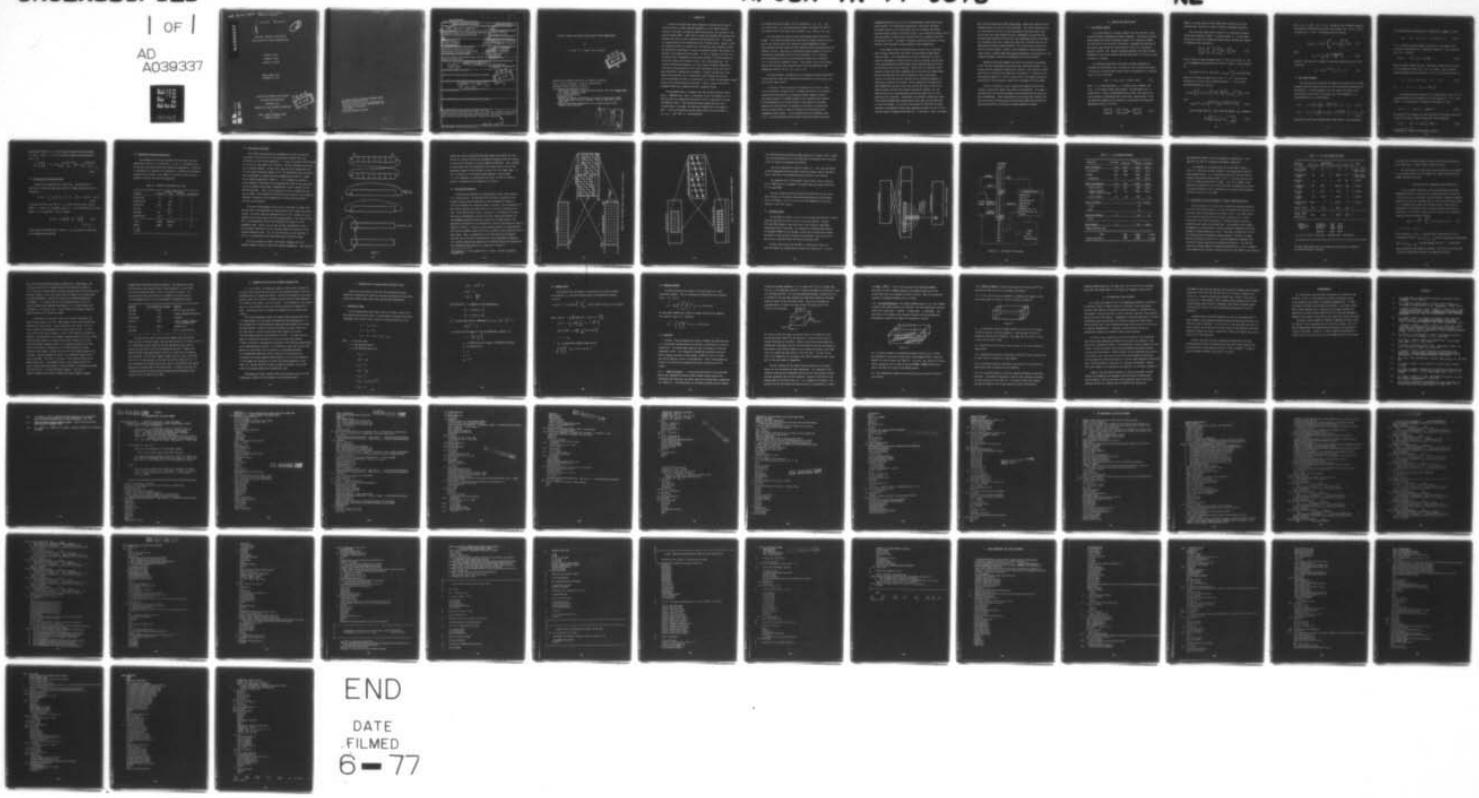
AD-A039 337 UNIVERSITY OF SOUTHERN CALIFORNIA LOS ANGELES DEPT O--ETC F/G 9/2
PIPELINE, PARALLEL AND SERIAL REALIZATION OF PHASE DEMODULATORS--ETC(U)
NOV 76 R S BUCY, K D SENNE, H YOUSSEF F44620-76-C-0085
AFOSR-TR-77-0576 NL

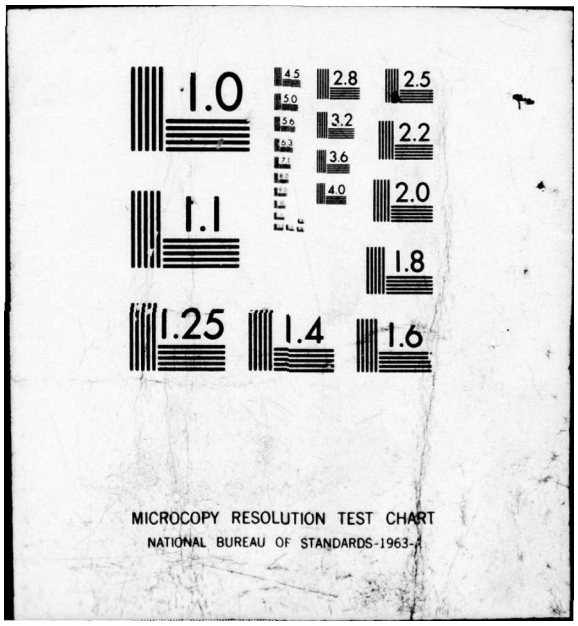
UNCLASSIFIED

| OF |
AD
A039337



END

DATE
FILMED
6-77


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0
5.6
6.3
MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

ICASE REPORT

PIPELINE, PARALLEL AND SERIAL REALIZATION OF PHASE DEMODULATORS

Richard S. Bucy

Kenneth D. Senne

Huessin M. Youssef

Report Number 76-31

November 22, 1976

INSTITUTE FOR COMPUTER APPLICATIONS

IN SCIENCE AND ENGINEERING

Operated by the

UNIVERSITIES SPACE RESEARCH ASSOCIATION

at

NASA's LANGLEY RESEARCH CENTER

Hampton, Virginia

AIR FORCE OFFICE OF SCIENTIFIC RESEARCH (AFSC)
NOTICE OF TRANSMITTAL TO DDC
This technical report has been reviewed and is approved for public release IAW AFR 190-12 (7b).
Distribution is unlimited.
A. D. BLOSE
Technical Information Officer

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

REPORT DOCUMENTATION PAGE	READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER: AFOSR - TR - 77 - 0576	2. GOVT ACCESSION NO. / 3. RECIPIENT'S CATALOG NUMBER: (9)
4. TITLE *(and Subtitle)*: PIPELINE, PARALLEL AND SERIAL REALIZATIONS OF PHASE DEMODULATORS.	5. TYPE OF REPORT & PERIOD COVERED: Interim rept.,
	6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s): (10) Richard S. Bucy, Kenneth D. Senne, Hussein Youssef	8. CONTRACT OR GRANT NUMBER(s): (15) F44620-76-C-0085, AF-AFOSR-2141-71
9. PERFORMING ORGANIZATION NAME AND ADDRESS: University of Southern California, Department of Aerospace Engineering, Los Angeles, California 90007	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: 61102F, (16) 2304/A1, (17) AL
11. CONTROLLING OFFICE NAME AND ADDRESS: (11) Air Force Office of Scientific Research/NM, Bolling AFB DC 20332	12. REPORT DATE: 22 Nov 76
	13. NUMBER OF PAGES: 67
14. MONITORING AGENCY NAME & ADDRESS *(if different from Controlling Office)*: (12) 69p. (18) AFOSR, ICASE	15. SECURITY CLASS. *(of this report)*: UNCLASSIFIED
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT *(of this Report)*

Approved for public release; distribution unlimited

17. DISTRIBUTION STATEMENT *(of the abstract entered in Block 20, if different from Report)*

(19) TR-77-0576, 76-31

18. SUPPLEMENTARY NOTES

19. KEY WORDS *(Continue on reverse side if necessary and identify by block number)*

20. ABSTRACT *(Continue on reverse side if necessary and identify by block number)*

Timing results for the CDC 6600, 7600, Star, the Illiac, PDP11-70, IBM 370-168 and presented for the phase demodulation problem. Architectural features of the machines and their impact on fast software are discussed.

402 019

DD FORM 1473 EDITION OF 1 NOV 65 IS OBSOLETE

PIPELINE, PARALLEL AND SERIAL REALIZATION OF PHASE DEMODULATORS[4]

by

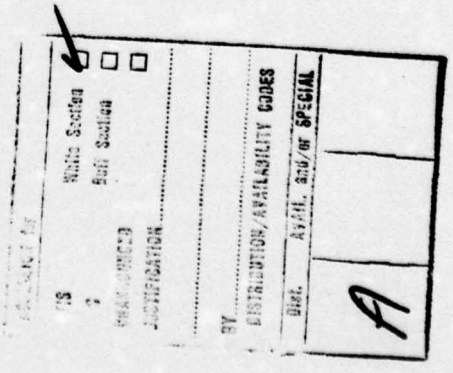
R. S. Bucy[1], K. D. Senne[2], and H. Youssef[3]

[1]University of Southern California, Los Angeles, California.

[2]M.I.T., Lincoln Laboratory, Lexington, Massachusetts.

[3]Lockheed Aircraft, Burbank, California.

[4]This research was supported in part by:
United States Air Force, Air Force Systems Command under Grant and Contract F44620-76-C-0085. AFOSR71-2141
M.I.T. Lincoln Laboratory
Lockheed Aircraft
Institute for Computer Applications in Science and Engineering (ICASE), NASA Langley Research Center, Hampton, Virginia, under NASA Contract NAS1-14101.
Institute for Advanced Computation (I.A.C.), Ames Research Center, Moffett Field, California.



1. INTRODUCTION

A theory of designing the "best" processor for estimating the state of a Markov process or signal observed indirectly as a zero memory nonlinear function of the state, corrupted by additive white noise, was developed in the early and middle 1960s. "Best" here is to mean the estimate that minimizes the expected loss, with the loss a function of the estimation error. This theory is a generalization of the Kalman-Bucy linear filtering theory. However, unlike the Kalman-Bucy theory, no blueprint for the black box which accepts as inputs the observations and produces as outputs the optimal estimate, the minimum loss estimate, can be deduced from the theory. In fact, in general, the black box, nonlinear filter corresponding to the optimal estimate is infinite dimensional in that the state of the nonlinear filter is not finite dimensional. For important technological problems, the signal process estimators generated by linearization and application of the linear Kalman-Bucy design often exhibit poor performance, sometimes even divergence. Further, even when the linear design seems effective, one is interested in what estimator has the best possible performance and what this performance is, in order to know whether further effort on this optimal design is justified. Finally, study of the optimal estimator allows one to generate effective suboptimal designs.

For the above reasons, it became clear that building nonlinear filters was an important task. Now it is quite easy to see, for example see [10], that the black box, determining the optimal filter, is specified by the maps with domain at time n, the observation process sample path up to and including time n and range, the conditional probability of the signal at time n given the observation process sample path up to and including n, $J_n(\cdot, \underline{z}_0, \ldots, \underline{z}_n)$, with $\underline{z}_i$ the observations.

Our problem then has two phases, one the replacement of $J_n(\cdot, \underline{z}_0 \ldots, \underline{z}_n)$ by a finite vector $\underline{J}_n$, the representation problem, and second, the realization problem which is the means used to generate $\underline{J}_{n+1}$ from $\underline{J}_n$ and $\underline{z}_{n+1}$.

The realization tool which has been used has been digital and hybrid systems. The problems considered have been the cubic sensor, passive receiver and the one, two and three state dimensional phase demodulation problem. These problems have been chosen for their importance, but also because the state dimension was low, a necessity if sufficiently precise estimates of the error performance of the optimal filter are to be generated. This is because error performance can so far only be evaluated by Monte Carlo simulation. This is also true for suboptimal filters. The problems which we have studied the most are the cubic sensor problem and the two-state dimensional phase demodulation problem; see [4], [11], [12], & [13].

The representations considered for the two-dimensional phase demodulation problem have been: point mass, Gauss-Hermite polynomials, Fourier Series, and Cubic Splines under tension.

On the basis of the structure of the equations to be solved in order to build an optimal nonlinear filter, it became clear early in our synthesis research effort that parallel or associative digital computers offered considerable speed-up in estimate production over standard serial machines of third generation; see [1] and [4] where these effects are discussed. Because of our lack of access to machines of the parallel or associative type in the early 1970's, a parallel processor was constructed using a contemporary hybrid system. In [2] the feasibility was considered, while in [3] , [5] the construction of the actual system was reported along with

subsequent Monte Carlo runs, all for a one-dimensional signal process cubic sensor problem. The results were encouraging; in particular the hybrid realization proved to be 16 times faster than an all-digital realization, using the digital portion of the hybrid system. Our ultimate aim, however, was to study the effect of using CDC Star 100 and Illiac IV as synthesis tools for the nonlinear filter. This paper is devoted to detailing the impact of these machines on the nonlinear filtering problem of phase demodulation.

In the summer of 1975, we obtained access to the Illiac IV and made plans to access the CDC Star 100, when it was delivered to NASA Langley Research Center. It was decided that in order to effectively be able to judge the improvement, we should initially realize a problem, the two-dimensional phase demodulation problem where we had extensive numerical results and experience on the serial CDC 6600. Further for this problem we could generate estimate and signal sequences with an existing serial 6600 program. We were convinced that this serial program was close to the fastest possible, and that that was not the case, was an interesting byproduct of producing an effective Star program. While studying this two-dimensional phase demodulation problem on the Illiac and the Star gave us good data on speed-up factors possible with pipeline and parallel machines, our real objective was to do a problem which was beyond the capabilities of third generation serial machines. Consequently, we developed a generalized three-state dimension demodulation problem where phase, phase rate and amplitude all must be estimates. In the Fall and the Spring of 1975-1976, Ken Senne developed a two-dimension optimal phase demodulator program for the Illiac IV using the Glypnir language and the Arpa net. In the month of June, the authors

were visiting scientists at ICASE, NASA Langley. During this time, the corresponding program for the Star 100 was developed. This latter program not only was very effective for Star (a large percentage of operations were of long vector type, over 4,000 component vectors, consequently assuring that Star was doing long streaming), but also this Star program was used as a prototype of an extremely fast serial program which was over 2-1/3 times faster on the 6600 than our original program reported in [4]; also see the listing of the cyclic point mass program. Finally, a Star program for a three-dimensional signal process, combining amplitude and phase estimation, was developed and time comparison between it and the corresponding serial version obtained.

Methods of obtaining extremely fast serial realization of the optimal filter are being investigated using optical and surface wave devices. It is curious that the mathematical mapping which permits a time correlator surface wave device to calculate multi-dimensional convolutions is intrinsic in the Star program--compare [9]. We will discuss later in this paper the possible application of array processors coupled with minicomputers to this problem.

In [10], a discussion is given comparing the speed-up possibilities inherent in realization method versus density representation. This paper is concerned only with realization methods and uses essentially only the point mass representation. Other representations can be used, but in general they must be accuracy calibrated against the point mass method, and the effect of pipeline and parallel realization on estimate speed-up is unclear.

2. COMPUTER REALIZATION SURVEY

2.1 The Physical Problem

It has been observed in previous studies ([4]) that nonlinear filters can be efficiently implemented on parallel computers. Since every so-called vector machine has unique limitations, however, it is necessary to adapt the algorithm to each particular architecture. Candidate architectures for the present study include the array processor (Illiac IV), the pipeline processor (CDC-Star 100), and the look-ahead processors (CDC 6600 and 7600). Benchmarks on fast serial machines (IBM 370-168 and PDP 11-70) have also been included for reference.

An interesting application for optimal nonlinear estimation was introduced by Mallinckrodt, Bucy, and Cheng [7], who considered the problem of tracking a first-order phase process based on measurements of a modulated signal in noise of the form

$$ds(t) = A \cos [\omega_0 t + x_1(t)]dt + dv(t) \qquad (2.1)$$

where A is a known amplitude, ω_0 is a known carrier frequency, and $x_1(t)$ is the message process being tracked. The measurement noise is assumed white. Using a voltage-controlled oscillator, the known carrier may be removed by heterodyning down to base band, producing both in-line and quadrature components and resulting in an equivalent two-dimensional measurement process of the form

$$\begin{bmatrix} dz_1(t) \\ dz_2(t) \end{bmatrix} = \begin{bmatrix} \cos x_1(t) \\ \sin x_1(t) \end{bmatrix} dt + \begin{bmatrix} dv_1(t) \\ dv_2(t) \end{bmatrix}, \qquad (2.2)$$

where A has been taken as unity without loss of generality, and the noise has been replaced by a vector of mutually independent quantities.

The first-order phase process studied in [7] consisted of Brownian motion with increment of length h having variance qh. In this paper, we describe a study of a second order phase process involving the integral of Brownian motion, expressed as

$$\begin{bmatrix} dx_1 \\ dx_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} dt + \begin{bmatrix} 0 \\ 1 \end{bmatrix} d\beta_t \qquad (2.3)$$

We will retain the same measurement model (1) and let the noises v_1 and v_2 be independent Brownian motions with paths of length h having variance rh.

