COMPUTATIONAL ANALYSIS OF THE Q-FUNCTION FOR INTERMEDIATE TO LARGE PARAMETER RANGE

by

Jeff L. Freeman

Department of Electrical Engineering

Submitted in Partial Fulfillment of the Requirements of the University Undergraduate Fellows Program

1985-86

Approved by:

P. E. Cantrell

May 1986

ABSTRACT

Computational Analysis of the Q-function

for Intermediate to Large Parameter Range. (April 1986)

Jeff L. Freeman, B.S., Texas A & M University

Advisor: Dr. Pierce E. Cantrell

Cantrell has developed a very efficient algorithm using Parl's method for accurately calculating the generalized Q-function $Q_m(\alpha,\beta)$. Parl's method using floating formats with the exponent larger than However, only values of m up to about 100 could be studied using the Real*8 floating point format. An investigation into the range m >100 is made. Real*8 F-floating are run to determine the limitations of the Parl method. Results are presented in this range of m using other floating point formats and Rice's asymptotic expansion for the non-central chi-square distribution and the probability of detection for a classical multi-observation detection problem. Also, plots of number of significant figures of the Rice uniform asmptotic expansion for different iterations verses m are given . Finally, a comparison of CPU time required for the Parl method and the Rice algorithm are presented for each iteration used in the significant figure plots.

ACKNOWLEDGMENTS

I appreciate all the support, patience, and criticism of P. Cantrell throughout the past year. I also appreciate the invaluable assistance of L. Schovanec in deriving some of the series expansions that are part of the Rice algorithm.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF FIGURES (Continued)

I. INTRODUCTION

The generalized Q-function [7], [8, pp. 219-223] is related to the non-central chi-square [4], Rayleigh, Poisson, and Rician distribution functions. Communication problems, radar detection [8], and transient M/M/1 queues are just a few of the applications for the Q-function.

The generalized Q-function is defined by the integral

$$Q_m(\alpha,\beta) = \int_\beta^\infty u\left(\frac{u}{\alpha}\right)^{m-1} \exp\left(\frac{-(\alpha^2+u^2)}{2}\right) I_{m-1}(\alpha u)du \tag{1}$$

where $I_\epsilon(z)$ is a modified Bessel function and is given by the series expression

$$I_\epsilon(z) = \sum_{k=0}^{\infty} \frac{\left(\frac{z}{2}\right)^{\epsilon+2k}}{k!\Gamma(\epsilon+k+1)}. \tag{2}$$

Of all the methods that have been proposed for approximating the Q-function, which include asymptotic and power series expansions, the Parl and Rice algorithms will be examined in this thesis. Parl's method is generally applicable for values of m less than 100 because of it is limited by the floating point format of the computer [6]. However, Cantrell's computer program of Parl's algorithm was modified to run using Real*8 G floating point which extended the exponent from 38 to 308, dropping only one significant figure. This extended the range of the Parl algorithm to values of m less than 500.

There are applications today that require a range of $m > 500$. The next section shows two of the possible applications and compares the results of using the Parl algorithm alone and then both the Parl and Rice algorithms combined.

Journal model is *IEEE Transactions on Automatic Control.*

II. APPLICATION

A. Non-central Chi-square distribution

In Figs. 1 and 2, the logarithm of the Q-function and its complement are plotted versus β for several values of α using the F-floating point format [6]. In each figure m is constant. The number of significant figures is shown on the curve when the significant figures drops below 11. Fig. 1 was plotted using Parl's algorithm alone; whereas Fig. 2 was plotted from a combined Parl and Rice procedure. If the Parl method fails, the Rice algorithm computes the result. Note that the combined algorithms are able to compute the Q-function results of Fig. 2 to 11 significant figures for $\alpha = 10$ and most of the range for $\alpha = 1$. The Parl method alone cannot be used to compute $\alpha = 1$ in Fig. 1 due to floating point overflow.

B. Probability of Detection

Figs. 3 and 4 represent plots of probability of detection for a multiple observation detection problem versus the SNR, α, with $2m$ degrees of freedom and a desired false alarm probability, P_{fa}, [6, p.12], [7], where