The optimal filter is specified by $J_{n+1|n+1}\begin{pmatrix} y_1 \\ y_2 \end{pmatrix}$, the conditional density of the phase and phase rate at time $(n+1)\Delta$, given discrete observations up to this time at sampling interval Δ. The following equations determine this density:

$$J_{n+1|n+1}\begin{pmatrix} y_1 \\ y_2 \end{pmatrix} = D_{n+1}(y_1) \int_{-\infty}^{\infty} \exp\left\{ - \frac{(y_2-\mu)^2}{2q\Delta} \right\} J_{n|n}\begin{pmatrix} y_1-\mu\Delta \\ \mu \end{pmatrix} d\mu \ , \qquad (2.4)$$

where

$$D_{n+1}(y_1) \stackrel{\Delta}{=} C_o \exp\left\{ \frac{z_1(n+1) \cos y_1 + z_2(n+1) \sin y_1}{r/\Delta} \right\} . \qquad (2.5)$$

It can be shown--see [4]-- that a modulated density $\tilde{J}_{n|n}$ defined as

$$\tilde{J}_{n|n}\begin{pmatrix} \sigma \\ \tau \end{pmatrix} \stackrel{\Delta}{=} \sum_{k=-\infty}^{\infty} \sum_{l=-\infty}^{\infty} J_{n|n}\begin{pmatrix} \sigma + 2\pi k \\ \tau + \frac{2\pi l}{\Delta} \end{pmatrix} . \qquad (2.6)$$

with $-\pi \leq \sigma < \pi$, and $-\pi/\Delta \leq \tau < \pi/\Delta$, carries all the information necessary for nonlinear filtering when the cyclic loss function $\frac{1}{2}(1 - \cos \varepsilon_1)$ with ε_1 the phase error, is used. The modulated density satisfies

$$\tilde{J}_{n+1|n+1} = D_{n+1}(\sigma) \int_{-\pi/\Delta}^{\pi/\Delta} a(\tau-\xi)\, \tilde{J}_{n|n}\begin{pmatrix}\sigma-\xi\Delta \\ \xi\end{pmatrix} d\xi, \qquad (2.7)$$

where

$$a(\tau-\xi) \triangleq \sum_{i=-\infty}^{\infty} \exp\left\{ - \frac{(\tau-\xi+2\pi i/\Delta)^2}{2q\Delta} \right\} \qquad (2.8)$$

We recall that the cyclic estimate, the one that minimizes the cyclic loss, is given by

$$x_n^* = \arg E\, e^{ix_1(n)} \Big| z_i^1,\ z_i^2 \quad i \leq n) \qquad (2.9)$$

2.2 Point Mass Placement

The most attractive formulation for numerical solution of the filtering problem on parallel machines involves the use of point-mass representation of the densities. A fixed rectangular grid will consist of m points in the phase variable and n points in the phase rate, with coordinates (i, j) corresponding to (σ,ξ) by the formulae

$$\sigma(i) = -\pi + 2\pi\left(\frac{i}{m}\right) + \pi\left(\frac{1}{m}\right) = \pi\left(\frac{1+2i}{m} - 1\right),\ i = 0,\ldots,\ m-1 \qquad (2.10)$$

$$\xi(j) = -\pi/\Delta + 2\pi/\Delta\left(\frac{j}{n}\right) + \pi/\Delta\left(\frac{1}{n}\right) = \pi/\Delta\left(\frac{1+2j}{n} - 1\right),\ j = 0,\ldots,n-1 \qquad (2.11)$$

Note that the phase rate variables may be used directly in the convolution

(2.7) but the phase variables must be interpolated in general, so that

$$\eta(k) = \sigma(i) - \Delta\xi\,(j) \Rightarrow \quad k = i + \frac{m}{2} - \frac{m}{n}\left(\frac{1}{2} + j\right) \tag{2.12}$$

If m is taken as an even integer, such that n/m is an integer, then (2.12) may be decomposed into a subdominant integer $[k]$ and a fractional part Δk, as follows:*

$$[k(i)] = i + \frac{m}{2} - \left(\frac{1}{2} + j\right) \text{ DIV } \left(\frac{n}{m}\right) - 1 \quad . \tag{2.13}$$

where DIV denotes integer division. Note that the addition of $\frac{1}{2}$ in (2.13) will not change the result since n/m is an integer. Next, we observe that the remainder after the division in (2.13) is the fractional part Δk:

$$\Delta k = \left(\frac{1}{2} + j\right) \text{ MOD } \left(\frac{n}{m}\right), \quad j = 0, \ldots, n-1 \tag{2.14}$$

or $$1 - \Delta k = 1 - \left(\frac{1}{2} + j\right) \text{ MOD } \left(\frac{n}{m}\right) .$$

The interpolated result desired will be between $[k(i)]$ and $[k(i)] + 1$, (evaluated modulo m) with weighting Δk on the former and $1 - \Delta k$ on the latter. If the convolution is formed for fixed phase (i), then

$$[k(i + 1)] = [k(i)] + 1 \quad \text{(modulo m)} . \tag{2.15}$$

The relation (2.15) suggests the recursion which will be used on the Illiac. (2.15) will be used to iterate, based on an initial condition

$$[k(\,)] = \left(\frac{m}{2} - 1 - j \text{ DIV } \frac{n}{m}\right) \text{ MOD } m \tag{2.16}$$

---

* Note that $[k]$ should be interpreted as modulo m.

During such iteration, i, Δk will be used to weight the term evaluated at $[k(i)]$, while $1 - \Delta k$ will be used to weight the term evaluated at $[k(1 + i)]$.

$$J^{i}_{n|n}\begin{pmatrix} n(k) \\ \xi(j) \end{pmatrix} \tilde{=} (1 - \Delta k)\, J_{n|n}\begin{pmatrix} n([k(i+1)]) \\ \xi(j) \end{pmatrix} + \Delta k\; J_{n|n}\begin{pmatrix} n([k(i)]) \\ \xi(j) \end{pmatrix} \tag{2.17}$$

2.3 Evaluating the Filtered Data Term

The term (2.8) consists of an infinite sum. The values which Γ may take on in the grid coordinates consist of integer multiples of $2\pi/n\Delta$, resulting in

$$a(\Gamma(p)) = \sum_{\ell=-\infty}^{\infty} \exp\left[-\frac{2}{q\Delta}\left(\frac{\pi}{\Delta}\right)^2\left(\frac{p}{n} + \ell\right)^2\right] \quad p = -2(n-1),\ldots,2(n-1) \tag{2.18}$$

It may be seen from (2.18) that $a(\cdot)$ is an even function which is cyclic, modulo n. Further, for reasonable values of q the contribution of all terms except $\ell = 0$, is negligible. Thus, we compute

$$a(\Gamma(p)) = \exp\left[-\frac{2}{q\Delta}\left(\frac{\pi}{\Delta}\right)^2\left(\frac{|p|}{n}\right)^2\right] \tag{2.19}$$

$$0 \leq |p| \leq n - 1 .$$

In the examples described below, 5 terms of $a(\cdot)$ were taken to be non-zero for the computing convolutions.

2.4 Assessment of Required Computations

The computations required to implement the point mass filter are summarized in Table 2.1, as a function of m and n. The sensor terms are the only ones which require math functions (exponentials). Since only m exponentials are required, this computation is generally insignificant compared with the overall filter update, so no special effort has been expended to optimize the required computations.

TABLE 2.1 ARITHMETIC OPERATIONS FOR FILTER

Function	Number of Operations			
	Multiplies	Adds	Divisions	Exponentials
Sensor Terms	2m	m	0	m
Interpolation	mn	2mn	0	0
Convolution	5mn	10mn	0	0
Row Sums	0	mn-m	0	0
Estimates	3m	3m-3	1	0
Normalization	mn+m	0	0	0
Total	7mn+6m	13mn+3m-3	1	m
Example	28864	53341	1	32

m = 32
n = 128

2.5 The Illiac IV Algorithm

The primary considerations for programming the Illiac IV array are the proper utilization of all of the 64 Processor Elements (PE's) and minimization of data routing between PE's. In order to accomplish the efficient use of the PE's, the values of m = 32 and n = 128 were selected and utilized for all machine examples. On the Illiac, two rows of PE storage were used for each value of the phase samples in JN. The interpolation and convolution were accomplished in a totally parallel fashion, using all 64 PE's, except for the operation depicted in Figure 2.1. It is necessary to perform a cyclic rotation of each phase row to accumulate the terms for the convolution (a cyclic rotation to the right is combined with a cyclic rotation to the left at each step). Since a cyclic routing on Illiac IV involves only one row at a time, it was necessary to form the two-row rotation by two single-row rotations, followed by an end element switch (involving three transfers with only one PE enabled).

The overall effectiveness of the Illiac IV algorithm is evident from the fact that so few operations are required which involve fewer than 64 PE's enabled. The sensor terms are computed with only 32 PE's enabled, but this operation only involves about 5% of the estimate update. The single PE transfers in the convolution are also only responsible for about 5% of the computation time. Finally, the row sums are done logarithmically with a PE utilization efficiency of 16.7%, and they account for another 5% of the estimate computation time. Thus, the net efficiency of this algorithm is 87.5%, or the equivalent of 56 times faster than a single PE program.

The Illiac program was coded in the Glypnir language, utilizing assembly language listings to reduce the overhead computations. The resulting

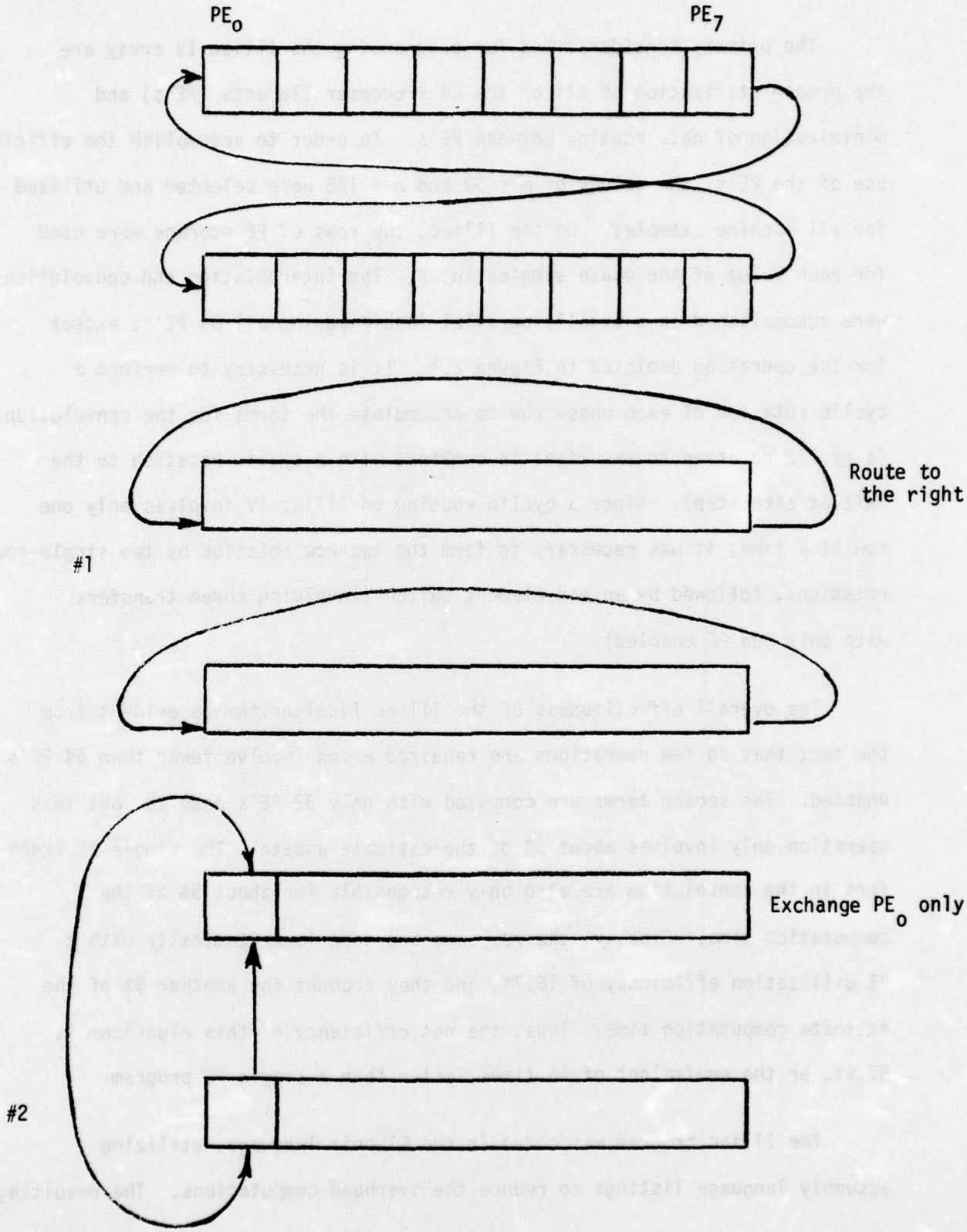


FIGURE 2.1

program was timed in the non-overlap mode, wherein the Control Unit (CU) and the PE's operate serially and considerable overhead cycles are required to allow error conditions to ring out between instructions. Thus, although we would expect the Illiac program to run at least twice as fast as the equivalent program for the CDC-STAR, in fact it ran five times slower. It will be reported in [17] how the program operates in overlap mode. It is also important to indicate that the Illiac IV Clock is running at 80 nsec. as compared with the design goal of 50 nsec.

2.6 The CDC-STAR Algorithm

The pipeline architecture is unconstrainted by small fixed resources (i.e., 64 processors). On the other hand, efficient utilization of the pipeline requires detailed attention to pre-arrangement of vectors to allow for streaming from consecutive memory locations. This consideration is particularly important for STAR, which has a relatively slow memory cycle time. Since the nonlinear filter is recursive, it is necessary to include the vector re-arrangement as part of the filter update and therefore the re-arrangement constitutes the major overhead of the STAR program. The operations on the matrix JN to precondition it for the convolution are shown in Figures 2.2 and 2.3. First, the column-ordered JN matrix is column-shuffled with itself to produce a matrix which has two copies of every phase variable in each column. Then, a scrambled JN can be formed which has the property that each row in the final convolved matrix can be generated by operating on a suitable interpolation between the two adjacent rows of the scrambled JN. The interpolation which does this is depicted in Figure 2.3.

---

*The Figures 2.3 - 2.4 are shown with m = 4 and n = 16 for illustrative purposes only.

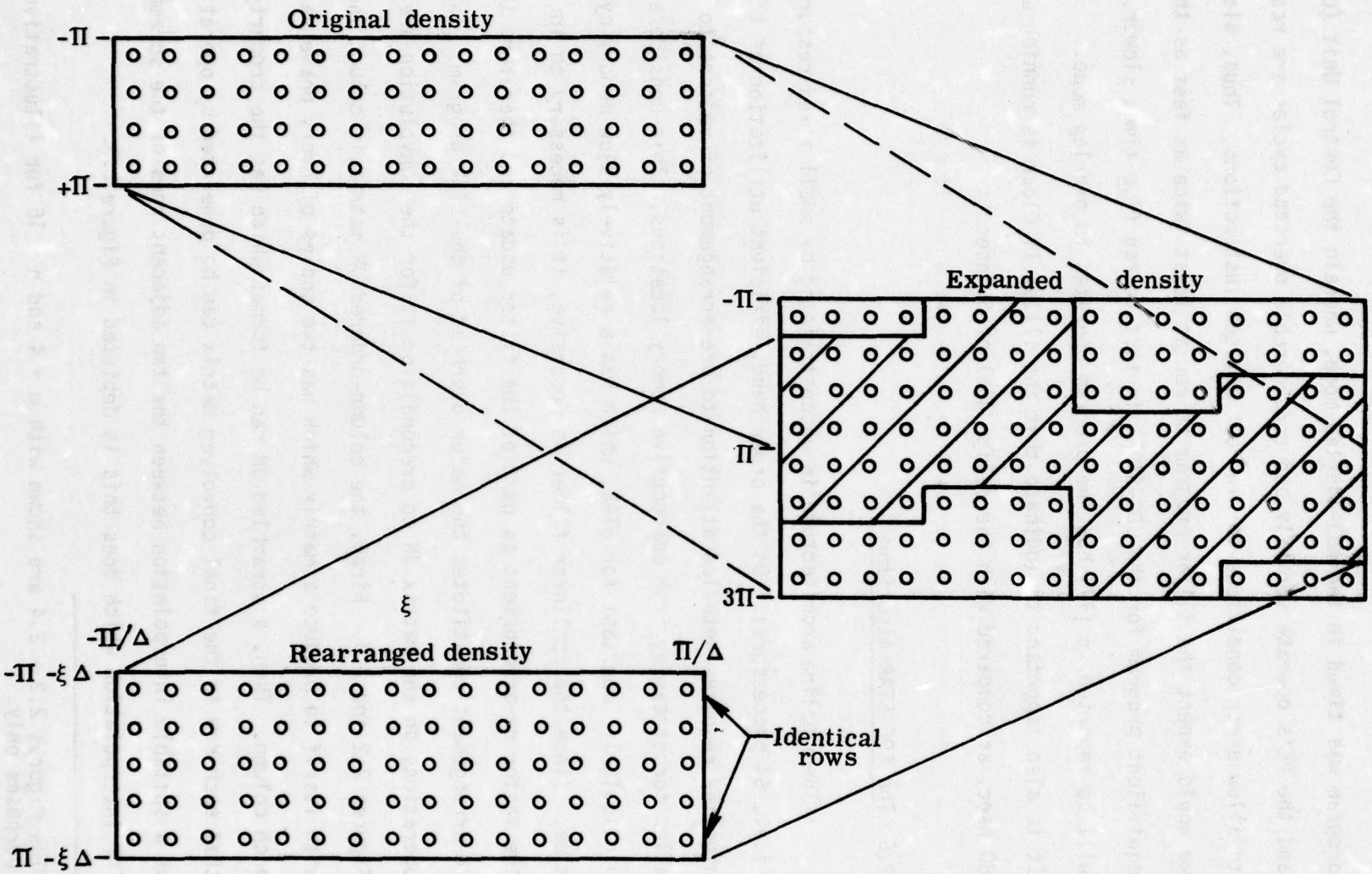


Figure 2.2.- Scrambling phase variable prior to convolution.

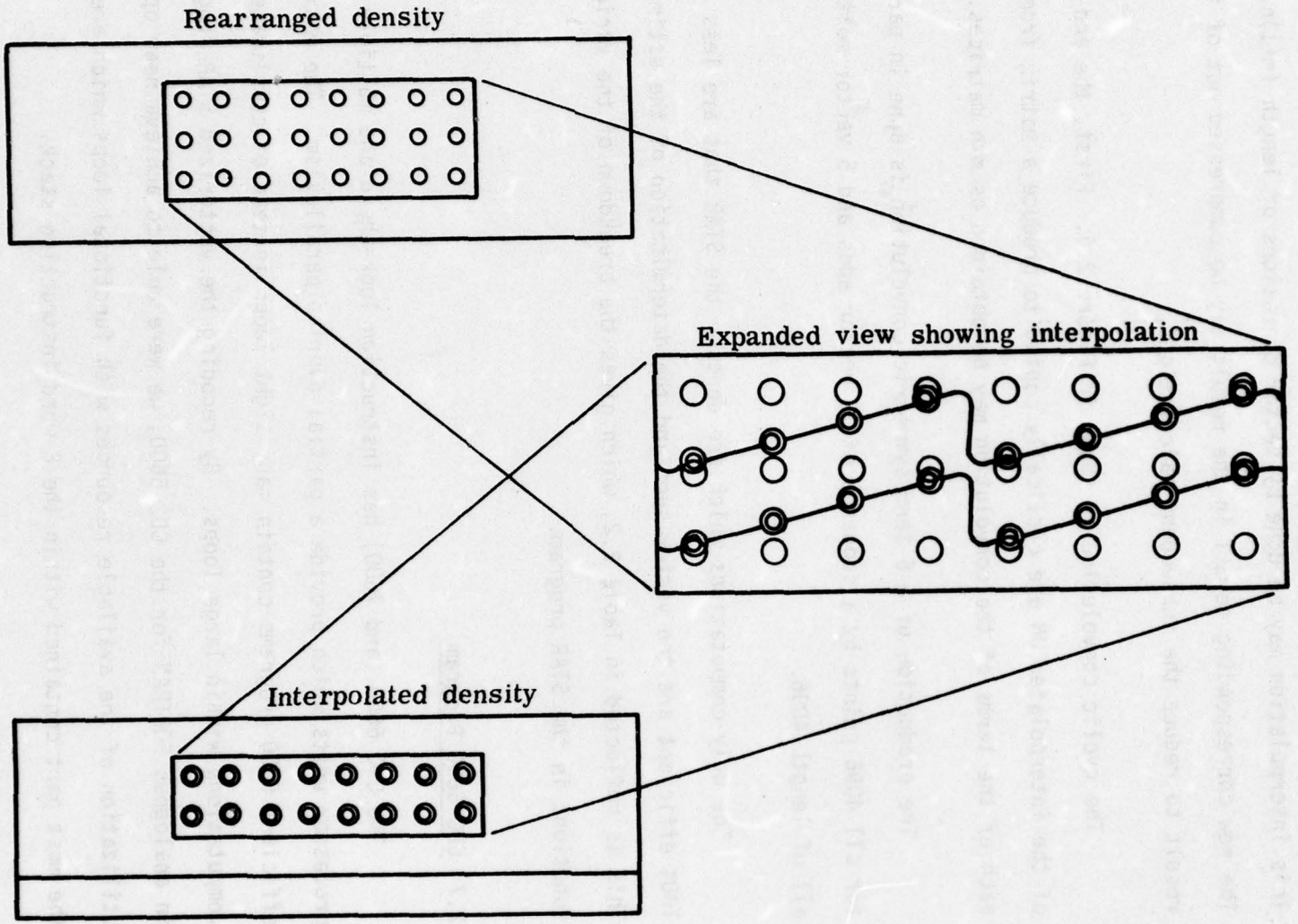


Figure 2.3.- Interpolation of scrambled matrix.

This interpolation may be done by vector operations of length (m+1)n, or 4224. The row corresponding to m+1 in the result may be compressed out of the final result to reduce the subsequent calculations.

The cyclic convolution is shown in Figure 2.4. First, the end columns of the interpolated JN are cyclically copied to produce a matrix from which each of the terms of the convolution may be obtained as mxn matrices.

The production of a 5-term symmetric convolution is done in parallel for all 4096 points by a sequence of 10 vector adds and 5 vector multiplies, all of length 4096.

The only computations which are done on the STAR that are less than 100% efficient are the vector sums and the determination of the estimates. This is reflected in Table 2.2, which gives the breakdown of the various functions in the STAR program.

2.7 CDC 6600 Program

The CDC 6600 (and 7600) has instruction look-ahead and multiple arithmetic processor units which provide a partial overlap parallelism. The most efficient 6600 programs contain many tight loops instead of complicated computations within large loops. By recoding the vectorized STAR program in analogous FORTRAN for the CDC 6600, we were able to achieve near optimal utilization of the available resources with functional loops which are for the most part contained within the 8-word instruction stack.

The basic data flow of the CDC 6600 is illustrated in Figure 2.5. Reads from memory are accomplished by setting the A Registers A1 - A5 with

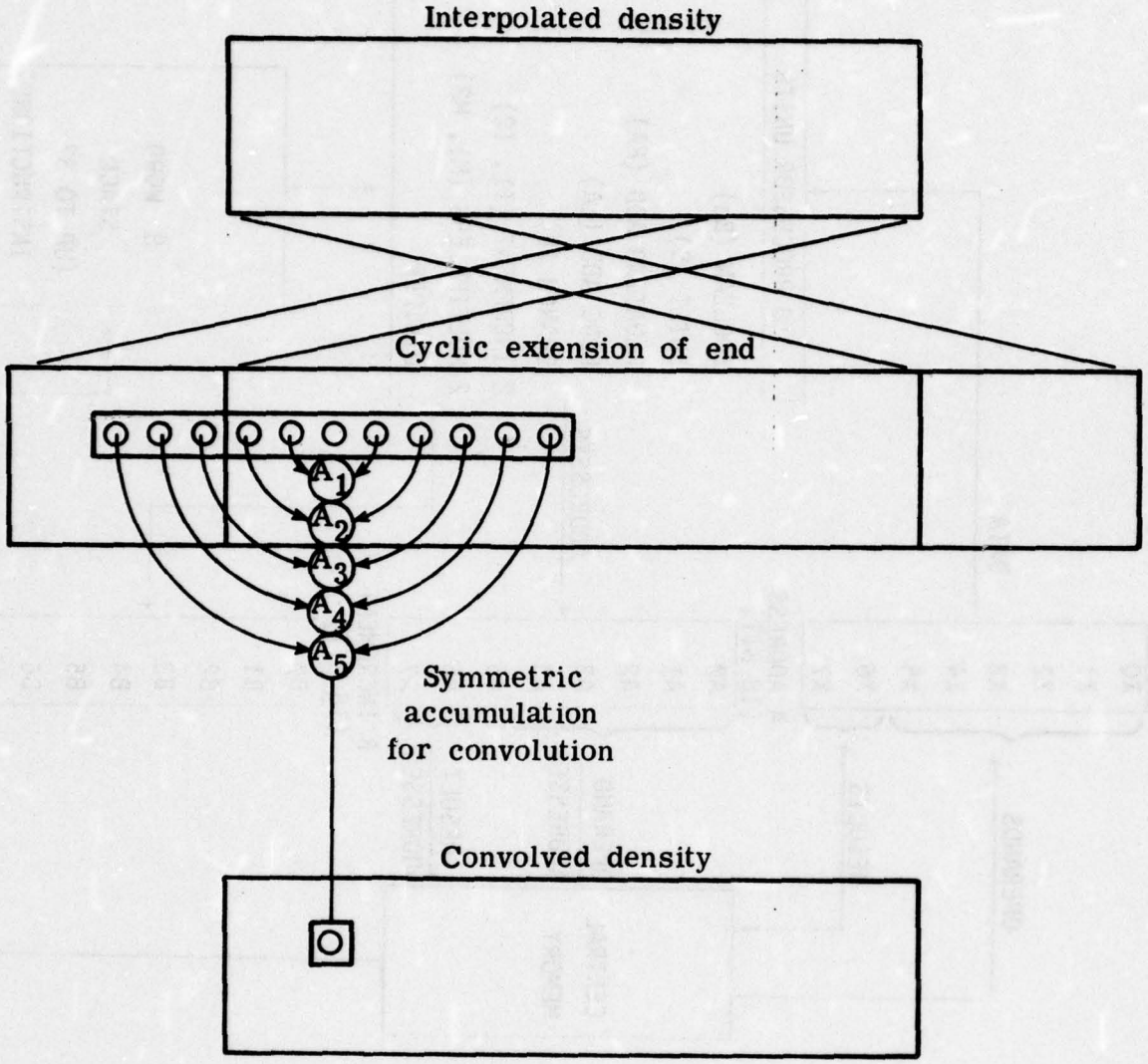


Figure 2.4.- Convolved density.

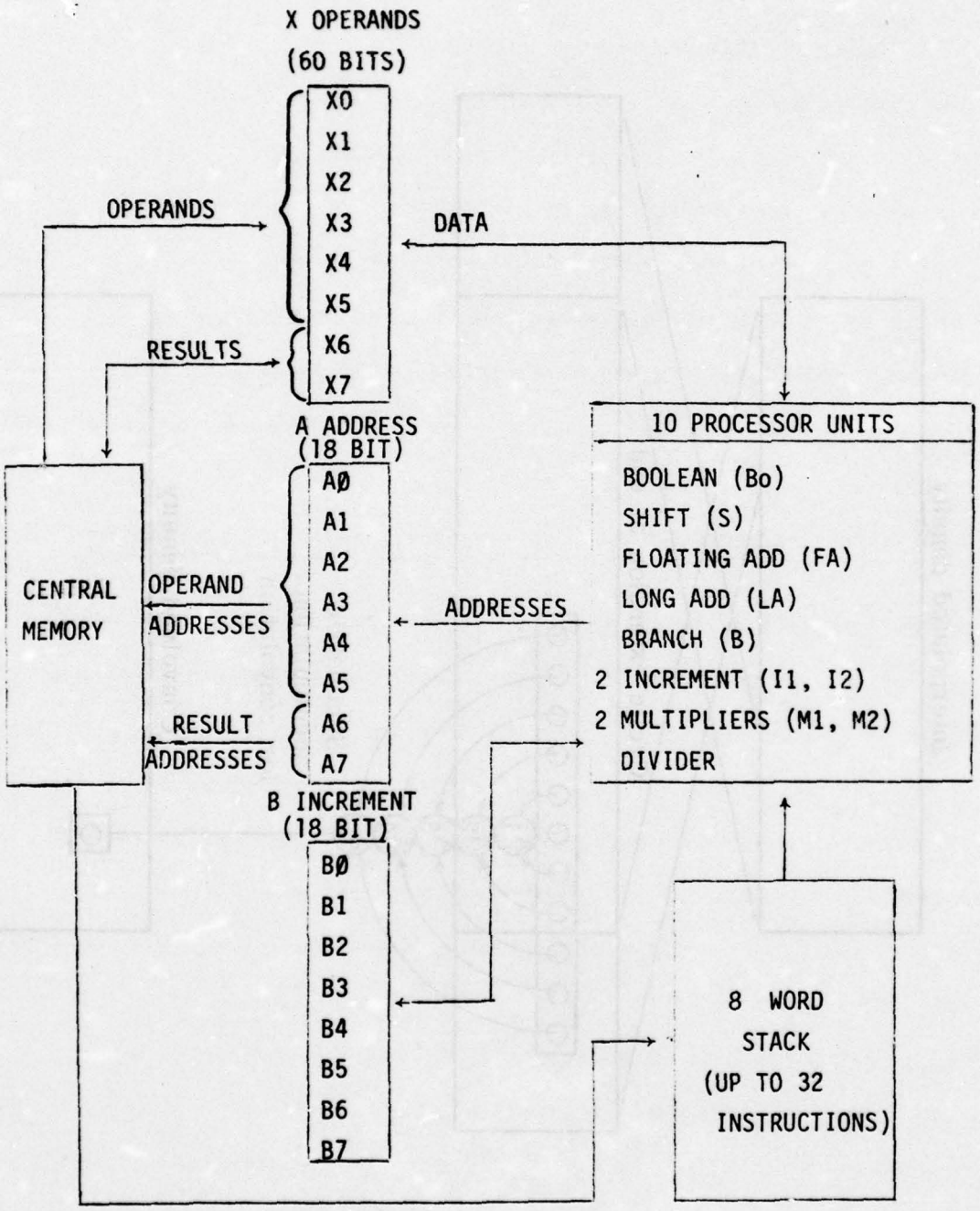


Figure 2.5. CDC 6600 CPU Architecture

TABLE 2.2 -- STAR PROGRAM BREAKDOWN

Operations	% of Total	Cycles Start-ups	Cycles Compute	Cycles Required	
Vector Arithmetic	63.4	5174	76793	78.5%	
12 Multiplies	33.3	1908	41152	70.1%	
46 Adds	26.4	3266	30861	86.4%	
1 Exponential	3.7	-	4780	100.0%	
Vector Rearrangement	31.3	1233	39292	83.7%	
1 Vector Transfer	5.8	1001	6464	68.0%	
2 Indexed Transfers (Block Lengths 32 & 33)	22.1	144	28480	100.0%	
1 Vector Compress	3.4	88	4348	0%	
Scalar Arithmetic	-	-	79	100.0%	
1 Divide			46		
3 Adds			33		
Subroutine Overhead	4.1	-	5142	0%	
3 Calls					
Miscellaneous	1.2	-	1537	100.0%	
Memory Conflicts, etc.					
TOTAL		6407	122843	=>	5.17msec
		5%	95%		
Minimum Achievable		6224	111296	=>	4.70msec

the appropriate address; writes are obtained by loading A6-A7. The B Registers are used for incrementing and address computation.

The breakdown of the computations for the CDC 6600 is shown in Table 2.3. Note that the major overhead is for reading and writing and miscellaneous waiting. It is interesting to note also that the ratio of multiply rates between STAR and 6600 is 25 to 1, while the add rate ratio is 17.5 (including normalization on 6600). Thus, for arithmetic alone, STAR would be expected to be 21 times faster than the 6600 on this problem. The achieved speed-up of 29 is explained by the slightly lower overhead on STAR (36.6% versus 52.9% for 6600).

2.8 Implications for Future Synthesis of Higher Dimensional Filters

The main purpose of this study was to measure the speed-up achievable by parallel and pipeline machines, in order to determine the feasibility of realizing 3 and 4 state dimensional nonlinear filters. In a later section of this paper we describe the fairly natural extension of our 2-state dimensional filter STAR 100 program to a 3-state dimensional problem of estimating phase, phase rate and amplitude. One outgrowth of the software development was a determination of machine structure limitations for natural synthesis of higher dimensional filters. Basically, we found that Illiac IV was limited to the two-dimensional problem because of limited P.E. memory and the availability of only 64 PE's, and most importantly, because of the increase of program complexity and overhead with dimension. The STAR 100 program also becomes complex to program and overhead prone, but at a somewhat higher dimension with the basic limitation that the total number of grid points be

TABLE 2.3-- CDC 6600 PROGRAM BREAKDOWN

Function (% Cycles)	Operations: Multiplies	Adds	Divisions	Reads	Writes, Exp.		Notes
Sensor Update (~0)	2m	m	0	3m	m	m	Not in stack (extn. refs.)
Interpolation (~16%)	mn	2mn	0	4mn+3n	mn	0	Outer Loop not in stack
Expansion (~0)	0	0	0	10m	10m	0	Instack
Convolution (~74%)	5mn	10mn	0	16mn+15	6mn	0	Instack
Row Sums (~4%)	0	mn-m	0	mn+m	m	0	Instack
Estimates (~0)	3m	3m-3	1	4m+11	3	0	Instack
Normalization (~6%)	mn+m	0	0	mn+2m	mn	0	Instack
Approx. Totals	7mn	13mn	0	22mn	8mn	0	
Approx. Number of Minor Cycles	70mn	91mn	0	66mn*	24mn*	0	

Summary: Minor Cycles (Approx.)			
	Arithmetic:	161mn	47.1%
	Read/Write:	90mn	26.3%
	Waiting:	91mn	26.6%
	Measured:	342mn	100.0%

(m = 32, n = 128 for test case) Measured Time = 140 msec/est.

*To some extent these cycles are concurrent, but the total of read/write and waiting of 52.9% is accurate.

less than 65K, the maximum length of vector for Star vector operations, as we show later, a 3-state problem is feasible on Star.

The implications for future machine design suitable for our problem are a pipeline machine which can compute with indexed vectors and with reduced operation cycle time.

3. TIME FACTORS FOR 2-DIMENSIONAL PHASE DEMODULATION

In [4] a complete description of the two-state dimensional phase demodulation problem is given as well as numerous numerical results and a listing of the cyclic nonlinear filtering program. Because we wished to compare the Illiac, Star and 6600 directly, the grid of 32×128 was chosen, i.e., 32 subdivisions in phase and 128 subdivisions in phase rate, to represent JN the conditional density of phase and phase rate given the observations. This grid was natural for the Illiac as it has 64 parallel channels so that the integral equation

$$J_n(x,y) = \frac{D_n(x)}{k} \int_{-\pi/\Delta}^{\pi/\Delta} S(y-\xi)\, J_{n-1}(x-\xi\Delta,\xi)\, d\xi \qquad (3.1)$$

$$y \in (-\pi/\Delta,\pi/\Delta) \quad x \in (-\pi,\pi)$$

with the values of $J_n(x_{i\cdot\cdot})$ could be found in two passes; the first simultaneously giving $\{J_n(x_i, y_j)\}_{j=1\ldots64}$ and the second giving simultaneously $\{J_n(x_i, y_j)\}_{j=65\ldots128}$ and then repeated for each i. We knew before that this mesh was fine enough for accuracy. The serial cyclic point mass filter with interpolation required .700 seconds per estimate.

The Illiac on the other hand produced estimates every .0308 second. The Illiac was not run at full capacity for our runs and we were unable to run it under the fastest possible conditions because other jobs had tied up the machine in August, so that our timing should be looked at as an upper bound to actual performance , clearly far away from the predicted theoretical performance. It is unclear whether in the production mode Illiac can achieve anywhere near theoretical performance, i.e., 64 times faster than CDC 6600 on a full parallel job. It may be that Illiac hardware cannot be made to operate at its theoretical speeds.

As described in the previous section for the Star 100 program, the density was carried as a $32 \times 128 = 4096$ component vector and the convolution developed as a sum of scalar times translates of this vector. This was done in order to take advantage of the pipeline speed of Star. At first with a direct Star Fortran program, a rate of .0118 second per estimate was obtained. By examination of the assembly language version of the Star program and timing each element, we replaced certain Star Fortran calls by more efficient assembly language routines, e.g, VGATHER and VSUM were replaced. After this was done, we achieved a rate of .0049 second per estimate. From this latter number it became clear that the serial 6600 program was not efficient. We were exceeding the factor of 25 theoretical speed ratio of Star to the 6600. This ratio is the ratio of Star multiplication time to 6600 multiply time. A new program, Starrun, was written which was the serial version of the Star program. Starrun achieved .178 second per estimate with the FTN, OPT = 2, Level 410 Fortran compiler. The FTN 4.5 compiler produces significantly different and faster assembly language

programs than the earlier optimizing compilers. By elimination of some multiplications and arranging things so that everything is in the stack, the Starrun program was made to run at .14 second per estimate. The following table gives the timing of our programs on various machines. *With considerable effort in each case devoted to generating fast code:*

Machines	Milliseconds Per Estimate	Compiler
CDC 6600	137.0	FTN 4.5 Level 410 OPT=2
STAR 100	4.7	Assemby Language Optimization
IBM 370-168	130.0	H Compiler
ILLIAC IV	30.8	Glypnir Language, Assembly Listing Optimization
PDP 11-70	870.0	FORTRAN 4 plus
CDC 7600	25	FTN 4.5 Level 410 OPT=2

In order to get a feel for the relative speeds of the Star and the 6600, the ratio of multiply times, when the pipe on the Star is filled is 25 to 1, i.e., 6600 takes 10 minor cycles at 100 nano seconds per cycle, while a multiply on Star takes 1 cycle at 40 nanoseconds per cycle. However, the 6600 has several look-ahead features as well as a stack which can speed up the processing. The FTN, OPT=2 compiler seeks to put all loops in the stack and to fill all the registers so that the ratio we have found 28.57 to 1 is very reasonable for a problem such as ours which is structured so that vector operations can be done most of the time on Star. Of course, using Star for a problem which does not have parallel structure would be wasteful as one would do as well or better on the 7600.