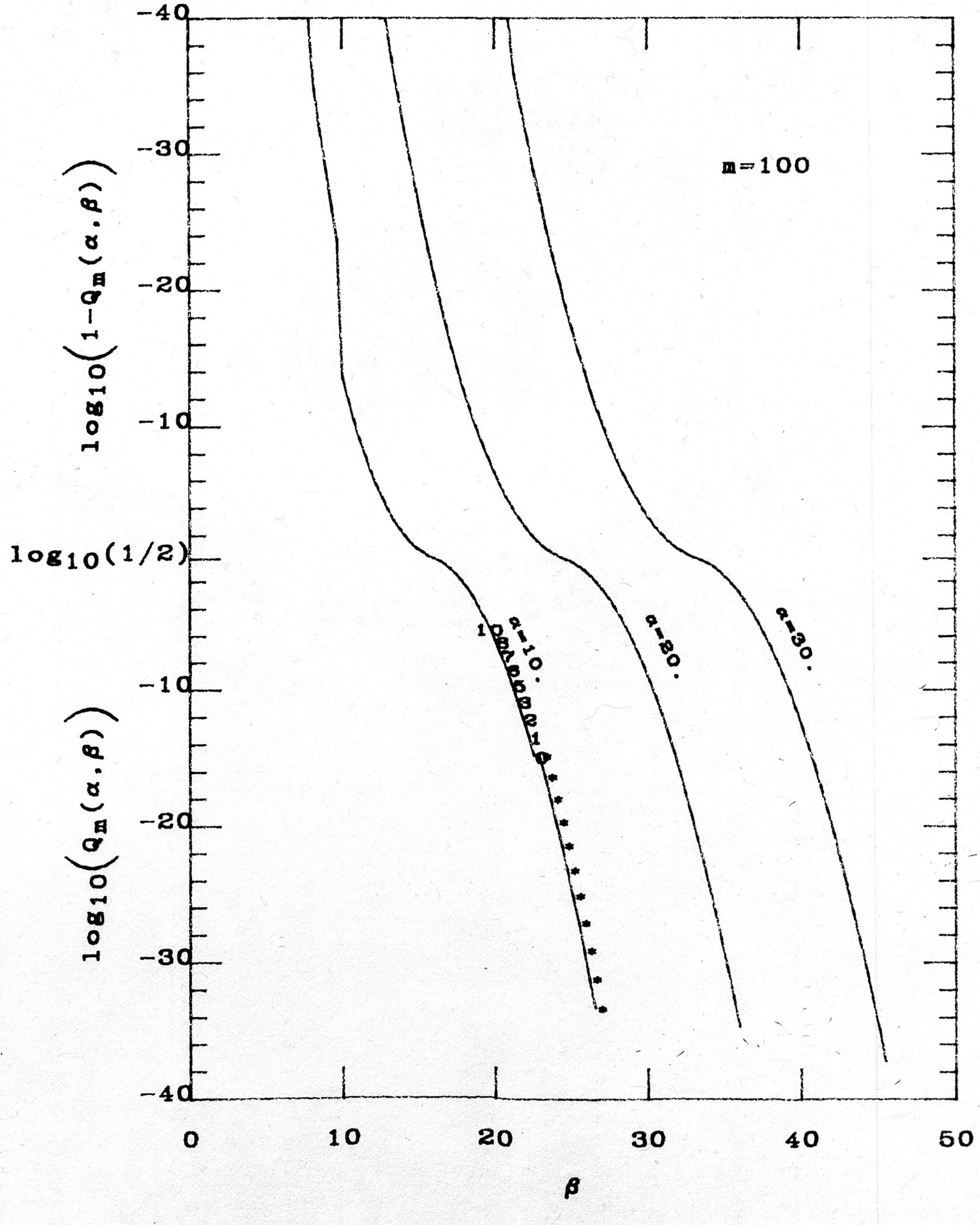
$$P_{fa} = Q_m(0, \sqrt{\gamma})$$

, and the probability of detection is given by

$$P_D = Q_m(\alpha, \sqrt{\gamma})$$

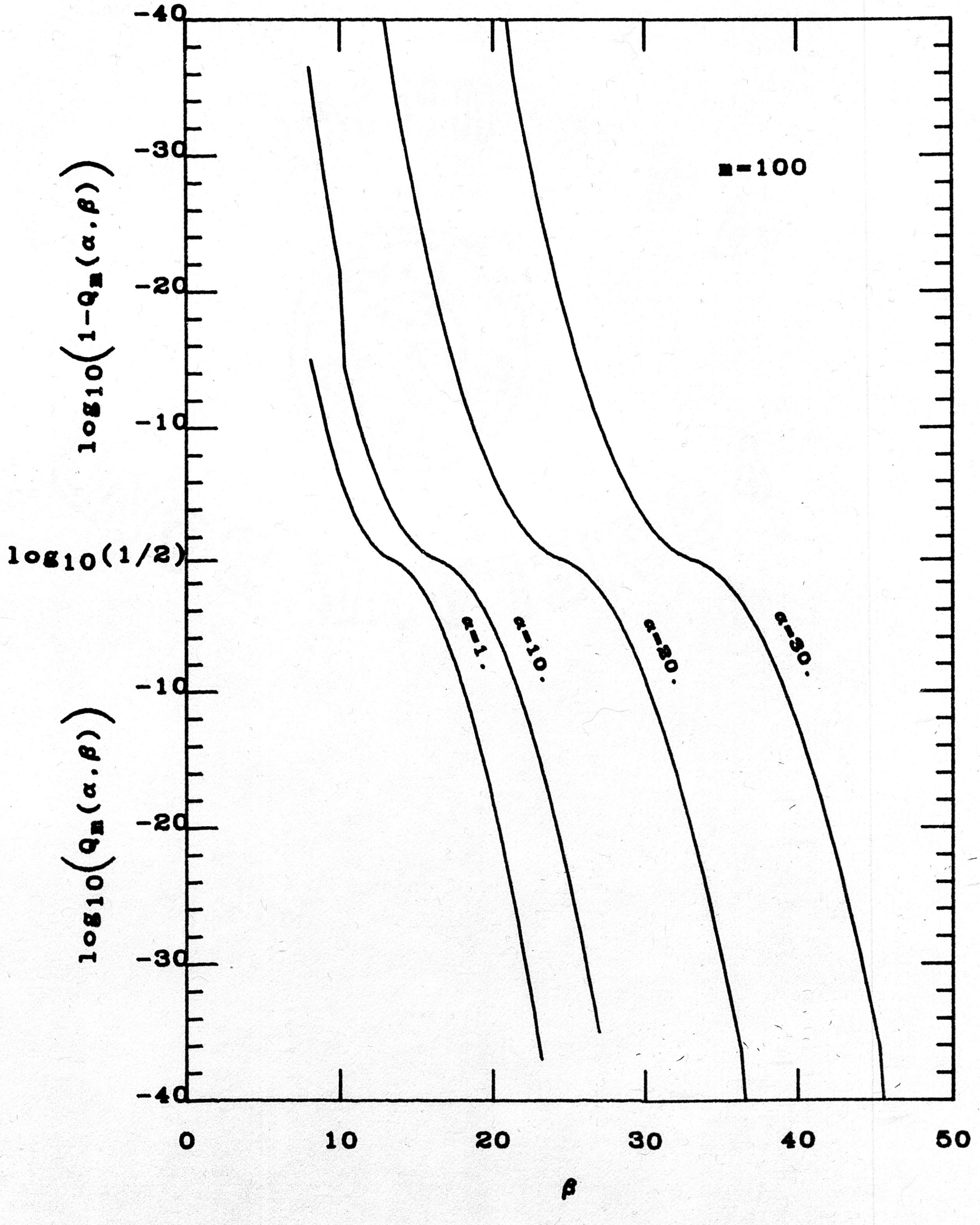
. Figs. 3 and 4 show only the central region $10^{-4} \leq P_D \leq .9999$.

In Fig. 1, the curves for $m = 100$, 150, and 200 cannot be computed over the entire range due to floating point overflow. As m increases, the minimum value of