4. BENCHMARK APPLICATION FOR DIFFERENT APPROXIMATIONS

It is very popular in engineering studies to come up with a compromise realization which approaches the accuracy of the optimum nonlinear filter with tremendous amount of computation savings. Without the existence of an accurate optimum scheme, which serves as a benchmark against which any other approximation schemes can be judged, little can be done. The development of such approximations is possible when the exact performance and amount of computations is known. Then and only then, the engineering compromise can be studied intelligently.

One of these compromise studies has recently been published--see [14]--as an approximation for the optimum two-dimensional phase demodulation. It can be generalized to three-dimensional phase-amplitude demodulation problems. This approximation is based on fitting the probability density function at each time step to a Gaussian distribution which produces the optimum first and second moments for the same error criterion. This means that the update is accurate and required only for the first and second moments. In order to be able to investigate the extension of this Gaussian approximation to the three-dimensional problem, it is essential to study very carefully the behavior of the optimum filter and establish a performance benchmark. This 3-dimensional problem was neither analyzed in the optimum way nor simulated on any digital computer before because of its size and complexity. A first attempt was carried out to realize the optimum demodulator on the CYBER 175 and CDC STAR-100 in order to establish boundary limits on the size of the program storage and the computation time.

The mathematical model, optimum filter and preliminary results for the 3-dimensional problem will be discussed in the following secions.

5. INTRODUCTION TO 3D-PROBLEM AND PRELIMINARY RESULTS

Achieving a good speed-up factor for the two-dimensional phase demodulation problem was the signal to move one step ahead and do the three-dimensional problem (phase, phase rate, and amplitude demodulation).

5.1 Mathematical Model

The three-dimensional state vector consists of phase x, phase rate y with additive white Gaussian noise u and amplitude A corrupted with another independent white Gaussian noise v additive to the amplitude mean value.

$$x_n = x_{n-1} + y_{n-1}\,\Delta$$

$$y_n = y_{n-1} + u_{n-1}$$

$$A_n = \alpha(a - A_{n-1})\Delta + A_{n-1} + v_{n-1}$$

where Δ is the time step

α is the damping constant

a is the amplitude mean value.

$$E\,u_n = 0$$

$$E\,u_n^2 = q_1\Delta$$

$$E\,v_n = 0$$

$$E\,v_n^2 = q_2\Delta$$

$$E\,x_o = 0$$

$$E\,x_o^2 = c_1\sqrt{2}\;q_1^{1/4}\,r^{3/4}$$

$$E\,y_o = 0$$

$$E\, y_o^2 = \sqrt{2}\, q_1^{3/4}\, r^{1/4}$$

$$E\, A_o = a$$

$$E\, A_o^2 = \frac{c_2\, q_2}{2\alpha}$$

The observation z is modelled in the following way;

$$z_n^1 = A_n \cos x_n + w_n^1$$

$$z_n^2 = A_n \sin x_n + w_n^2$$

$w_n^{1,2}$ is white Gaussian noise independent of u_n, v_n with $E\, w_n^{1,2} = 0$

$$E(w_n^{1,2})^2 = \frac{r}{\Delta}$$

Δ is chosen to be the same as in the two-dimensional problem, i.e.,

$$\Delta = .1 \left[\sqrt{2}\, \left(\frac{r}{q_1}\right)^{1/4}\right]$$

c_1 is chosen to be 2.19 (same as 2-dimensional problem)

c_2 is chosen to be 4

$\alpha = 1$

$a = 1$

$q_2 = .01$

5.2 Optimum Filter

The optimum filter realization is carried out by a recursive update of the density J_n using the discrete form of the presentation theorem --see [4].

$$J_n(x,y,A) = K_n\, S_n(x,A) \int_{-\infty}^{\infty} \int_{-\pi/\Delta}^{\pi/\Delta} a_1(y-z)\, a_2(A-\beta\eta-a\alpha\Delta) J_{n-1}(x-z\Delta,z,\eta)\,dz\,d\eta$$

where $S_n(x,A) = \text{Exp}\left[\frac{\Delta}{r}\left[A(z_n^1 \cos x + z_n^2 \sin x) - \frac{A^2}{2}\right]\right]$

$$a_1(y-z) = \sum_{k=-\infty}^{\infty} \text{Exp}\left[-\frac{1}{2q_1\Delta}\left(y - z + \frac{2\pi k}{\Delta}\right)^2\right]$$

$$a_2(A-\beta\eta-a\alpha\Delta) = \text{Exp}\left[-\frac{1}{2q_2\Delta}(A-\beta\eta-a\alpha\Delta)^2\right]$$

$$\beta = 1 - a\alpha\Delta$$

K_n is normalizing constant chosen so that

$$\int_{-\infty}^{\infty} \int_{-\pi/\Delta}^{\pi/\Delta} \int_{-\pi}^{\pi} J_n(x, y, A)\, d\,x\, dy\, dA = 1 \quad .$$

5.3 Optimum Estimates

The phase estimates are produced in the same manner as in two-dimensional problems. They are obtained by minimizing the error function $L_1(e) = 2(1 - \cos e)$

$$x_n^* = \arg \int_{-\infty}^{\infty} \int_{-\pi/\Delta}^{\pi/\Delta} \int_{-\pi}^{\pi} e^{ix} J_n(x, y, A)\, dx\, dy\, dA \quad .$$

The amplitude estimates are produced by simply minimizing the quadratic loss function $L_2(e) = e^2$, leading to

$$A_n^* = \int_{-\infty}^{\infty} \int_{-\pi/\Delta}^{\pi/\Delta} \int_{-\pi}^{\pi} A\, J_n(x, y, A)\, dx\, dy\, dA \quad .$$

5.4 Simulation

A standard Fortran program was written to simulate the model described in 5.1 and produce optimum density and estimates as described in 5.2 and 5.3 respectively. The random number generator used is the same as in the two-dimensional problem. This program was debugged and executed on the next fastest computer available at NASA Langley --CYBER 175, which is faster than the CDC 6600 by a factor of approximately 3. Then a final program was written for the STAR which produced output which checked with the C175 results.

5.4.1 CYBER 175 Program -- A point mass realization of the conditional density was simulated by choosing 16×96×16 masses located inside a box representing the three axes, the phase, phase rate and amplitude, respectively (see Figure 5.1). The 16×96 masses for the phase and phase rate are located

inside the rectangle bounded by $(-\pi,\pi]$ in phase and $(-\frac{\pi}{\Delta}, \frac{\pi}{\Delta}]$ in phase rate. The grid in the amplitude direction is floating according to the location of the prior estimate position. For every new amplitude estimate, the whole box will be centered on the amplitude estimate with eight masses equally distributed above the center and the same number below. The mesh size between the amplitude masses is chosen to be equal to half the square root of the variance of the estimate.

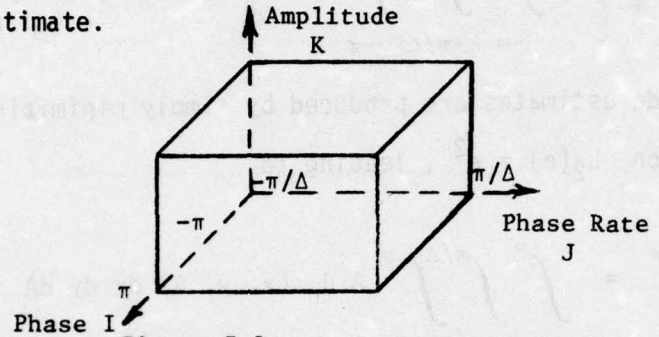


Figure 5.1

The reason we chose the number of discrete masses to be 16×96×16 is that the maximum vector length for STAR operation is limited to 65K as well as storage limitations on the CYBER. Because we know from the two-dimensional problem that 16×96 masses were not accurate enough for density representation the purpose of the 16×96×16 scheme is just to debug the program on the CYBER and try to produce the same results on the STAR later. This way, we can have a valid comparison for the speed of the STAR as compared to other computers for the same amount of computation.

The main structure of the program follows essentially the original program for the two-dimensional phase demodulator. The interpolative and scrambling matrices were precomputed once and for all, and called out inside the mass convolution loop for each iteration. Because of the choice of the dynamic model for the phase rate, $a_1(y - z)$ is symmetric and constant. The cut-off limit for the significant terms of $a_1(y-z)$ is determined by a radius

of length = $\sqrt{50\ q_1 \Delta}$. This is not the case for the amplitude dynamics which does not have a symmetry because of the damping factor and all the sixteen terms have to be computed for every iteration. When this program was executed, it produced estimates every 12 seconds.

5.4.2 CDC STAR-100 Program-- The standard FORTRAN program in 5.4.1 was changed to adapt to the vector notations for the STAR. The three-dimensional array was swept in the right order I (phase) - J (phase rate) - K (amplitude). The main steps of the program follow the same structure of the two-dimensional STAR program, except for the added third dimension as follows:

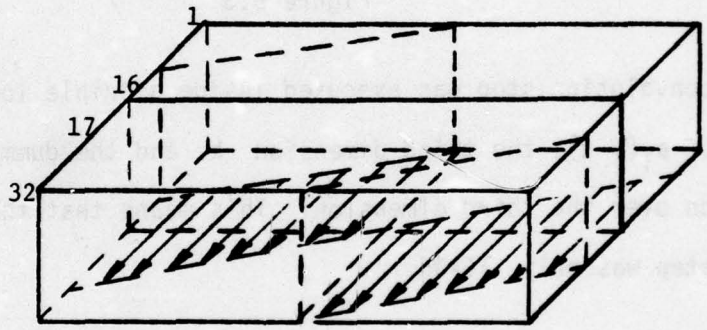


Figure 5.2

(i) In order to perform the scrambling according to the rule (I-J) inside the convolution in the most efficient way, a block mapping VXTOD is used with length 17 for 96×16 blocks--see Figure 5.2. Of course, that required an initial step of copying the box on itself by using the 'MERGE' COMMAND and carrying a copy of the three axis array for the density masses.

(ii) The interpolation between two successive rows was carried on the box of size 17×96×16.

(iii) 'COMPRESS' COMMAND was finally used to get rid of the extra 17^{th} element and to bring the box into its normal size 16×96×16.

(iv) At this step the box was expanded in J direction far enough to cover the maximum number of terms of $a_1(y-z)$ as shown in Figure 5.3.

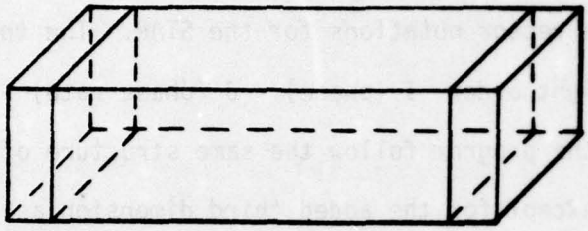


Figure 5.3

(v) The convolution step was executed inside a triple loop for the number of terms of $a_1(y-z)$, the third dimension k and the dummy variable for integration over the third dimension. This means that the length of vector for each step was only 16×96.

(vi) Multiplication by the sensor terms was done in one vector operation of length 24,576.

(vii) Normalization constant was obtained by folding the density successively on itself until it collapsed to single element.

(viii) The density then was multiplied by the inverse of the constant and transferred to the old density for new iteration.

(ix) This normalized density is used for producing estimates for phase and amplitude. It took 89 milliseconds cpu time for every iteration to produce the same results as on the CYBER 175. This present speed factor between the STAR and CYBER will be cut down drastically after converting the

standard FORTRAN program for the CYBER into a serial version in an analogous way to the STAR program where all the arrays are treated as single vectors.

6. EXTENSIONS AND FUTURE RESEARCH

As is clear from equation (3.1), the computational problem of realization of nonlinear filters in general consists of two tasks; first, the passage from the filter density to the one-step predictor density, a convolution task; and secondly, using the new piece of data to construct from the 1-step predictor density the next filter density. Independently of how the densities are finitely represented, these two partitions of the density update are always present. The convolution task is time consuming when done serially and calls for parallel or pipeline implementation. In general, strictly parallel devices are exhorbitantly expensive as dimension and/or griding increase.

Currently, we are investigating optical realization of the convolution task for two state dimensional problems with Drs. Steier and Sawchuck of the University of Southern California. The problem here is one of accuracy and loop closure. We hope to achieve with the optical methods the accuracy obtained with the hybrid realization. Another approach for the convolution task would be to map space on the line--see [9]--and achieve multi-dimension convolutions by using mega-hertz surface wave devices to realize single dimension convolutions; again, here the problem of loop closure is important and unsolved. However, this latter method is not confined to two-dimensions as the optical approach is.

Probably the most promising methods in terms of high-speed, low-cost systems are contemporary mini-computers used with special floating point array processors, with the convolution task essentially done in the array processor and the mini-computer used to accomplish the second task.

An example of such an array processor with theoretical floating point arithmetic speeds only 2.78 slower than the STAR 100 is the AP-120B of Floating Point Systems Inc. We plan to program and evaluate such a system in the course of the next year. Most probably, special purpose digital pipeline devices offer the most promise when accuracy is important, while optical and other analog devices would be useful for filtering problems when accuracy is not that important.

Another problem which merits attention is what are the interactions of other methods of representing the density besides the point mass used here, and the use of pipeline machines. It is not at all clear that either the Spline or Fourier filters can be made to run 30 times faster on the STAR than on the 6600; essentially because the vectors involved are a factor of 10 smaller and start-up times are no longer negligible.

Finally, there are two other contemporary machines which may offer interesting possibilities for producing fast nonlinear filters; they are the Cray machine and a vector machine produced by Texas Instruments. We hope to be able to report on these in the course of the year.

ACKNOWLEDGMENTS

We received help from many people which made this comparison study possible. At I.A.C., Alan Birholtz was helpful in getting us on the Illiac system and Jerry Marin for extended programming help. At ICASE Jim Ortega, Bob Voigt provided helpful discussions. John Knight, Rudine Smith, Everett Johnson, and J. Lambiotte of the Langley Analysis and Computation Division provided significant help in algorithm development. Humberto Torres helped us with getting turn-around and learning the system.

The month of June residence at ICASE was supported for Bucy by ICASE, for Senne by Lincoln Laboratory and for Youssef by Lockheed. Help with the CDC 6600 program and systems questions was given us by Herb Spies of Eglin Air Force Base. Colonel W. J. Rabe of AFOSR was responsible for getting time for us on the Illiac and on the 6600 at Eglin Air Force Base.

REFERENCES

[1] R. S. Bucy and K. D. Senne, "Digital Synthesis of Nonlinear Filters," Automática, 7 (1971), 287-298.

[2] R. S. Bucy, M. J. Merritt, and D. S. Miller, "Hybrid Synthesis of the Optimal Discrete Nonlinear Filter," Stochastics, 1 (1974), 151-211.

[3] L. Basañez, P. Brunet, R. S. Bucy, R. Huber, D. S. Miller, and J. Pagés, "A Hybrid Computer Optimal Filter," Proceedings of the Sixth Symposium on Nonlinear Estimation Theory and Its Applications, (1975), San Diego, California.

[4] R. S. Bucy, C. Hecht, K. D. Senne, "An Engineer's Guide to Building Nonlinear Filters," Final Report SRL-TR-72-0004, Project 7904, Frank J. Seller Research Laboratory, USAF Academy, Colorado 80840 (1972).

[5] L. Basañez, P. Brunet, R. S. Bucy, R. Huber, D. S. Miller and J. Pagés, "Simulation and Implementation of a Hybrid Computer Algorithm for an Optimal Nonlinear Filtering," Proceedings of the 9th Symposium on System Science, Honolulu, Hawaii (1976).

[6] R. S. Bucy, A. J. Mallinckrodt, H. Youssef, "High Speed Convolution of Periodic Functions," to appear J.S.I.A.M. Applied Math.

[7] R. S. Bucy, H. Youssef, "Optimal phase demodulation," I.E.E.E. Trans. Autom. Contr., AC-21, 5, 1976, 732-736.

[8] H. Youssef, "Interpolative Spline Filters," PhD Thesis, Aerospace Eng. Dept., University of Southern California, 1975.

[9] R. S. Bucy, H. Youssef, "Fourier Realization of the Optimal Phase Demodulator," Proc. 4th Symp. on Nonlinear Estimation Theory and Its Applications, San Diego, Western Periodicals, 1973, 34-38.

[10] R. S. Bucy, "Linear and Nonlinear Filtering," Proc. I.E.E.E., 58, 1970, 854-864.

[11] R. S. Bucy, J. Pages, "A Priori Bounds for the Cubic Sensor Problem," to appear.

[12] R. S. Bucy, H. Youssef, "Dependence of the Optimal Phase Demodulator on Statistical Parameters," I.E.E.E. Trans. Autom. Contr., AC-20, 2, 1975, 259-260.

[13] R. S. Bucy, C. Hecht, K. D. Senne, "New Methods for Nonlinear Filtering," Rev. Francais d' Automatique, J-1, 1973, 3-54.

[14] H. M. Youssef, "Suboptimal Phase Demodulation," Proc. of 6th Symp. on Nonlinear Estimation and Its Applications, San Diego, 1975.

[15] D. Casseres, "Illiac IV Machine Reference Manual for the Programmer," Inst. for Advanced Computation Doc. #PG-11700-0000-A, June 1975.

[16] STAR-100 Computer System Reference Manual. Control Data Corporation Publication No. 60256000, 1975.

[17] R. S. Bucy, K. D. Senne, H. M. Youssef, "Pipeline Software for Filtering, to appear.

APPENDIX

A. TWO-DIMENSIONAL CDC-6600 PROGRAM

```
C <BUCY>STAR.FOR;3     4-NOV-76 10:01:44    EDIT BY BUCY
      PROGRAM CYCLIC(INPUT=129,OUTPUT=129,TAPE5=INPUT,TAPE6=OUTPUT)
C       DESCRIPTION OF INPUT PARAMETERS
C
C             Y1EST,Y2EST - THE EXPECTED VALUE OF INITIAL POSITION
C             ALP110 - STEADY STATE ERROR VARIANCE IN DECIBELS
C             DELF - THE RATIO OF DELTA TO FILTER TIME CONSTANT
C             Q22C - THE CONTINUOUS DRIVING VARIANCE
C             NUM1,NUM2 - ARE USED IN CYCLIC AND PROBE CNLY AND COUNT T
C                         NUMBER OF PARTITION POINTS IN RECTANGULAR GRI
C             NO2 - THE TOTAL NUMBER OF POINTS (ESTIMATES) IN EACH SAMP
C
C
C       DESCRIPTION OF DATA SET
C
C             DATA MUST BE PUNCHED IN THE FOLLOWING ORDER:
C
C                Y1EST,Y2EST,ALP110,DELF,Q22C,NUM1,NUM2,NO2
C
C             ALL REAL PARAMETERS (Y1EST THRU Q22C) HAVE A 10 SPACE FIE
C             ALL INTEGER PARAMETERS (NUM1 THRU NO2) HAVE A 5 SPACE FIE
C             AND MUST BE RIGHT JUSTIFIED IN THIER RESPECTIVE FIELDS.
C
C
C       COMMENTS
C
C             THE MAIN FLOW THROUGH THE PROGRAM IS GOVERNED BY KOUNT.
C             KOUNT COUNTS THE POINTS IN EACH PATH.  A BLOCK IS A SECTI
C             OF THE PROGRAM THAT HAS NO TRANSFER IN OR OUT EXCEPT
C             THROUGH COMMON.
C
C
C       ***************************************************************
C
      COMMON XDAT(130,5) ,XHAT(2)
      COMMON Q(2,2),PBAR(2,2),PN(2,2),AN(2),F(2,2),PDUMY(2,2),
     *PDUMY2(2,2),PNF(2,2)
      LOGICAL LOW,UP
      COMMON /RN/ NZZZ(3), XNZ(2)
      COMMON /GN/ DZZZ1, JGAUSS, XZZZ(2)
      COMMON /PROB/ PI2,PI,ALP110,DELF,Q22C,Y1EST,Y2EST,
     1   A11,A22,CONST,DELT,FTC,PIDLT,P110,R11,RX,QQ,Q22
C     ***************************  START BLOCK 1  ***********************
    2 CONTINUE
      JGAUSS=0
      Y1EST=0.0
      Y2EST=0.0
      ALP110=-3.00
      DELF=0.1
      Q22C=0.01
      NUM1=33
      NUM2=127
      NO2=130
      NO3=1
      IF(EOF(5)) 2200,5
```

```
5    CONTINUE
     WRITE(6,651) Y1EST,Y2EST,ALP110,DELF,Q22C,NUM1,NUM2,NO2
651  FORMAT(* *,*  CYCLIC INPUT*,4X,5F10.5,3I5)
     P110=10.**(ALP110/10.)
     QQ=Q22C**(.25)
     RX=(P110/(SQRT(2.0)*QQ))**(4.0/3.0)
     FTC=SQRT(2.0)* RX**(.25) /QQ
     DELT=DELF*FTC
     Q22=Q22C*DELT
     P11=RX/DELT
     P220=P110*SQRT(Q22C/RX)
      ISAMP=1
      NSAMP=0
      SUMP1=0.0
      SUMP2=0.0
      CONTINUE
     A11=10.**((ALP110+1.4)/10.)
     A22=     P220
     KOUNT=1
     DELSQ=DELT**2
     PI=3.1415926536
     PI2=2.0*PI
     PIDLT=PI/DELT
     CONST=-2.0*PIDLT*PIDLT/Q22
     PINV=1.0/PI
     PI2DLT=2.0*PIDLT
     U1=NUM1
     U2=NUM2
     XLIM=.75*PI
     LIMNL=0
     LIMKB=0
     Q(1,1)=0.0
     Q(2,2)=Q22
     A=DELT*PINV*SQRT(10.0*Q22)
     IA=A+0.5
     IY2=U2/PI2DLT*SQRT(50.0*Q22) + .5
     CALL NLF(0,0,Z1,Z2,SHAT,CHAT,TNLF)
11   CALL NLF(1,0,Z1,Z2,SHAT,CHAT,TNLF)
     XHAT(1)=Y1EST
     XHAT(2)=Y2EST
     XHAT1=Y1EST
     XHAT2=Y2EST
     PN(1,1)=A11
     PN(2,2)=A22
     PN(1,2)=0.
     PN(2,1)=0.
     R=R11
     DEV1= SQRT(A11)
     DEN=1.0/(PN(1,1)+R)
     F(1,1)=1.0
     F(1,2)=DELT
     F(2,1)=0.
     F(2,2)=1.0
     DEV2= SQRT(A22)
```

```
      DEV3= SQRT(R11)
      CALL GAUSS(JSEED,DEV1,Y1EST,X1)
      KOUNT=1
      XDAT(KOUNT,1)=X1
      CALL GAUSS(JSEED,DEV2,Y2EST,X2)
      CALL GAUSS(JSEED,DEV3,COS(X1),Z1)
      CALL GAUSS(JSEED,DEV3,SIN(X1),Z2)
      DEVQ2= SQRT(Q22)
      R=R11
      WRITE(6,1509)
 205  FORMAT(*0*,8X,*POSIT.*,5X,*POSIT. MOD 2 PI*,2X,*EST. POSIT.*,9X,
     **Z1 AND Z2*,19X,*CYCLIC LOSS*,5X,* K-B EST. AND P11*)
      GO TO 470
C     ****************************  END BLOCK 1  ************************
C     ****************************  START BLOCK 2  **********************
  450 CONTINUE
      X1=X1 + X2*DELT
      XDAT(KOUNT,1)=X1
      CALL GAUSS(JSEED,DEVQ2,X2,X2)
      CALL GAUSS(JSEED,DEV3,COS(X1),Z1)
      CALL GAUSS(JSEED,DEV3,SIN(X1),Z2)
C     ************************  RICCATI EQUATION UPDATE  ****************
      PDUMY(1,1)=(R*(PN(1,1)+2.0*PN(1,2)*DELT)-PN(1,2)**2*DELSQ)*DEN
     *  + PN(2,2)*DELSQ
      PDUMY(1,2)=PN(1,2)*(R-PN(1,2)*DELT)*DEN + PN(2,2)*DELT
      PDUMY(2,2)=-PN(1,2)**2*DEN + PN(2,2) + Q(2,2)
      PN(1,1)=PDUMY(1,1)
      PN(1,2)=PDUMY(1,2)
      PN(2,2)=PDUMY(2,2)
      PN(2,1)=PN(1,2)
      DEN =1.0/(PN(1,1) + R)
C     ****************************  END BLOCK 2  ************************
C     ****************************  START BLOCK 3  **********************
  470 CONTINUE
      CALL NLF(1,1,Z1,Z2,SHAT,CHAT,TNLF)
      WRITE (6,5687) TNLF
 5687 FORMAT(F10.5)
      CXHAT = ATAN2(SHAT,CHAT)
  367 PLOSS=2.0*(1.0-SQRT(SHAT**2+CHAT**2))
      XDAT(KOUNT,2)=CXHAT
      XDAT(KOUNT,3)=PLOSS
      PNF(1,1)=PN(1,1)*R*DEN
      PNF(1,2)=PN(1,2)*R*DEN
      PNF(2,1)=PNF(1,2)
      PNF(2,2)=PN(2,2) - PN(1,2)**2*DEN
C     ****************************  FILTER UPDATE  **********************
      SINF1=SIN(XHAT1)
      COSF1=COS(XHAT1)
      XHAT(1)=XHAT1+DEN*(-PN(1,1)*SINF1*Z1+PN(1,1)*COSF1*Z2)
      XHAT(2)=XHAT2+DEN*(-PN(1,2)*SINF1*Z1+PN(1,2)*COSF1*Z2)
      X1FF=XHAT(1)
   34 CONTINUE
      IF(X1FF.GT.PI) GO TO 88
      IF(Y1FF.LT.-PI) GO TO 89
      GO TO 90
```

```
   88 X1FF=X1FF-PI2
      GO TO 84
   89 X1FF=X1FF+PI2
      GO TO 84
   90 CONTINUE
      IF(ABS(CXHAT).GT.XLIM)LIMNL=LIMNL+1
      IF(ABS(X1FF).GT.XLIM)LIMKB=LIMKB+1
C     ************************  PREDICTOR UPDATE  **********************
      XHAT1=XHAT(1) + DELT*XHAT(2)
      XHAT2=XHAT(2)
      XDAT(KOUNT,4)=XHAT(1)
      XDAT(KOUNT,5)=PNF(1,1)
      X1MOD=X1
  184 CONTINUE
      IF(X1MOD.GT.PI) GO TO 188
      IF(X1MOD.LT.-PI) GO TO 189
      GO TO 190
  188 X1MOD=X1MOD-PI2
      GO TO 184
  189 X1MOD=X1MOD+PI2
      GO TO 184
  190 CONTINUE
      IKBSLP=0
      X1F2=ABS(XHAT(1)-X1)
  339 IF(X1F2.GT.PI) GO TO 340
      GO TO 341
  340 CONTINUE
      X1F2=X1F2-PI2
      IKBSLP = IKBSLP+1
      GO TO 339
  341 CONTINUE
      ERRLF=ABS(X1FF-X1MOD)
      ERRNL=ABS(CXHAT-X1MOD)
      IF(ERRLF.GT.PI) ERRLF=ABS(ERRLF -PI2)
      IF(ERRNL.GT.PI) ERRNL=ABS(ERRNL-PI2)
      ERROF=ABS(X1MOD-CXHAT)
      WRITE(6,201)KOUNT,XDAT(KOUNT,1),X1MOD,XDAT(KOUNT,2),Z1,Z2, (XDAT
     *(KOUNT,I),I=3,5)
  201 FORMAT(*0*,I3,1X,1P3E14.6,4X,1P2E14.6,4X,1P3E14.6 /)
      IF(KOUNT.EQ.NO2) GO TO 505
      KOUNT=KOUNT + 1
      GO TO 450
  505 CONTINUE
      SUMP=0.0
      SUMC=0.0
      DO 1501 I=31,NO2
        XD=ABS(XDAT(I,1)-XDAT(I,2))
 1498   CONTINUE
        IF(XD.GT.PI) GO TO 1499
        GO TO 1500
 1499   XD=XD-PI2
        GO TO 1498
 1500   SUMP=(XD)**2+SUMP
        SUMC=XDAT(I,3)+SUMC
 1501   CONTINUE
```

```
      H=NO2-30
      SUMP=SUMP/H
      SUMC=SUMC/H
        XNSAMP=NSAMP
        XAA=XNSAMP+1.0
      SUMP1=(SUMP+XNSAMP*SUMP1)/XAA
      DSUMP1=ALOG10(SUMP1)*10.
      WRITE(6,1508)
1508  FORMAT(*0*,5X,*NONLINEAR CYCLIC ESTIMATOR*)
      WRITE(6,1511)SUMP1,DSUMP1
1511  FORMAT(*0*,*AVERAGE STATISTICAL VARIANCE =*,1PE13.6, 10X,
     * *AVERAGE COMPUTED VARIANCE =*,1PE13.6//)
      SUMP=0.0
      SUMC=0.0
      DO 1601 I=31,NO2
        XD=ABS(XDAT(I,1)-XDAT(I,4))
1698    CONTINUE
        IF(XD.GT.PI) GO TO 1699
        GO TO 1700
 699    XD=XD-PI2
        GO TO 1698
1700    SUMP=(XD)**2+SUMP
        SUMC=XDAT(I,5)+SUMC
1601    CONTINUE
      SUMP=SUMP/H
      SUMC=SUMC/H
      SUMP2=(SUMP+XNSAMP*SUMP2)/XAA
      DSUMP2=ALOG10(SUMP2)*10.
      WRITE(6,1509)
1509  FORMAT(*0*,5X,*PE-LINEARIZED K-B FILTER*)
      WRITE(6,1511)SUMP2,DSUMP2
      NSAMP=NSAMP+1
      IF(ISAMP.EQ.NO3)GO TO 2200
      ISAMP=ISAMP+1
      GO TO 11
      ****************************  END BLOCK 3  ************************
2200  WRITE(6,2201)
2201  FORMAT (*0*,40X,*NORMAL COMPLETION*)
      STOP
      END
```

```
      SUBROUTINE GAUSS(JS,SD,XM,X)
      DIMENSION  NST(2)
      COMMON /RN/ N1, N2, MC, T1, T2
      COMMON /GN/ TWOPI, J, XR(2)
      IF (J) 10, 10, 20
   10 J=2
      TWOPI=8.*ATAN(1.)
      NST(1)=102943
      NST(2)=185617
      XR(1)=BANF(NST,1)
      GO TO 35
   20 GO TO (30,40), J
   30 J=2
      XR(1)=BANF(NST,0)
   35 XR(2)=BANF(NST,0)
      X1=SQRT(ABS(-2.*ALOG(XR(1))))
      XR(2)=TWOPI*XR(2)
      XR(1)=X1*SIN(XR(2))
      XR(2)=X1*COS(XR(2))
      X=XR(1)*SD+XM
      RETURN
   40 J=1
      X=XR(2)*SD+XM
      RETURN
      END
```



```
      FUNCTION BANF(NS,MODE)
      DIMENSION  NS(2), NC(2)
      COMMON /RN/ N1, N2, MP, T1, T2
      DATA  M1, M2/244734, 158551/
C        MODE=0 TO CONTINUE, OTHERWISE RESTART WITH
C        INTEGER NUMBER NS(1)*2**18+NS(2)
      IF (MODE) 10, 100, 10
   10 N1=NS(1)
      N2=NS(2)
      T1=2.**(-18)
      T2=2.**(-36)
      MP=2**18
  100 DO 200 I=1,2
      GO TO (110,120),I
  110 K=M2*N2
      GO TO 190
  120 K=M1*N2+M2*N1+KD
  190 KD=K/MP
  200 NC(I)=K-KD*MP
      N1=NC(2)
      N2=NC(1)
      XN1=N1
      XN2=N2
      BANF=XN1*T1+XN2*T2
      RETURN
      END
```

```
      SUBROUTINE NLF(MC,SAMP,Z1,Z2,SHAT,CHAT,TNLF)
      INTEGER MC,SAMP
      REAL Z1,Z2,SHAT,CHAT,TNLF
      INTEGER I,I1,JC,J1,J2,K1,K2,KL,KH,NJ1,NJ2,NK1,NK2,NTERM,
     1    NTERM32,NSIZE,NSIZE32
      REAL A11,A22,CL,CNORM,CONST,CR,PI,PIDEL,Q22,SI,T,TT,
     1    Y1EST,Y2EST,TEMP,TEMP1,TEMP2
      REAL IN(8)
      REAL TROW(32)
      REAL COSY(32),SINY(32),SN1(32),S1(32),S2(32),SIGMA(32)
      REAL PSI(128),A(10),DELJ(128)
      INTEGER JNS(4224)
      REAL JN(4096),JN1(4096),J0(4096),JNA(4756)
      COMMON /PROB/ TWOPI,PI,ALP110,DELF,Q22C,Y1EST,Y2EST,
     1    A11,A22,CONST,DEL,FTC,PIDEL,P110,RDEL,RK,QQ,Q22
      COMMON /NLFC/ NC,NT,NTERM,NTERM33,S1,S2,SIGMA,PSI,A,COSY,
     1    DELJ,J0,JNA,JNS,SINY
      EQUIVALENCE (JN1(1),JNA(321))
      IF (SAMP.LE.0) GO TO 100
      SET CLOCK
      T=SECOND(T)
      EVALUATE SENSOR TERMS
      DO 10 I=1,32
   10 SN1(I)=EXP(Z1*S1(I)+Z2*S2(I))
      FORM THE INTERPOLATED JN AND PUT IN JN1
      J1=0
      J2=0
      DO 30 I=1,128
      TEMP=DELJ(I)
      DO 20 J=1,32
      K1=J1+J
      KL=JNS(K1)
      KH=JNS(K1+1)
      TEMP1=JN(KL)
      K2=J2+I
   20 JN1(K2)=TEMP1+TEMP*(JN(KH)-TEMP1)
      J1=J1+33
   30 J2=J2+32
      EXPAND INTERPOLATED MATRIX ON BOTH SIDES
      J1=NJ1
      J2=NJ2
      K1=NK1
      K2=NK2
      DO 40 I=1,NTERM32
      JNA(J1)=JNA(J2)
      JNA(K1)=JNA(K2)
      J1=J1+1
      J2=J2+1
      K1=K1+1
   40 K2=K2+1
      CONVOLUTION
      DO 50 I=1,4096
      J=I+NSIZE32
   50 JN(I)=JNA(J)
```

```
      J1=NSIZE32
      J2=J1
      DO 60 I=1,NTERM
      J1=J1+32
      J2=J2-32
      TEMP=A(I)
      DO 60 J=1,4096
      K1=J1+J
      K2=J2+J
   60 JN(J)=JN(J)+TEMP*(JNA(K1)+JNA(K2))
C     CUMULATE ROW SUMS
      DO 90 I=1,32
      I1=I
      TEMP2=JN(I1)
      DO 70 J=1,127
      I1=I1+32
   70 TEMP2=TEMP2+JN(I1)
   90 TROW(I)=TEMP2
C     ACCUMULATE ESTIMATES AND NORMALIZATION CONSTANT
      CNORM=TROW(1)*SN1(1)
      SHAT=SINY(1)*CNORM
      CHAT=COSY(1)*CNORM
      DO 85 I=2,32
      TEMP2=TROW(I)*SN1(I)
      SHAT=SHAT+SINY(I)*TEMP2
      CHAT=CHAT+COSY(I)*TEMP2
   85 CNORM=CNORM+TEMP2
      CNORM=1.0/CNORM
      SHAT=SHAT*CNORM
      CHAT=CHAT*CNORM
C     TRANSFER NORMALIZED DENSITY
      DO 90 I=1,32
      I1=I
      TEMP2=SN1(I)*CNORM
      DO 90 J=1,129
      JN(I1)=TEMP2*JN(I1)
   90 I1=I1+32
C     TIMEOUT
      TNLF=SECOND(TT)-T
      RETURN
C     INITIALIZE SAMPLE PATH BY TRANSFERING J0 TO JN
  100 IF (MC.LE.0) GO TO 200
      DO 110 I=1,4096
  110 JN(I)=J0(I)
      RETURN
C     GLOBAL INITIALIZATIONS FOR NONLINEAR FILTER
  200 NSIZE=10
      NTERM=64.0*SQRT(50.*Q22)/PIDEL+0.5
      IF (NTERM.GT.NSIZE) NTERM=NSIZE
      NSIZE32=NSIZE*32
      NTERM32=NTERM*32
      NK2=NSIZE32+1
      NJ1=NK2-NTERM32
      NJ2=NSIZE32+4097-NTERM32
      NK1=NSIZE32+4097
```

```
      PHASE VARIABLES
      DO 210 I=1,32
      SIGMA(I)=PI*((2.*I-1.)/32.-1.)
      COSY(I)=COS(SIGMA(I))
      SINY(I)=SIN(SIGMA(I))
      S1(I)=COSY(I)/RDEL
  210 S2(I)=SINY(I)/RDEL
      PHASE RATE VARIABLES
      DO 220 I=1,128
  220 PSI(I)=PIDEL*((2.*I-1.)/128.-1.)
      SETUP THE TRANSFER MATRIX
      DO 240 J=1,128
      J1=(J-1)*32
      J2=(J-1)*33
      DO 230 I=1,32
      I1=J1+1+MOD(46-(J-1)/4+I,32)
      I2=J2+I
  230 JNS(I2)=I1
  240 JNS(J2+33)=JNS(J2+1)
      SETUP THE INTERPOLATION VECTOR
      IN(1)=0.875
      IN(2)=0.625
      IN(3)=0.375
      IN(4)=0.125
      IN(5)=IN(1)
      IN(6)=IN(2)
      IN(7)=IN(3)
      IN(8)=IN(4)
      J=MOD(NTERM,4)
      DO 245 I=1,4
  245 DELJ(I)=IN(I)
      DO 250 I=5,125,4
      DELJ(I)=DELJ(I-4)
      DELJ(I+1)=DELJ(I-3)
      DELJ(I+2)=DELJ(I-2)
  250 DELJ(I+3)=DELJ(I-1)
      EVALUATE CONVOLUTION TERMS A(I)
      DO 280 I=1,NTERM
      TEMP=I/128.
      TEMP=CONST*TEMP*TEMP
      A(I)=0.
      IF (TEMP.GT.-47) A(I)=EXP(TEMP)
  280 CONTINUE
      CONSTRUCT THE A PRIORI DENSITY
      CNORM=1.0/(TWOPI*SQRT(A11*A22))
      CL=-0.5/A22
      SI=-0.5/A11
      DO 290 I=1,32
      CR=SIGMA(I)-Y1EST
      CR=CR*CR*SI
      J1=0
      DO 290 J=1,128
      J2=J1+I
      TEMP=PSI(J)-Y2EST
      JO(J2)=EXP(TEMP*TEMP*CL+CR)*CNORM
  290 J1=J1+32
      RETURN
      END
```