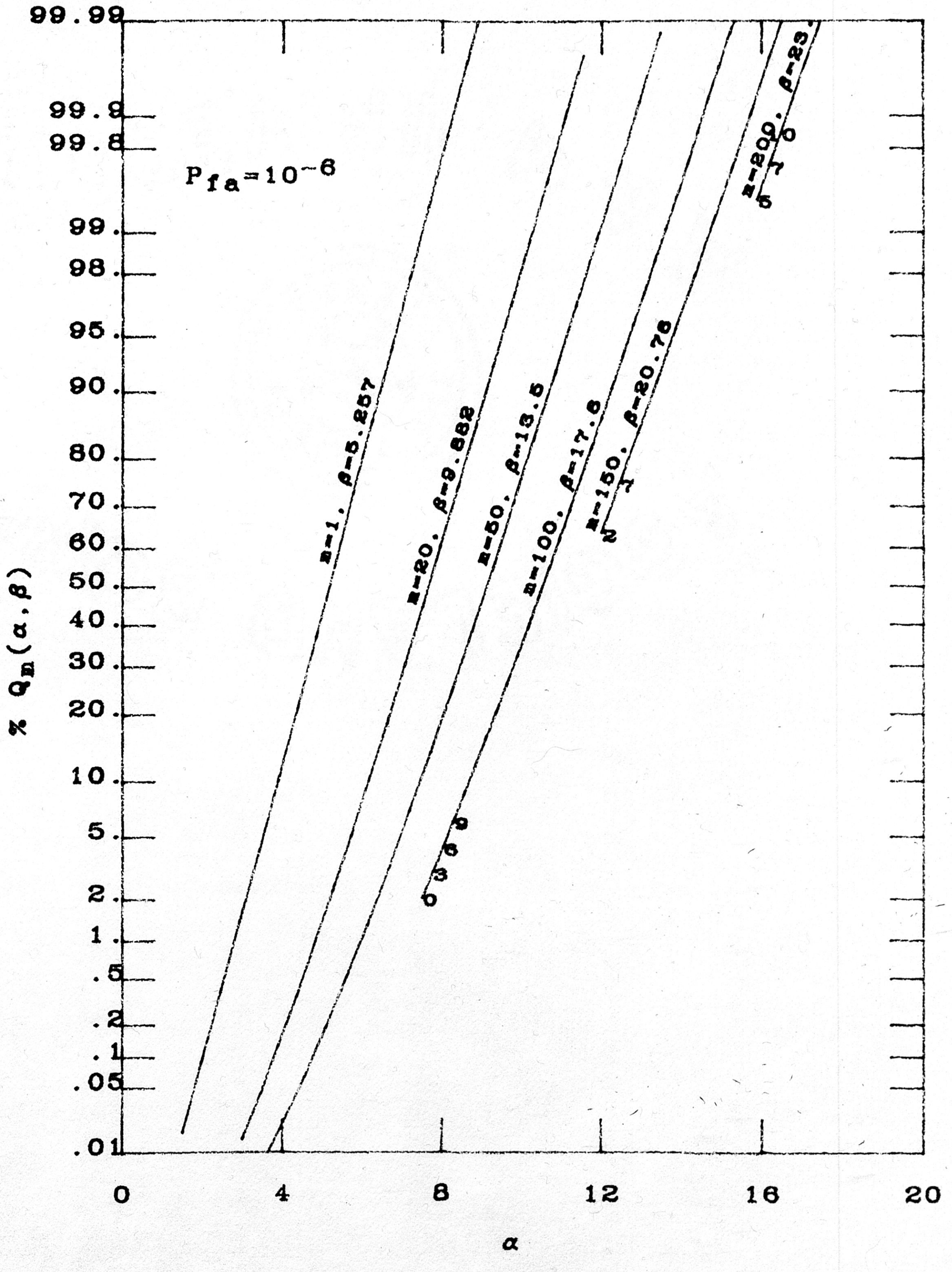
-40
-30
-20
-10
log10(1/2)
-10
-20
-30
-40
0
10
20
30
40
50
β
log10(1 − Q_m(α, β))
log10(Q_m(α, β))
m=100
α=10.
α=20.
α=30.
10

Fig. 1.



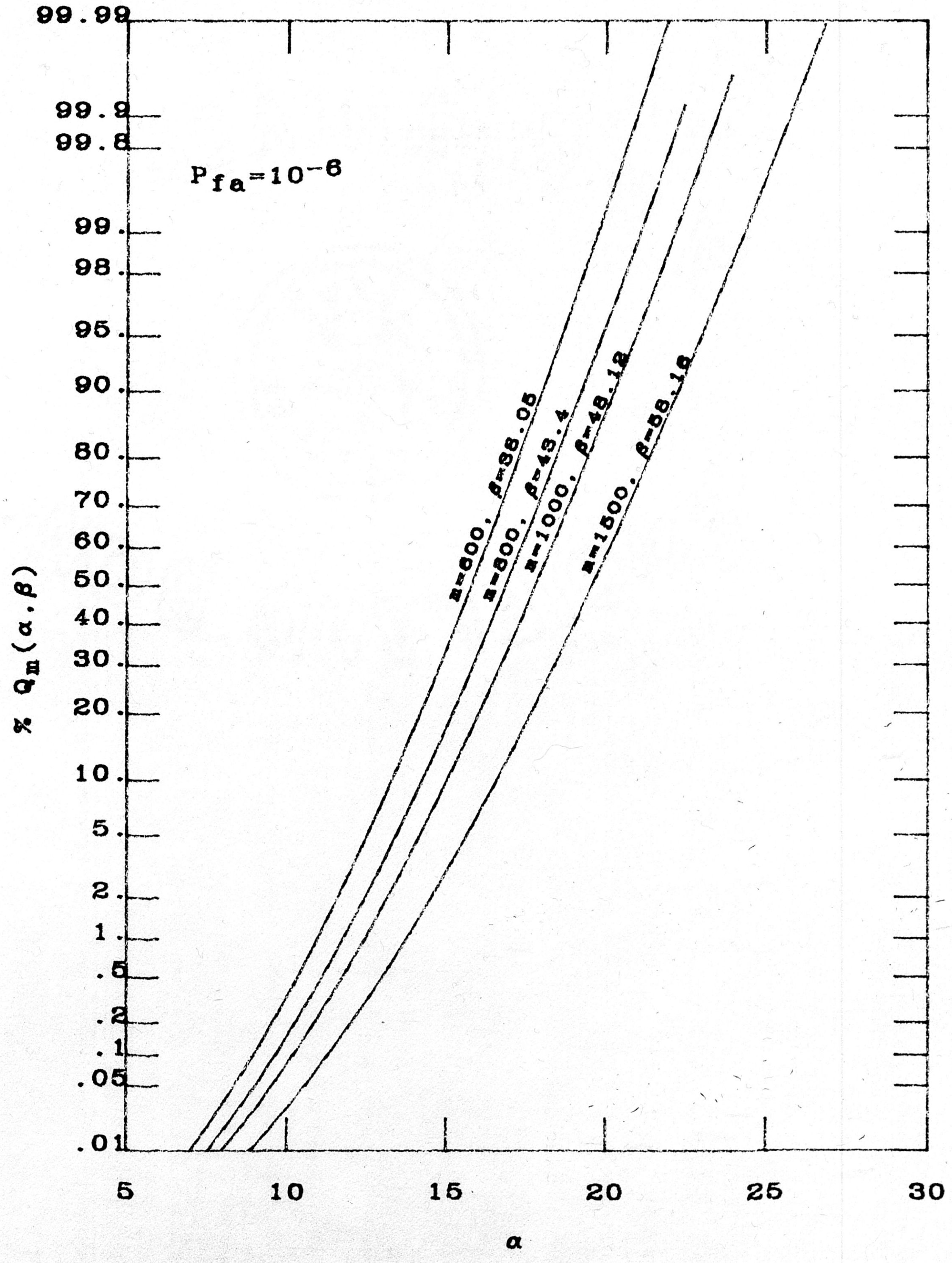
-40
-30
-20
-10
$\log_{10}(1/2)$
-10
-20
-30
-40
$\log_{10}\left(1-Q_m(\alpha,\beta)\right)$
$\log_{10}\left(Q_m(\alpha,\beta)\right)$
m=100
α=1.
α=10.
α=20.
α=30.
0
10
20
30
40
50
β