B. TWO-DIMENSIONAL CDC-STAR-100 PROGRAM

```
      PROGRAM MAIN(INPUT,OUTPUT,TAPE5=INPUT,TAPE6=OUTPUT)
C     SAMPLE PATH VARIABLES
      REAL SX1(130),SCHAT(130),SSHAT(130),SX1HATNL(130),SERRNL(130),
     1 TNLF(130),SPLOSSNL(130),SX1HATPL(130),SERRPL(130),SP11PL(130)
C     CUMULATIVE  SAMPLE PATH VARIABLES
      REAL CERRNL(130),CESQNL(130),CEVARNL(130),CDBNL(130),CCSNL(130)
      REAL CERRPL(130),CESQPL(130),CEVARPL(130),CDBPL(130),CCSPL(130)
C     MONTE CARLO SUMMARY STATISTICS
      REAL XERRNL,XESQNL,XEVARNL,XDBNL,XCSNL,
     1     XERRPL,XESQPL,XEVARPL,XDBPL,XCSPL
C     SINGLE SAMPLE VARIABLES
      REAL CHAT,SHAT,X1HAT,P11,X1,Z1,Z2
C     CONSTANTS
      REAL CS(130),DBEPS,EPS(130),TWOPI,ZERO(130),ONE(130),PI2(130)
C     WORKING VARIABLES
      INTEGER I,J,K,L
      REAL T,TEMP(130)
      LOGICAL PATH,CUMPATH
      BIT BT(130)
C     PROBLEM SETUP VARIABLES
      REAL ALP110,DELF,Q22C,Y1EST,Y2EST
      INTEGER NMC,NSAMP,MD,ND
C     DERIVED PROBLEM CONSTANTS
      REAL A11,A22,CONST,DEL,FTC,PI,PIDEL,P110,RDEL,RX,QQ,Q22
C     PROBLEM COMMON
      COMMON /PROB/ TWOPI,PI,ALP110,DELF,Q22C,Y1EST,Y2EST,A11,A22,
     1   CONST,DEL,FTC,PIDEL,P110,RDEL,RX,QQ,Q22,MD,ND
C     SET PRINTOUT CONTROL
      PATH=.TRUE.
      CUMPATH=.TRUE.
C     READ INPUT PARAMETERS
   10 READ (5,5000,END=500) Y1EST,Y2EST,ALP110,DELF,Q22C,NMC,NSAMP,MD
 5000 FORMAT(5E10.4,3I5)
C     COMPUTE THE CONSTANTS
      MD=(MD/2)*2
      IF (MD.LT.20) MD=32
      IF (MD.GT.64) MD=32
      ND=4*MD
      PI=4.0*ATAN(1.0)
      TWOPI=2.0*PI
      P110=10.**(ALP110/10.)
      QQ=Q22C**(.25)
      RX=(P110/(SQRT(2.0)*QQ))**(4.0/3.0)
      FTC=SQRT(2.0)*RX**(.25)/QQ
      DEL=DELF*FTC
      Q22=Q22C*DEL
      PIDEL=PI/DEL
      RDEL=RX/DEL
      A11=10.0**((ALP110+1.4)/10.)
      A22=2.0*P110/(FTC*FTC)
      CONST=-2.0*PIDEL*PIDEL/Q22
      CS(1:130)=0.75*PI
      PI2(1:130)=TWOPI
      ZERO(1:130)=0.
      EPS(1:130)=1.E-50
```

```
      DBEPS=ALOG10(EPS(1))
      ONE(1;130)=1.
C     INITIALIZE CUMULATIVE SAMPLE PATH VARIABLES
      CERRNL(1;130)=0.
      CESQNL(1;130)=0.
      CCSNL(1;130)=0.
      CERRPL(1;130)=0.
      CESQPL(1;130)=0.
      CCSPL(1;130)=0.
C     PRINTOUT PROBLEM PARAMETERS
      WRITE (6,6000) NMC,NSAMP,ALP110,DELF,Q22C,Y1EST,Y2EST,
     1   P110,QQ,RX,FTC,DEL,Q22,PIDEL,RDEL,A11,A22,CONST,CS(1)
 6000 FORMAT(1H1,31X,18HPROBLEM PARAMETERS/
     1   1H0,11X,9HPARAMETER,6X,5HVALUE,11X,9HPARAMETER,
     2   6X,5HVALUE/1H0,14X,3HNMC,9X,I4,14X,5HNSAMP,8X,I4/
     3   13X,6HALP110,3X,E15.8,8X,4HDELF,4X,E15.8/
     4   14X,4HQ22C,4X,E15.8,8X,5HY1EST,3X,E15.8/
     5   14X,5HY2EST,3X,E15.8,8X,4HP110,4X,E15.8/
     6   15X,2HQQ,5X,E15.8,9X,2HRX,5X,E15.8/
     7   15X,3HFTC,4X,E15.8,9X,3HDEL,4X,E15.8/
     8   15X,3HQ22,4X,E15.8,8X,5HPIDEL,3X,E15.8/
     9   14X,4HRDEL,4X,E15.8,9X,3HA11,4X,E15.8/
     A   15X,3HA22,4X,E15.8,8X,5HCONST,3X,E15.8/
     B   15X,2HCS,5X,E15.8)
C     BEGIN THE MONTE CARLO PROCESS
C     INITIALIZATION OF THE SUBROUTINES
      CALL STATE(0,0,X1,Z1,Z2)
      CALL NLF(0,0,0.,0.,SHAT,CHAT,T)
C     MONTE CARLO LOOP
      DO 200 K=1,NMC
C     INITIALIZATION OF SAMPLE PATH VARIABLES
      CALL STATE(K,0,X1,Z1,Z2)
      CALL NLF(K,0,Z1,Z2,SHAT,CHAT,T)
      CALL PLL(K,0,Z1,Z2,X1HAT,P11)
C     SAMPLE PATH LOOP
      DO 100 J=1,NSAMP
      CALL STATE(K,J,X1,Z1,Z2)
      CALL NLF(K,J,Z1,Z2,SHAT,CHAT,T)
      CALL PLL(K,J,Z1,Z2,X1HAT,P11)
C     STORE THE SAMPLE VARIABLES
      SX1(J)=X1
      SSHAT(J)=SHAT
      SCHAT(J)=CHAT
      TNLF(J)=T
      SX1HATPL(J)=X1HAT
  100 SP11PL(J)=P11
C     VECTOR ACCUMULATE THE SAMPLE PATH AVERAGES
      SX1HATNL(1;130)=VATAN2(SSHAT(1;130),SCHAT(1;130);
     1   SX1HATNL(1;130))
      SERRNL(1;130)=SX1(1;130)-SX1HATNL(1;130)
      CALL MOD2PI(SERRNL)
      SPLOSSNL(1;130)=SSHAT(1;130)*SSHAT(1;130)+SCHAT(1;130)*
     1   SCHAT(1;130)
      SPLOSSNL(1;130)=2.0*(1.0-VSQRT(SPLOSSNL(1;130);SPLOSSNL(1;130)))
      CERRNL(1;130)=VAVG(CERRNL(1;130),SERRNL(1;130),K;CERRNL(1;130))
      TEMP(1;130)=SERRNL(1;130)*SERRNL(1;130)
```

```
      CESQNL(1;130)=VAVG(CESQNL(1;130),TEMP(1;130),K;CESQNL(1;130))
      IF (K.LE.1) GO TO 110
      CEVARNL(1;130)=CESQNL(1;130)-CERRNL(1;130)*CERRNL(1;130)
      BT(1;130)=(CEVARNL(1;130).LE.EPS(1;130))
      CEVARNL(1;130)=Q8VMASK(EPS(1;130),CEVARNL(1;130),BT(1;130):
     1   CEVARNL(1;130))
      CDBNL(1;130)=VALOG10(CEVARNL(1;130);CDBNL(1;130))
      CDBNL(1;130)=10.0*CDBNL(1;130)
  110 BT(1;130)=(SERRNL(1;130).GT.CS(1;130))
      TEMP(1;130)=Q8VMASK(ONE(1;130),ZERO(1;130),BT(1;130):
     1   TEMP(1;130))
      CCSNL(1;130)=VAVG(CCSNL(1;130),TEMP(1;130),K;CCSNL(1;130))
      SERRPL(1;130)=SX1(1;130)-SX1HATPL(1;130)
      CALL MOD2PI(SERRPL)
      CERRPL(1;130)=VAVG(CERRPL(1;130),SERRPL(1;130),K;CERRPL(1;130))
      TEMP(1;130)=SERRPL(1;130)*SERRPL(1;130)
      CESQPL(1;130)=VAVG(CESQPL(1;130),TEMP(1;130),K;CESQPL(1;130))
      IF (K.LE.1) GO TO 120
      CEVARPL(1;130)=CESQPL(1;130)-CERRPL(1;130)*CERRPL(1;130)
      BT(1;130)=(CEVARPL(1;130).LE.EPS(1;130))
      CEVARPL(1;130)=Q8VMASK(EPS(1;130),CEVARPL(1;130),BT(1;130):
     1   CEVARPL(1;130))
      CDBPL(1;130)=VALOG10(CEVARPL(1;130);CDBPL(1;130))
      CDBPL(1;130)=10.0*CDBPL(1;130)
  120 BT(1;130)=(SERRPL(1;130).GT.CS(1;130))
      TEMP(1;130)=Q8VMASK(ONE(1;130),ZERO(1;130),BT(1;130):
     1   TEMP(1;130))
      CCSPL(1;130)=VAVG(CCSPL(1;130),TEMP(1;130),K;CCSPL(1;130))
      IF (PATH.AND.(K.EQ.1   )) GO TO 130
      GO TO 200
C     PRINT SAMPLE PATH VARIABLES
  130 WRITE (6,6010)
 6010 FORMAT(1H1,36X,21HSAMPLE PATH VARIABLES/
     1   38X,19H(FIRST SAMPLE PATH)/)
      WRITE (6,6011)
 6011 FORMAT(/18X,9HSX1-      ,6X,
     1   36H            PHASE VARIABLES          /
     2   20X,1HI,13X,11HSX1(I)      ,8X,13HSX1(I+1)      /)
      WRITE(6,6100)((I,SX1(I),SX1(I+1)),I=1,129,2)
      WRITE (6,6012)
 6012 FORMAT(/18X,9HSSHAT-    ,6X,
     1   36H          SIN(SX1) ESTIMATES          /
     2   20X,1HI,13X,11HSSHAT(I)      ,8X,13HSSHAT(I+1)      /)
      WRITE(6,6100)((I,SSHAT(I),SSHAT(I+1)),I=1,129,2)
      WRITE (6,6013)
 6013 FORMAT(/18X,9HSCHAT-      ,6X,
     1   36H          COS(SX1) ESTIMATES          /
     2   20X,1HI,13X,11HSCHAT(I)      ,8X,13HSCHAT(I+1)      /)
      WRITE(6,6100)((I,SCHAT(I),SCHAT(I+1)),I=1,129,2)
      WRITE (6,6014)
 6014 FORMAT(/18X,9HSX1HATNL-      ,6X,
     1   36H           NONLINEAR ESTIMATES          /
     2   20X,1HI,13X,11HSX1HATNL(I)      ,8X,13HSX1HATNL(I+1)      /)
      WRITE(6,6100)((I,SX1HATNL(I),SX1HATNL(I+1)),I=1,129,2)
      WRITE (6,6015)
 6015 FORMAT(/18X,9HSERRNL-      ,6X,
     1   36H            NONLINEAR ERRORS          /
```

```
     2   20X,1HI,13X,11HSERRNL(I)      ,8X,13HSERRNL(I+1)      /)
      WRITE(6,6100)((I,SERRNL(I),SERRNL(I+1)),I=1,129,2)
      WRITE (6,6016)
 6016 FORMAT(/18X,9HSPLOSSNL-      ,6X,
     1   36H          NONLINEAR PLOSS FUNCTION           /
     2   20X,1HI,13X,11HSPLOSSNL(I)      ,8X,13HSPLOSSNL(I+1)      /)
      WRITE(6,6100)((I,SPLOSSNL(I),SPLOSSNL(I+1)),I=1,129,2)
      WRITE (6,6017)
 6017 FORMAT(/18X,9HTNLF-      ,6X,
     1   36H     NONLINEAR EXECUTION TIMES           /
     2   20X,1HI,13X,11HTNLF(I)      ,8X,13HTNLF(I+1)      /)
      WRITE(6,6100)((I,TNLF(I),TNLF(I+1)),I=1,129,2)
      WRITE (6,6018)
 6018 FORMAT(/18X,9HSX1HATPL-      ,6X,
     1   36H       PHASE LOCK LOOP ESTIMATES           /
     2   20X,1HI,13X,11HSX1HATPL(I)      ,8X,13HSX1HATPL(I+1)      /)
      WRITE(6,6100)((I,SX1HATPL(I),SX1HATPL(I+1)),I=1,129,2)
      WRITE (6,6019)
 6019 FORMAT(/18X,9HSERRPL-      ,6X,
     1   36H          PHASE LOCK LOOP ERRORS           /
     2   20X,1HI,13X,11HSERRPL(I)      ,8X,13HSERRPL(I+1)      /)
      WRITE(6,6100)((I,SERRPL(I),SERRPL(I+1)),I=1,129,2)
      WRITE (6,6020)
 6020 FORMAT(/18X,9HSP11PL-      ,6X,
     1   36HP11 FROM PHASE LOCK RICATTI EQUATION          /
     2   20X,1HI,13X,11HSP11PL(I)      ,8X,13HSP11PL(I+1)      /)
      WRITE(6,6100)((I,SP11PL(I),SP11PL(I+1)),I=1,129,2)
  200 CONTINUE
C     PRINT CUMULATIVE SAMPLE PATH VARIABLES
      IF (CUMPATH.AND.(NMC.GT.1)) GO TO 310
      GO TO 400
  310 WRITE (6,6030)
 6030 FORMAT(1H1,30X,32HCUMULATIVE SAMPLE PATH VARIABLES /)
      WRITE (6,6031)
 6031 FORMAT(/18X,9HCERRNL-      ,6X,
     1   36H     CUMULATIVE NONLINEAR ERRORS     /
     2   20X,1HI,13X,11HCERRNL(I)   ,8X,13HCERRNL(I+1)   /)
      WRITE (6,6100)((I,CERRNL(I),CERRNL(I+1)),I=1,129,2)
      WRITE (6,6032)
 6032 FORMAT(/18X,9HCERRPL-      ,6X,
     1   36HCUMULATIVE PHASE LOCK ERRORS          /
     2   20X,1HI,13X,11HCERRPL(I)   ,8X,13HCERRPL(I+1)   /)
      WRITE (6,6100)((I,CERRPL(I),CERRPL(I+1)),I=1,129,2)
      WRITE (6,6033)
 6033 FORMAT(/18X,9HCESQNL-      ,6X,
     1   36HCUMULATIVE NONLINEAR SQUARED ERRORS     /
     2   20X,1HI,13X,11HCESQNL(I)   ,8X,13HCESQNL(I+1)   /)
      WRITE (6,6100)((I,CESQNL(I),CESQNL(I+1)),I=1,129,2)
      WRITE (6,6034)
 6034 FORMAT(/18X,9HCESQPL-      ,6X,
     1   36HCUMULATIVE PHASE LOCK SQUARED ERRORS     /
     2   20X,1HI,13X,11HCESQPL(I)   ,8X,13HCESQPL(I+1)   /)
      WRITE (6,6100)((I,CESQPL(I),CESQPL(I+1)),I=1,129,2)
      WRITE (6,6035)
```

```
 6035 FORMAT(/18X,9HCEVARNL-      ,6X,
     1   36HCUMULATIVE NONLINEAR ERROR VARIANCE    /
     2   20X,1HI,13X,11HCEVARNL(I)   ,8X,13HCEVARNL(I+1)  /)
      WRITE (6,6100)((I,CEVARNL(I),CEVARNL(I+1)),I=1,129,2)
      WRITE (6,6036)
 6036 FORMAT(/18X,9HCEVARPL-      ,6X,
     1   36HCUMULATIVE PHASE LOCK ERROR VARIANCE    /
     2   20X,1HI,13X,11HCEVARPL(I)   ,8X,13HCEVARPL(I+1)  /)
      WRITE (6,6100)((I,CEVARPL(I),CEVARPL(I+1)),I=1,129,2)
      WRITE (6,6037)
 6037 FORMAT(/18X,9HCDBNL-        ,6X,
     1   36HCUMULATIVE NONLINEAR ERROR DECIBELS    /
     2   20X,1HI,13X,11HCDBNL(I)    ,8X,13HCDBNL(I+1)    /)
      WRITE (6,6100)((I,CDBNL(I),CDBNL(I+1)),I=1,129,2)
      WRITE (6,6038)
 6038 FORMAT(/18X,9HCDBPL-        ,6X,
     1   36HCUMULATIVE PHASE LOCK ERROR DECIBELS    /
     2   20X,1HI,13X,11HCDBPL(I)    ,8X,13HCDBPL(I+1)    /)
      WRITE (6,6100)((I,CDBPL(I),CDBPL(I+1)),I=1,129,2)
      WRITE (6,6039)
 6039 FORMAT(/18X,9HCCSNL-        ,6X,
     1   36HCUMULATIVE NONLINEAR CYCLE SLIPS          /
     2   20X,1HI,13X,11HCCSNL(I)    ,8X,13HCCSNL(I+1)  /  /)
      WRITE (6,6100)((I,CCSNL(I),CCSNL(I+1)),I=1,129,2)
      WRITE (6,6040)
 6040 FORMAT(/18X,9HCCSPL-        ,6X,
     1   36HCUMULATIVE PHASE LOCK CYCLE SLIPS    /
     2   20X,1HI,13X,11HCCSPL(I)    ,8X,13HCCSPL(I+1)    /)
      WRITE (6,6100)((I,CCSPL(I),CCSPL(I+1)),I=1,129,2)
C     COMPUTE THE MONTE CARLO AVERGES
  400 IF (NSAMP.LE.30) GO TO 10
      I=NSAMP-30
      T=I
      XERRNL=Q8SSUM(CERRNL(31:I))/T
      XESQNL=Q8SSUM(CESQNL(31:I))/T
      XCSNL=Q8SSUM(CCSNL(31:I))/T
      XERRPL=Q8SSUM(CERRPL(31:I))/T
      XESQPL=Q8SSUM(CESQPL(31:I))/T
      XCSPL=Q8SSUM(CCSPL(31:I))/T
      XEVARNL=XESQNL-XERRNL*XERRNL
      XDBNL=DBEPS
      IF (XEVARNL.GT.EPS(1)) XDBNL=10.0*ALOG10(XEVARNL)