Fig. 2.



99.99
99.9
99.8
99.
98.
95.
90.
80.
70.
60.
50.
40.
30.
20.
10.
5.
2.
1.
.5
.2
.1
.05
.01
0
4
8
12
16
20
α
% $Q_m(\alpha,\beta)$
$P_{fa}=10^{-6}$
m=1, β=5.257
m=20, β=9.682
m=50, β=13.5
m=100, β=17.6
m=150, β=20.76
m=200, β=23.

Fig. 3.



$P_{fa}=10^{-6}$
m=600, β=38.05
m=800, β=43.4
m=1000, β=48.12
m=1500, β=58.16
% $Q_m(\alpha,\beta)$
99.99
99.9
99.8
99.
98.
95.
90.
80.
70.
60.
50.
40.
30.
20.
10.
5.
2.
1.
.5
.2
.1
.05
.01
5
10
15
20
25
30
α

Fig. 4.

α for which $Q_m(\alpha, \sqrt{\gamma})$ can be calculated also increases [6, p. 12]. Cantrell [6] gave an equation to predict α_{min}, and we verified its validity in predicting α_{min}, even for Real*8 G format. We note that this α_{min} tends to change experimentally more than predicted as m increases. Therefore we added a small safety factor term to Cantrell's equation in obtaining plots for $m \geq 100$.

In section III Rice's uniform asymtotic expansion [3] is introduced to calculate the Q-function where Parl's method suffers from floating point overflow. In addition, the CPU time when using Parl's method increases as the value of m increases so it is anticipated that for some m Rice's asymptotic expansion will be as accurate as Parl's method and require a smaller execution time.

I. RICE'S ALGORITHM

A. Method

In Appendix A the Rice approximation is derived, which with a change of variables gives [3, pp. 1990-1991]

$$1 - Q_m(\alpha,\beta) = \frac{1}{2}\{1 - \mathrm{erf}[v_1(s,r)x^{\frac{1}{2}}]\} + \frac{1}{2}(\pi x)^{-\frac{1}{2}} \exp(-x[v_1(s,r)]^2) \sum_{n=0}^{\infty} p_{n1}(x,s,r)x^{-n} \qquad (3)$$

where $x = 2m$, $s = \beta/x$, $r = \alpha/x$, v_1 is a function of s and r, and p_{n1} is a function of x, s, and r.

Essentially, Rice's algorithm is an error function plus a correction series for the tails of the distribution function. Thus, by implementing this general algorithm in a program, Rice's expansion can be compared in terms of relative error with respect to the Parl method calculated with a relative error of 1 x 10^{-12}.

From extensive evaluation of the usable range of the Rice algorithm, we found that the Rice algorithm is reasonably accurate above the value of m=50. Depending on the number of terms in the Rice tail series, significant figures are determined. Increasing the number of iterations generally increases the accuracy of the Rice algorithm. The number of significant figures is obtained from two investigations. First, the parameters α, β, m are varied individually over a range where the Chernoff bound of the Q-function or complementary Q-function exceeds 10^{-24} Secondly, the parameters are varied on a plot of the probability of detection versus SNR. For this application, the Q-function is between 10^{-4} to .9999. The next subsection will discuss the problems encountered in computation using the Rice algorithm.

B. Algorithm Problems

The Rice algorithm had several major problems associated with computation which are all related to a loss of significance from catastrophic cancellation. Even with Real*16, the algorithm failed in the initial research stages.

For certain values of α, β, and m the algorithm did not give a usable relative error. The tail series which is the right most expression in (3) can be represented by

$$P = K(p1 - 1)$$

where K is a constant and $p1$ is an iterative term. When the algorithm fails, the constant is very large and $p1$ is very close to one. At this point the series P is computed as a very small difference between two large numbers. Thus, even with Real*16 implementation, much precision is lost which makes the algorithm fail [3,p. 1990]. However, a Taylor expansion at this point solves the problem (see Appendix B).

The first method of investigating the significant figures must be limited by a Chernoff bound of 10^{-24}. On the extreme tails of the distribution function the mechanism for loss of significant figures can be determined by examining (3). The loss of accuracy occurs when calculating Q and not $1-Q$, therefore an expression for Q must be derived since the two terms to the right of the equal sign in (2) are nearly equal at this point and on the order of 10^{-24} (this was determined experimentally). In most applications, a Chernoff bound of 10^{-24} is more than sufficient.

II. DISCUSSION OF RESULTS

A. Significant Figure Plots

In the plots that have significant figures verses m the significant figures increase as m increases. Fig. 5 is a plot of the number of worst case significant figures versus m. The worst case simply means that for each m, α and β were varied within the Chernoff bound of 10^{-24} and the minimum significant figure obtained was used as a data point for that particular value of m. For this and all of the significant figure plots, the relative error of the modified Parl algorithm is used as a comparison since it was calculated with a relative error of less than 10^{-12}. By computing the Q-function for the Parl and Rice methods, a relative error for the Rice iteration can be obtained. In Fig. 5 the legend has four different linestyles that indicate the number of iterations of the Rice algorithm. As the iterations increase, the accuracy increases at the expense of some CPU time. Additional iterations are not shown since we found no improvement in accuracy. Note that 11 significant figures is the maximum(Parl's algorithm was only calculated to 11 significant figures) which is reached in Fig. 9 for Real*16. As m increases, the number of significant figures for