      XEVARPL=XESQPL-XERRPL*XERRPL
      XDBPL=DBEPS
      IF (XEVARPL.GT.EPS(1)) XDBPL=10.*ALOG10(XEVARPL)
C     PRINT THE MONTE CARLO AVERAGES
      WRITE (6,6050) NMC,XERRNL,XERRPL,XESQNL,XESQPL,XEVARNL,
     1   XEVARPL,XDBNL,XDBPL,XCSNL,XCSPL
 6050 FORMAT(1H1,25X,30HMONTE CARLO SUMMARY STATISTICS//
     1   32X,1H(,I4,14H SAMPLE PATHS)//
     2   36X,16HNONLINEAR FILTER,5X,15HPHASE LOCK LOOP//
     3   14X,14H***VARIABLE***,7X,12H***VALUES***,9X,
     4   12H***VALUES***//14X,13HAVERAGE ERROR,9X,
     5   E15.8,6X,E15.8/
     6   10X,21HAVERAGE SQUARED ERROR,5X,E15.8,6X,E15.8/
     7   14X,14HERROR VARIANCE,8X,E15.8,6X,E15.8      /
     8   14X,14HVARIANCE IN DB,8X,E15.8,6X,E15.8/
     9   11X,19HAVERAGE CYCLE SLIPS,6X,E15.8,6X,E15.8)
```

```
6100 FORMAT(20X,I4,8X,E15.8,4X,E15.8)
     GO TO 10
 500 STOP
     END
     SUBROUTINE VPROP(A,I)
     REAL A(16640)
     INTEGER I
     INTEGER MD1,MD2,MD3,MD4,MD5,MD6,MD7,
    1    MD8,MD9,MD10,MD11,MD12,MD13,MD14
     COMMON /CPROP/ MD1,MD2,MD3,MD4,MD5,MD6,MD7,
    1    MD8,MD9,MD10,MD11,MD12,MD13,MD14
     IF (I.GT.0) GO TO 10
     A(MD2;MD1)=A(1;MD1)
     A(MD4;MD3)=A(1;MD3)
  10 A(MD6;MD5)=A(1;MD5)
     A(MD8;MD7)=A(1;MD7)
     A(MD10;MD9)=A(1;MD9)
     A(MD12;MD11)=A(1;MD11)
     A(MD14;MD13)=A(1;MD13)
     RETURN
     END
     SUBROUTINE MOD2PI(A)
     REAL A(130),X(130),Y(130)
     BIT BT(130)
     COMMON /PROB/ TWOPI,PI
     X(1:130)=PI
  10 BT(1:130)=(A(1:130).GT.X(1:130))
     IF (Q8SCNT(BT(1:130)).EQ.0) GO TO 20
     Y(1:130)=A(1:130)-TWOPI
     A(1:130)=Q8VMASK(Y(1:130),A(1:130),BT(1:130):A(1:130))
     GO TO 10
  20 X(1:130)=-PI
  30 BT(1:130)=(A(1:130).LT.X(1:130))
     IF (Q8SCNT(BT(1:130)).EQ.0) RETURN
     Y(1:130)=A(1:130)+TWOPI
     A(1:130)=Q8VMASK(Y(1:130),A(1:130),BT(1:130):A(1:130))
     GO TO 30
     END
     REAL FUNCTION VAVG(AV,X,I;*)
     DESCRIPTOR AV,X,VAVG
     INTEGER I
     REAL XI
     XI=I
     VAVG =((XI-1.)*AV+X)/XI
     RETURN
     END
     REAL FUNCTION RNF(INIT)
     INTEGER INIT,K,KD,M1,M2,N1,N2,NT,MP
     REAL T1,T2
     COMMON /RNUM/ K,KD,M1,M2,N1,N2,NT,MP,T1,T2
     IF (INIT.EQ.0) GO TO 10
     N1=102943
     N2=185617
     N1=244734
     N2=158551
     M1=244734
```

```
      M2=158551
      T1=2.**(-18)
      T2=2.**(-36)
      MP=2**18
   10 K=M2*N2
      KD=K/MP
      NT=K-KD*MP
      K=M1*N2+M2*N1+KD
      KD=K/MP
      N1=K-KD*MP
      N2=NT
      RNF=N1*T1+N2*T2
      RETURN
      END
      REAL FUNCTION GAUSS(INIT,SD,XM)
      INTEGER INIT,I,J
      REAL SD,XM,TWOPI,X,XR1,XR2
      COMMON /GNUM/ I,J,XR2
      COMMON /PROB/ TWOPI
      IF (INIT.EQ.0) GO TO 10
      I=1
      J=1
   10 IF (J.NE.1) GO TO 20
      J=2
      XR1=RNF(I)
      I=0
      XR2=RNF(0)
      X=SQRT(ABS(-2.*ALOG(XR1)))
      XR2=TWOPI*XR2
      XR1=X*SIN(XR2)
      XR2=X*COS(XR2)
      GAUSS=XR1*SD+XM
      RETURN
   20 J=1
      GAUSS=XR2*SD+XM
      RETURN
      END
      SUBROUTINE STATE(MC,SAMP,X1,Z1,Z2)
      INTEGER MC,SAMP,INIT
      REAL X1,Z1,Z2,DEV1,DEV2,DEV3,DEV4,X2
      COMMON /STA/ DEV1,DEV2,DEV3,DEV4,INIT
      COMMON /PROB/ TWOPI,PI,ALP110,DELF,Q22C,Y1EST,Y2EST,A11,A22,
     1   CONST,DEL,FTC,PIDEL,P110,RDEL,RX,QQ,Q22,MD,ND
      IF (MC.NE.0) GO TO 10
      DEV1=SQRT(A11)
      DEV2=SQRT(A22)
      DEV3=SQRT(RDEL)
      DEV4=SQRT(Q22)
      INIT=1
      RETURN
   10 IF (SAMP.GT.0) GO TO 20
      X1=GAUSS(INIT,DEV1,Y1EST)
      INIT=0
      X2=GAUSS(0,DEV2,Y2EST)
      RETURN
```

```
   20 IF (SAMP.LE.1) GO TO 30
      X1=X1+X2*DEL
      X2=GAUSS(0,DEV4,X2)
   30 Z1=GAUSS(0,DEV3,COS(X1))
      Z2=GAUSS(0,DEV3,SIN(X1))
      RETURN
      END
      SUBROUTINE PLL(MC,SAMP,Z1,Z2,X1HAT,PF11)
      INTEGER MC,SAMP
      REAL Z1,Z2,X1HAT,PF11,DEN,SINF1,COSF1,
     1   X2HAT,PD11,PD12,PD22,PN11,PN12,PN22
      COMMON /PLD/ DEN,X2HAT,PN11,PN12,PN22
      COMMON /PROB/ TWOPI,PI,ALP110,DELF,Q22C,Y1EST,Y2EST,A11,A22,
     1   CONST,DEL,FTC,PIDEL,P110,RDEL,RX,Q0,Q22
      IF (MC.LE.0) RETURN
      IF (SAMP.LE.0) GO TO 20
      IF (SAMP.LE.1) GO TO 10
      X1HAT=X1HAT+DEL*X2HAT
      PD11=(RDEL*(PN11+2.0*PN12*DEL)-PN12*PN12
     1   *DEL*DEL)*DEN+PN22*DEL*DEL
      PD12=PN12*(RDEL-PN12*DEL)*DEN+PN22*DEL
      PD22=-PN12*PN12*DEN+PN22+Q22
      PN11=PD11
      PN12=PD12
      PN22=PD22
      DEN=1.0/(PN11+RDEL)
   10 PF11=PN11*RDEL*DEN
      SINF1=SIN(X1HAT)
      COSF1=COS(X1HAT)
      X1HAT=X1HAT+DEN*(-PN11*SINF1*Z1+PN11*COSF1*Z2)
      X2HAT=X2HAT+DEN*(-PN12*SINF1*Z1+PN12*COSF1*Z2)
      RETURN
   20 X1HAT=Y1EST
      X2HAT=Y2EST
      PN11=A11
      PN22=A22
      PN12=0.
      DEN=1.0/(PN11+RDEL)
      RETURN
      END
      SUBROUTINE NLF(MC,SAMP,Z1,Z2,SHAT,CHAT,TNLF)
C
C-----------------------------------------------------------------------
C
C
C        POINTMASS FILTER FOR TWO DIMENSIONAL PHASE ESTIMATION
C        PROGRAMMED BY KENNETH D. SENNE                JUNE 24, 1976
C
C
C-----------------------------------------------------------------------
C
      INTEGER MC,SAMP,MD,ND,MND,MND1
      INTEGER MD1,MD2,MD3,MD4,MD5,MD6,MD7,MD8,MD9,MD10,MD11,MD12,MD13,
     1   MD8,MD9,MD10,MD11,MD12,MD13,MD14
      REAL Z1,Z2,SHAT,CHAT,TNLF
      INTEGER I,I1,J,J1,J2,K,NC,NT,NTERM,NTERM33
```

```
      REAL A11,A22,CL,CNORM,CONST,CR,PI,PIDEL,Q22,T,
     1         TT,Y1EST,Y2EST,TEMP,TEMP1,TEMP2
      REAL S1(64),S2(64),SIGMA(64),PSI(256),A(256)
      REAL IN(8)
      INTEGER DJNS
      INTEGER JNS(16640)
      REAL COSY(16640),DELJ(16640),J0(16640),JN(16640),JN1(16640),
     1   JNA(33280),SINY(16640),SN1(16640)
      DESCRIPTOR DS1,DS2,DSN1,DSN2,DJN,DJNS,DJNA1,DJNA2,DDELJ,DJN1,
     1   DJNAENT,DJNABNT,DJNANC,DJNAJ1,DJNAJ2,DSINY,DCOSY,DSN1A
      COMMON /PROB/ TWOPI,PI,ALP110,DELF,Q22C,Y1EST,Y2EST,A11,A22,
     1   CONST,DEL,FTC,PIDEL,P110,RDEL,RX,QQ,Q22,MD,ND
      COMMON /NLFC/ NC,NT,NTERM,NTERM33,S1,S2,SIGMA,PSI,A,COSY,DELJ,
     1   J0,JN,JNS,SINY
      COMMON /CPROP/ MD1,MD2,MD3,MD4,MD5,MD6,MD7,
     1   MD8,MD9,MD10,MD11,MD12,MD13,MD14
      IF (SAMP.LE.0) GO TO 100
C
C-----------------------------------------------------------------------
C
C        SAMPLE PATH UPDATE TAKES PLACE HERE
C
C
C     SET CLOCK
C
      CALL Q8CLOCKS(T,TT)
C
C     EVALUATE SENSOR TERMS
C
      DSN1=Z1*DS1
      DSN2=Z2*DS2
      DSN1=DSN1+DSN2
      DSN1=VEXP(DSN1:DSN1)
      SN1(MD1)=0.
C
C     PROPAGATE SENSOR TERMS
C
      CALL VPROP(SN1,0)
C
C     SCRAMBLE THE JN TO ORDER FOR J(N+1)
C
      DJNA1=Q8VGATHR(DJN,DJNS:DJNA1)
C
C     FORM THE INTERPOLATED MATRIX
C
      DJN=DJNA2-DJNA1
      DJN1=DDELJ*DJN
      DJNA1=DJNA1+DJN1
C
C     COPY THE END COLUMNS
C
      DJNAENT=DJNABNT
C
C     INITIALIZE CONVOLUTION (A(0)=1)
C
      DJN1=DJNANC
C
```

```
C       CONVOLUTION LOOP
C
        J1=NC
        J2=NC
        DO 10 I=1,NTERM
        J1=J1+MD1
        J2=J2-MD1
        ASSIGN DJNAJ1,JNA(J1;MND)
        ASSIGN DJNAJ2,JNA(J2;MND)
        DJN=DJNAJ1+DJNAJ2
        DJN=A(I)*DJN
   10   DJN1=DJN1+DJN
C
C       MULTIPLY BY SENSOR TERMS
C
        DJN1=DSN1A*DJN1
C
C       GET NORMALIZATION CONSTANT
C
        CNORM=Q8SSUM(DJN1)
        CNORM=1.0/CNORM
C
C       TRANSFER THE NORMALIZED DENSITY
C
        DJN=CNORM*DJN1
C
C       CUMULATE ESTIMATES
C
        DJNA1=DSINY*DJN
        SHAT=Q8SSUM(DJNA1)
        DJNA1=DCOSY*DJN
        CHAT=Q8SSUM(DJNA1)
C
C       TIMEOUT
C
        CALL Q3CLOCKS(TNLF,TT)
        RETURN
C
C-------------------------------------------------------------------------
C
C
C-------------------------------------------------------------------------
C
C          SAMPLE PATH INITIALIZATION TAKES PLACE HERE
C
  100   IF (MC.LE.0) GO TO 200
C
C       TRANSFER THE INITIAL DENSITY FOR NEW SAMPLE PATH
C
        JN(1;MND)=JO(1;MND)
        RETURN
C
C-------------------------------------------------------------------------
```

```
C
C
C-----------------------------------------------------------------------
C
C        GLOBAL INITIALIZATIONS OCCUR HERE FOR THE ENTIRE RUN
C
C
C      DETERMINE THE NUMBER OF CONVOLUTION POINTS
C
  200 NTERM=(ND/2.)*SQRT(50.*Q22)/PIDEL+0.5
      MD1=MD+1
      MD2=MD1+1
      MD3=MD1*2
      MD4=MD3+1
      MD5=MD3*2
      MD6=MD5+1
      MD7=MD5*2
      MD8=MD7+1
      MD9=MD7*2
      MD10=MD9+1
      MD11=MD9*2
      MD12=MD11+1
      MD13=MD11*2
      MD14=MD13+1
      MND=MD1*ND
      MND1=MND+1
      NTERM33=MD1*NTERM
      NT=2*NTERM33
      NC=NTERM33+1
C
C      SET UP THE VECTOR DESCRIPTORS FOR THE UPDATE FUNCTIONS
C
      ASSIGN DS1,S1(1;MD)
      ASSIGN DS2,S2(1;MD)
      ASSIGN DSN1,SN1(1;MD)
      ASSIGN DSN2,JNA(1;MD)
      ASSIGN DJN,JN(1;MND)
      ASSIGN DJNS,JNS(1;MND)
      ASSIGN DJNA1,JNA(1;MND)
      ASSIGN DJNA2,JNA(2;MND)
      ASSIGN DDELJ,DELJ(1;MND)
      ASSIGN DJN1,JN1(1;MND)
      ASSIGN DJNAENT,JNA(MND1;NT)
      ASSIGN DJNABNT,JNA(1;NT)
      ASSIGN DJNANC,JNA(NC;MND)
      ASSIGN DSN1A,SN1(1;MND)
      ASSIGN DSINY,SINY(1;MND)
      ASSIGN DCOSY,COSY(1;MND)
C
C      PHASE VARIABLES
C
      DO 210 I=1,MD
      SIGMA(I)=PI*((2.*I-1.)/MD -1.)
      COSY(I)=COS(SIGMA(I))
      SINY(I)=SIN(SIGMA(I))
      S1(I)=COSY(I)/RDEL
```

```
  210 S2(I)=SINY(I)/RDEL
      SINY(MD1)=0.
      COSY(MD1)=0.
      CALL VPROP(SINY,0)
      CALL VPROP(COSY,0)
C
C     PHASE RATE VARIABLES
C
      DO 220 I=1,ND
  220 PSI(I)=PIDEL*((2.*I-1.)/ND  -1.)
C
C     SETUP THE TRANSFER MATRIX
C
      DO 240 J=1,ND
      J1=MOD(ND-1-NTERM+J,ND)*MD1
      J2=(J-1)*MD1
      DO 230 I=1,MD
      I1=J1+1+MOD(MD+MD/2+32-(135-NTERM+J)/4+I,MD)
      I2=J2+I
  230 JNS(I2)=I1
      J1=J2+MD1
  240 JNS(J1)=JNS(J2+1)
C
C     SETUP INTERPOLATION MATRIX
C
      IN(1)=0.875
      IN(2)=0.625
      IN(3)=0.375
      IN(4)=0.125
      IN(5)=IN(1)
      IN(6)=IN(2)
      IN(7)=IN(3)
      IN(8)=IN(4)
      J=MOD(NTERM,4)
      DO 245 I=1,4
      I1=(I-1)*MD1+1
      J1=I+4-J
      T=IN(J1)
  245 DELJ(I1:MD1)=T
      CALL VPROP(DELJ,1)
C
C     EVALUATE CONVOLUTION TERMS A(I)
C
      DO 280 I=1,NTERM
      T=I
      TT=ND
      TEMP=T/TT
      TEMP=CONST*TEMP*TEMP
      A(I)=0.
      IF (TEMP.GT.-47) A(I)=EXP(TEMP)
  280 CONTINUE
C
C     CONSTRUCT THE A PRIORI DENSITY
C
```

```
      CNORM=1.0/(TWOPI*SQRT(A11*A22))
      CL=-0.5/A22
      SI=-0.5/A11
      DO 290 I=1,MD
      I1=I
      CR=SIGMA(I)-Y1EST
      CR=CR*CR*SI
      DO 290 J=1,ND
      TEMP=PSI(J)-Y2EST
      JO(I1 )=EXP(TEMP*TEMP*CL+CR)*CNORM
  290 I1=I1+MD1
C
C     WRITE OUT PARAMS OF NLF
C
      WRITE (6,6000) MD,ND,MD1,ND
 6000 FORMAT(1HO,26X,27HPOINT MASS NONLINEAR FILTER//1H ,
     1   28X,24HVERSION 2, CODED 6/27/76//1H ,
     2   18X,I3,1HX,I3,25H DENSITIES REPRESENTED BY ,I3,1HX,I3)
      WRITE (6,6001) NTERM,(A(I),I=1,NTERM)
 6001 FORMAT(1H ,33X,7HA(1)-A(,I2,2H) /(1X,5E15.8))
      RETURN
C
C-----------------------------------------------------------------------
C
      END

 0.0        0.0        -3.01      0.1        0.1          200   130    64
 0.0        0.0        -1.79      0.1        0.01         200   130    64
  END OF FILE
```