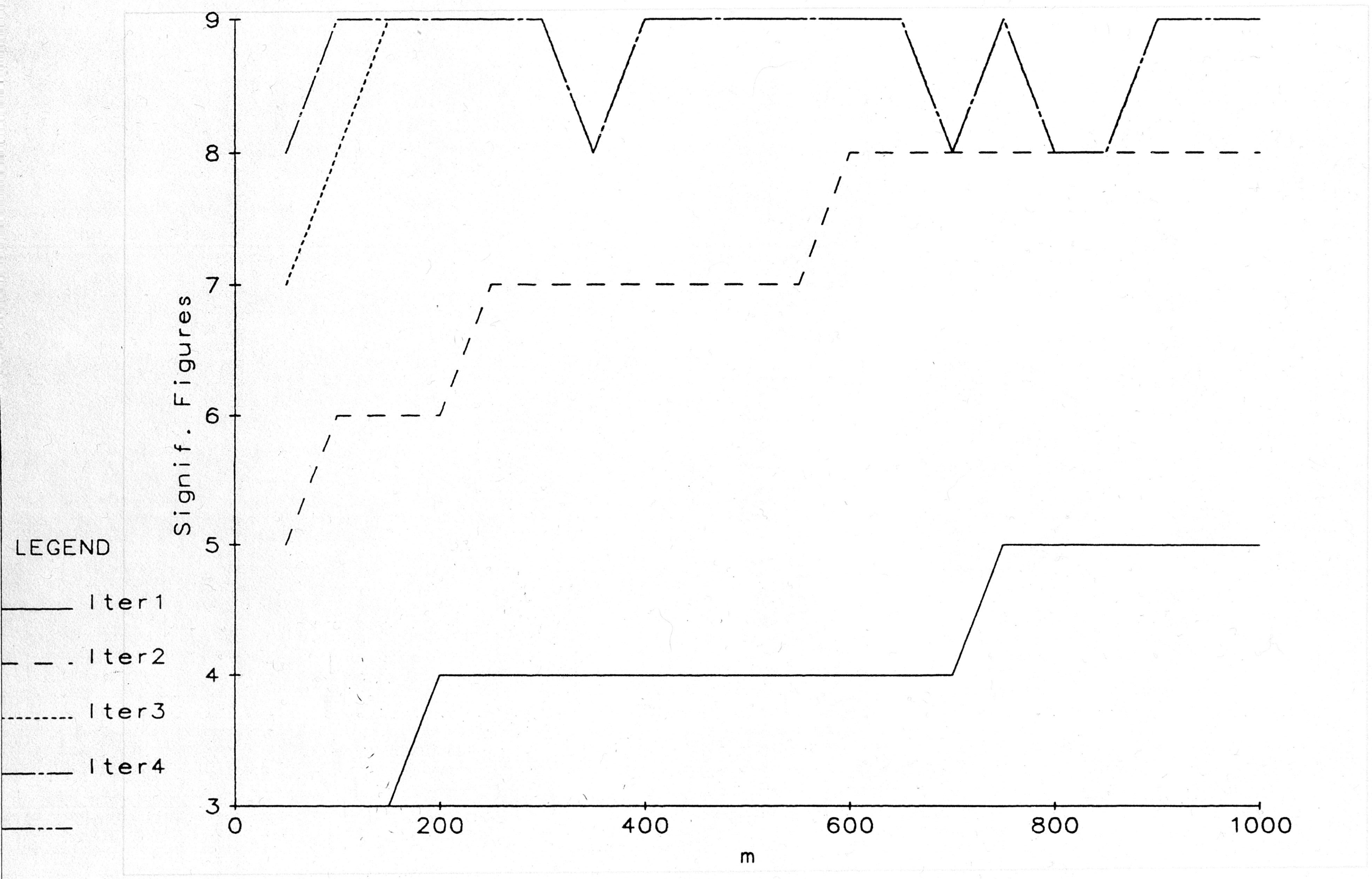


FIG **5** REAL*16/**H** FLOATING CHERNOFF 10^-24

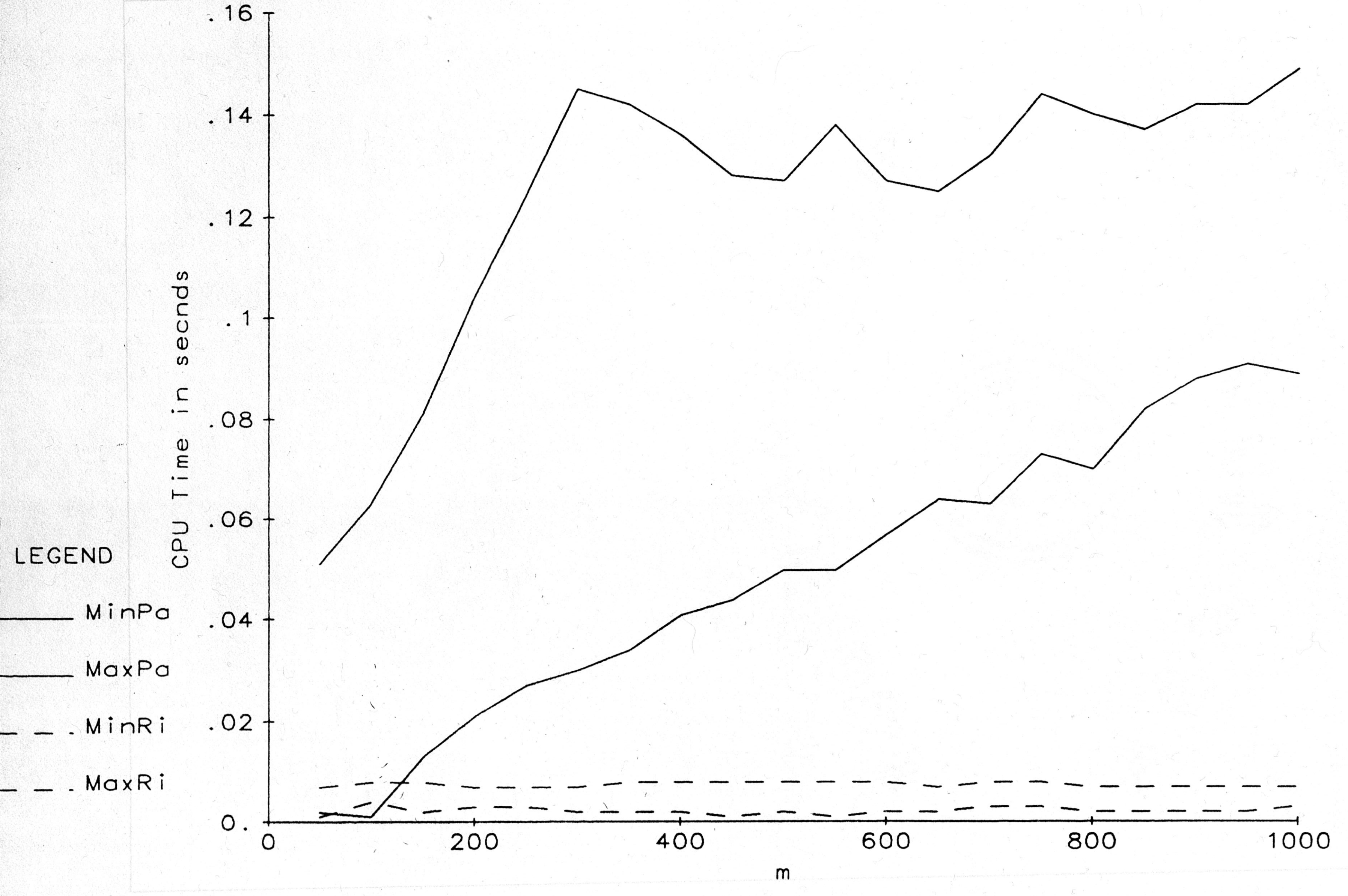


FIG **6** REAL*16/H CHERNOFF 10^-24 ITER 1 a fixed number of terms in the asymtotic expansion increases. These data can be used to obtain a given number of significant figures when using Rice's method.

The numerical data of the significant figure plots is used in calculating the CPU time of each algorithm, on an equal basis as will be explained in the following section.

B. CPU Time Plots

The next plots of CPU time verses m are derived from using the data of the first plots to keep the relative error of the Parl and Rice method the same as the subroutine times are plotted.

We expect the CPU plots to give us an optimum value of m to switch over from using the Parl algorithm to using the Rice algorithm for a given significant figure requirement. For example, in Fig. 6 the CPU time versus m is plotted for the first iteration of the Rice algorithm. The maximum and minimum CPU times of each algorithm are shown, the solid line representing the minimum and maximum times of Parl's method and the dotted line representing the minimum and maximum times of Rice's method. To keep the relative error of the two algorithms equal to each other, the data from Fig. 5 was used as input. By sending the relative error of the Rice algorithm to the Parl algorithm(the input from Fig. 5), the relative errors of each are assured to be the same since the Parl algorithm iterates until it meets a desired relative error. In Fig. 6 the value of m to switch from Parl's algorithm to Rice's asymtotic expansion is about 50, which is obtained from the min/max CPU times of both algorithms.

I. CONCLUSION