C. THREE-DIMENSIONAL CDC-STAR-100 PROGRAM

```
      PROGRAM MAIN(INPUT,OUTPUT,PUNCH,TAPE6=OUTPUT,TAPE5=INPUT)
      DIMENSION XDAT(130,10),XHAT(4),Y1(35),Y2(135),
     *EXPON(35),EXDON(17),Y3(17),PNF(3,3)    ,YA(35),EXP33(16,16),
     *Q(3,3),PBAR(3,3),PN(3,3),AN(3),F(3,3),PDUMMY(3,3),PDUMY2(3,3)
      COMMON/GN/JGAUSS,XZZZ(2)
      REAL COSY(24576),SINY(24576),SN1(24576),SN2(1536),JN(24576),
     *JN1(26112),JNA(54272),DELJ(26112),S1(16),S2(16),YB(24576),D(2000)
      INTEGER  JNS(1536),JNF(3072)
      BIT B3(26112)
      DESCRIPTOR DB,KJNF,DJNA,KJNS,DJN1,DC
      ASSIGN DB,JN(1;16)
      ASSIGN KJNF,JNF(1;3072)
      ASSIGN DJNA,JNA(1;49152)
      ASSIGN KJNS,JNS(1;1536)
      ASSIGN DJN1,JN1(1;26112)
      ASSIGN DC,JNA(1;17)
      JGAUSS=0
      READ(5,63)Y3EST,Q33C,ALF,GAM,NUM3
63    FORMAT(4F10.5,I5)
      READ(5,64)Y1EST,Y2EST,ALP110,DELF,Q22C,NUM1,NUM2,NO2,NO3
64    FORMAT(5F10.5,4I5)
      P110=10.**(ALP110/10.)
      QQ=Q22C**(.25)
      RX=(P110/(SQRT(2.0)*QQ))**(4.0/3.0)
      FTC=SQRT(2.0)*RX**(.25)/QQ
      DELT=DELF*FTC
      Q22=Q22C*DELT
      R11=RX/DELT
      P220=P110*SQRT(Q22C/RX)
      ALFD=ALF*DELT
      BET=1.0-ALFD
      A11=10.**((ALP110+GAM)/10.)
      A22=P220
      P330=0.5*Q33C/ALF
      A33=2.0*P330
      Q33=Q33C*DELT
      DEV1=SQRT(A11)
      DEV2=SQRT(A22)
      DEV3=SQRT(R11)
      DEV4=SQRT(A33)
      DEVQ2=SQRT(Q22)
      DEVQ3=SQRT(Q33)
      KOUNT=1
      ISAMP=1
      NSAMP=0
      SUMP1=0.0
      SUMP2=0.0
      SUMP3=0.0
```

```
      DELSQ=DELT**2
      PI=4.*ATAN(1.)
      PI2=2.0*PI
      PIDLT=PI/DELT
      PINV=1.0/PI
      PI2DLT=2.0*PIDLT
      U1=NUM1
      U2=NUM2
      U3=NUM3
      IY2=U2/PI2DLT*SQRT(50.0*Q22)+.5
      NTERM=IY2
      NTERM1=NTERM+1
      NTERM16=NTERM*16
      NC=NTERM16+1
      NT=2*NTERM16
      NTA1536=NT+1536
      R55=0.5/R11
      CL=-0.5/A22
      CM=-0.5/A33
      SI=-0.5/A11
C     ***************GRID Y1,Y2 AND YA ********************************
      EDG1=PI/U1
      EDG2=PIDLT/U2
      DO 40 I=1,NUM1
        X=I-1
        X=X/U1
 40     Y1(I)=-PI+X*PI2+EDG1
      DO 50 I=1,NUM2
        X=I-1
        X=X/U2
 50     Y2(I)=-PIDLT+X*PI2DLT+EDG2
      DO 51 I=1,IY2
        X=I
        X=X/U2
 51     Y3(I)=X
      DO 55 I=1,NUM1
        COSY(I)=COS(Y1(I))
 55     SINY(I)=SIN(Y1(I))
      S1(1;16)=COSY(1;16)/R11
      S2(1;16)=SINY(1;16)/R11
      CALL VPROP(SINY,1)
      CALL VPROP(COSY,1)
      XP=0.5*SQRT(A33)
      YY3=(NUM3-1.0)/2.0+1.0
      IY3=YY3
      DO 60 I=1,NUM3
        XX=I-YY3
 60     YA(I)=Y3EST+XP*XX
C     ******************** DYNAMIC EXPONENTIALS ************************
      IF(IY2.EQ.0)GO TO 153
      DO 150 I=1,IY2
 150    EXDON(I)=EXP(-0.5/Q22*(Y3(I)**2))
      DO 152 I=1,IY2
        EXPON(I)=EXDON(IY2+1-I)
 152    EXPON(IY2+1+I)=EXDON(I)
```

```
153     EXPON(IY2+1)=0.5
        IYY=2*IY2+1
        LTERM2=0
        DO 720 I=1,16
        DO 720 K=1,16
        XNUM=(K-I)*XP+ALFD*(YA(I)-1.)
        XNUM=-0.5/Q33*XNUM**2
        EXP33(I,K)=0.0
        IF(XNUM.LT.-27.)GO TO 720
        EXP33(I,K)=EXP(XNUM)
720     CONTINUE
        LTERM=0
        LTERM16=LTERM*16
        LTERM1=LTERM+1
        LC=LTERM16+1
        NS=NTA1536*LTERM+NC
        I=0
        DO 339 K=1,16
        DO 339 N=1,16
        DO 339 J=NTERM1,IYY
        I=I+1
339     D(I)=EXPON(J)*EXP33(N,K)
C       ***************************** INITIAL DENSITY **********************
        IJK=0
        DO 160 K=1,16
        ZZZ=CM*(YA(K)-Y3EST)**2
        DO 160 J=1,96
        YYY=ZZZ+CL*(Y2(J)-Y2EST)**2
        DO 160 I=1,16
        IJK=IJK+1
        XXX=YYY+SI*(Y1(I)-Y1EST)**2
        IF(XXX.LT.-27.)GO TO 159
        JN(IJK)=EXP(XXX)
        GO TO 160
159     JN(IJK)=0.0
160     CONTINUE
C       ******************* INTEGER ARRANGEMENT ****************************
        DO  225 I=1,26112
225     B3(I)=B'1'
        I=0
        DO 300 K=1,16
        DO 300 J=1,96
        I=I+17
300     B3(I)=B'0'
        DO 320 K=1,3071,2
        JNF(K)=(K-1)*8
320     JNF(K+1)=JNF(K)
        I=0
        J1=0
        DO  332 K=1,16
        DO 332 J=1,96
        I=I+1
        I1=J1+MOD(23-(J-1)/6,16)
        JNS(I)=I1
332     J1=J1+32
        DELJ(1:17)=11./12.
        DELJ(18:17)=0.75
```

```
      DELJ(35;17)=7./12.
      DELJ(52;17)=5./12.
      DELJ(69;17)=0.25
      DELJ(86;17)=1./12.
      CALL VPROP1(DELJ)
 11   CONTINUE
      KOUNT=1
      CALL GAUSS(JSEED,DEV1,Y1EST,X1)
      XDAT(KOUNT,1)=X1
      CALL GAUSS(JSEED,DEV2,Y2EST,X2)
      CALL GAUSS(JSEED,DEV4,Y3EST,X3)
      ACOS=X3*COS(X1)
      ASIN=X3*SIN(X1)
      CALL GAUSS(JSEED,DEV3,ACOS,Z1)
      CALL GAUSS(JSEED,DEV3,ASIN,Z2)
      GO TO 470
 450  CONTINUE
      X1=X1+X2*DELT
      XDAT(KOUNT,1)=X1
      CALL GAUSS(JSEED,DEVQ2,X2,X2)
      X3=X3*BET+ALFD
      CALL GAUSS(JSEED,DEVQ3,X3,X3)
      ACOS=X3*COS(X1)
      ASIN=X3*SIN(X1)
      CALL GAUSS(JSEED,DEV3,ACOS,Z1)
      CALL GAUSS(JSEED,DEV3,ASIN,Z2)
      XP=0.5*SQRT(ALOSS)
      DO 600 I=1,NUM3
        XX=I-YY3
        YA(I)=X3EST+XX*XP
 600  CONTINUE
      DO 730 I=1,16
      DO 730 K=1,16
      XNUM=(K-I)*XP+ALFD*(YA(I)-1.)
      XNUM=-0.5/Q33*XNUM**2
      EXP33(I,K)=0.0
      IF(XNUM.LT.-27.)GO TO 730
      EXP33(I,K)=EXP(XNUM)
 730  CONTINUE
      I=0
      DO 340 K=1,16
      DO 340 N=1,16
      DO 340 J=NTERM1,IYY
      I=I+1
 340  D(I)=EXPON(J)*EXP33(N,K)
 470  CONTINUE
C     ************************ SENSOR FUNCTION ***************************
      CALL Q3CLOCKS(T,TT)
      S1(1;16)=Z1*S1(1;16)
      S2(1;16)=Z2*S2(1;16)
      S1(1;16)=S1(1;16)+S2(1;16)
      J=1
      DO  500 K=1,16
      S2(1;16)=S1(1;16)*YA(K)
      SN2(1;16)=VEXP(S2(1;16);SN2(1;16))
```

```
      CALL VPROP(SN2,0)
      SN1(J;1536)=SN2(1;1536)
      S2(1;16)=-R55*YA(K)*YA(K)
      SN2(1;16)=VEXP(S2(1;16);SN2(1;16))
      CALL VPROP(SN2,0)
      SN1(J;1536)=SN1(J;1536)*SN2(1;1536)
  500 J=J+1536
C     ******************** MAIN LOOP STARTS ******************************
      CALL Q8VXTOV(X'02',0,KJNF,0,DB,0,DJNA)
      CALL Q8VXTOV(X'02',0,KJNS,0,DC,0,DJN1)
      JNA(1;26112)=JN1(2;26112)-JN1(1;26112)
      JNA(1;26112)=DELJ(1;26112)*JNA(1;26112)
      JN1(1;26112)=JN1(1;26112)+JNA(1;26112)
      JNA(1;24576)=Q8VCMPRS(JN1(1;26112),B3(1;26112);JNA(1;24576))
      JN1(1;24576)=JNA(1;24576)
      J=1
      I=1
      DO 510 K=1,16
      N=I+NTERM16
      JNA(N;1536)=JN1(J;1536)
      JNA(I;NTERM16)=JNA(I+1536;NTERM16)
      JNA(N+1536;NTERM16)=JN1(J;NTERM16)
      J=J+1536
  510 I=I+NTA1536
      N=1
      I1=0
      JK=1
      DO 700 I=1,16
      JN1(N;1536)=0.0
      DO 690 K=1,16
      J1=NS+I1
      J2=NS+I1
      DO 680 J=1,NTERM1
      JN(1;1536)=JNA(J1;1536)+JNA(J2;1536)
      JN(1;1536)=JN(1;1536)*D(JK)
      JN1(N;1536)=JN1(N;1536)+JN(1;1536)
      JK=JK+1
      J1=J1+16
  680 J2=J2-16
  690 CONTINUE
      N=N+1536
  700 I1=I1+NTA1536
      JN1(1;24576)=JN1(1;24576)*SN1(1;24576)
      CNORM=SUMLOG(JN1)
      IF(CNORM.LT.1.0E-20)CNORM=1.0
      CNORM=1./CNORM
      JN(1;24576)=CNORM*JN1(1;24576)
      JNA(1;24576)=COSY(1;2457)*JN(1;24576)
      CHAT=SUMLOG(JNA)
      JNA(1;24576)=SINY(1;24576)*JN(1;24576)
      SHAT=SUMLOG(JNA)
      CXHAT=ATAN2(SHAT,CHAT)
      J=1
      DO 343 K=1,16
      YB(J;1536)=YA(K)
```

```
 343  J=J+1536
      JNA(1;24576)=YB(1;24576)*JN(1;24576)
      X3EST=SUMLOG(JNA)
      JNA(1;24576)=JNA(1;24576)*YB(1;24576)
      ALOSS=SUMLOG(JNA)
      ALOSS=ALOSS-X3EST*X3EST
C     ************************** MAIN LOOP ENDS **************************
      CALL Q3CLOCKS(TNLF,TT)
      WRITE(6,201)KOUNT,X1,X2,X3,Z1,Z2,CXHAT,X3EST,ALOSS
 201  FORMAT(     I5,1X,1P3E14.6,4X,1P2E14.6,4X,1P3E14.6 )
      WRITE(6,8880)TNLF
 8880 FORMAT(1PE12.6)
      IF(KOUNT.EQ.NO2)GO TO 505
      KOUNT=KOUNT+1
      GO TO 450
 505  CONTINUE
      SUMP=0.0
      SUMC=0.0
      XDAT(KOUNT,2)=CXHAT
      XDAT(KOUNT,3)=ALOSS
      XDAT(KOUNT,4)=X3EST
      DO 1501 I=31,NO2
        XD=ABS(XDAT(I,1)-XDAT(I,2))
 1498 CONTINUE
        IF(XD.GT.PI)GO TO 1499
      GO TO 1500
 1499 XD=XD-PI2
      GO TO 1498
 1500 SUMP=SUMP+XD**2
 1501 CONTINUE
      H=NO2-30
      SUMP=SUMP/H
      XNSAMP=NSAMP
      XAA=XNSAMP+1.0
      SUMP1=(SUMP+XNSAMP*SUMP1)/XAA
      DSUMP1=ALOG10(SUMP1)*10.0
      DO 1601 I=31,NO2
        XD=ABS(XDAT(I,1)-XDAT(I,4))
 1698 CONTINUE
        IF(XD.GT.PI)GO TO 1699
      GO TO 1700
 1699 XD=XD-PI2
      GO TO 1698
 1700 SUMC=SUMC+XD**2
 1601 CONTINUE
      SUMC=SUMC/H
      SUMP2=(SUMC+XNSAMP*SUMP2)/XAA
      DSUMP2=ALOG10(SUMP2)*10.0
      WRITE(6,1511)NO3,SUMP1,DSUMP1,SUMP2,DSUMP2
 1511 FORMAT(I10,1P4E14.6)
      NSAMP=NSAMP+1
      IF(ISAMP.EQ.NO3)GO TO 2200
      ISAMP=ISAMP+1
      GO TO 11
```

```
2200 CONTINUE
     STOP
     END
     FUNCTION SUMLOG(A)
     REAL A(26112),C(12288)
     C(1;12288)=A(1;12288)+A(12289;12288)
     C(1;6144)=C(1;6144)+C(6145;6144)
     C(1;3072)=C(1;3072)+C(3073;3072)
     C(1;1536)=C(1;1536)+C(1537;1536)
     C(1;768)=C(1;768)+C(769;768)
     C(1;384)=C(1;384)+C(385;384)
     C(1;192)=C(1;192)+C(193;192)
     C(1;96)=C(1;96)+C(97;96)
     C(1;48)=C(1;48)+C(49;48)
     C(1;24)=C(1;24)+C(25;24)
     C(1;12)=C(1;12)+C(13;12)
     C(1;6)=C(1;6)+C(7;6)
     C(1;3)=C(1;3)+C(4;3)
     SUMLOG=C(1)+C(2)+C(3)
     RETURN
     END
     SUBROUTINE VPROP(A,I)
     REAL A(24576)
     IF(I.EQ.2)GO TO 10
     A(17;16)=A(1;16)
     A(33;32)=A(1;32)
     A(65;32)=A(1;32)
10   A(97;96)=A(1;96)
     A(193;192)=A(1;192)
     A(385;384)=A(1;384)
     A(769;768)=A(1;768)
     IF(I.EQ.0)RETURN
     A(1537;1536)=A(1;1536)
     A(3073;3072)=A(1;3072)
     A(6145;6144)=A(1;6144)
     A(12289;12288)=A(1;12288)
     RETURN
     END
     SUBROUTINE VPROP1(A)
     REAL A(26112)
     A(103;102)=A(1;102)
     A(205;204)=A(1;204)
     A(409;408)=A(1;408)
     A(817;816)=A(1;816)
     A(1633;1632)=A(1;1632)
     A(2265;2264)=A(1;2264)
     A(3265;3264)=A(1;3264)
     A(6529;6528)=A(1;6528)
     A(13057;13056)=A(1;13056)
     RETURN
     END
     FUNCTION RNNF(NS,MODE)
```

```
      DIMENSION  NS(2), NC(2)
      COMMON /RN/ N1, N2, MP, T1, T2
      DATA  M1, M2/244734, 158551/
C        MODE=0 TO CONTINUE, OTHERWISE RESTART WITH
C        INTEGER NUMBER NS(1)*2**18+NS(2)
      IF (MODE) 10, 100, 10
   10 N1=NS(1)
      N2=NS(2)
      T1=2.**(-18)
      T2=2.**(-36)
      MP=2**18
  100 DO 200 I=1,2
      GO TO (110,120),I
  110 K=M2*N2
      GO TO 190
  120 K=M1*N2+M2*N1+KD
  190 KD=K/MP
  200 NC(I)=K-KD*MP
      N1=NC(2)
      N2=NC(1)
      XN1=N1
      XN2=N2
      RNNF=XN1*T1+XN2*T2
      RETURN
      END
      SUBROUTINE GAUSS(JS,SD,XM,X)
      DIMENSION  NST(2)
      COMMON /RN/ N1, N2, MC, T1, T2
      COMMON /GN/ J,XR(2)
      IF (J) 10, 10, 20
   10 J=2
      TWOPI=8.*ATAN(1.)
      NST(1)=244734
      NST(2)=158551
      NST(1)=102943
      NST(2)=185617
      XR(1)=RNNF(NST,1)
      GO TO 35
   20 GO TO (30,40), J
   30 J=2
      XR(1)=RNNF(NST,0)
   35 XR(2)=RNNF(NST,0)
      X1=SQRT(ABS(-2.*ALOG(XR(1))))
      XR(2)=TWOPI*XR(2)
      XR(1)=X1*SIN(XR(2))
      XR(2)=X1*COS(XR(2))
      X=XR(1)*SD+XM
      RETURN
   40 J=1
      X=XR(2)*SD+XM
      RETURN
      END

 1.0        0.01        1.0        1.4        16
 0.0        0.0        -3.0        0.1        0.01        16    96   130     1
END OF FILE
```