From the CPU plots, the Parl algorithm is much slower than the Rice algorithm for $m \geq 50$. If both algorithms are implemented in one program, an entire range of parameters can be usable. Thus, for $m \geq 50$ the value of m to switch from Parl's algorithm to Rice's expansion is much more dependent on the number of significant figures required. In general, for $m \geq 50$ Rice's algorithm should be used due to the fast execution time. These results are summarized in the table on the following page for different m values and different significant figure requirements.

Depending on the software and computer system available, this table and the plots will allow one to write a program to compute the Q-function for a large range of parameters with the Rice subroutine given in Appendix C.

SUMMARY of RESULTS

TABLE 1 CHERNOFF of 10^{-24}

	Real*8	NOG/G	Real*16	/G
Rice Iter.	m	Signif. Fig.	m	Signif. Fig.
1	200	4	125	3
2	270	4	50	5
3	200	4	50	7
4	-	-	50	8
5	-	-	-	-

TABLE 2 PDET for CHERNOFF of 10^{-4} to .9999

	Real*8	NOG/G	Real*16	/G
Rice Iter.	m	Signif. Fig.	m	Signif. Fig.
1	150	4	50	3
2	150	6	50	5
3	-	-	50	7
4	-	-	50	9
5	-	-	50	11

REFERENCES

[1] A.K. Ojha, "Comparative Performance of Algorithms for Machine Computation of the Q-function," M.S. Thesis,Dept. of Electrical Engineering, Texas A & M University, May 1985.

[2] S. Parl, "A new method of calculating the generalized Q-function," *IEEE Trans. Inform. Theory*, vol. IT-26, pp. 121-124, Jan. 1980.

[3] S.O. Rice, "Uniform asymptotic expansions for saddle point integrals-appl. to cdf's," *Bell. Sys. Tech. J.*, vol. 47, pp. 1971-2013, Nov. 1968.

[4] G.H. Robertson, "Computation of the noncentral chi-square distribution," *Bell. Sys. Tech. J.*, vol. 48, pp. 210-207, Jan. 1969.

[5] J.L. Schonfelder, "Chebyshev Expansions for the Error and Related Functions," *Math. of Computation.* , ED. : , vol.32, chap. 1232-1240, pp. Oct. 1978.

[6] P.E. Cantrell, "On the calculation of the generalized Q-function via Parl's method," IEEE Trans. Inform. Theory, vol. IT-32, Nov. 1986.

[7] J.I. Marcum, "A statistical theory of target detection by pulsed radar:math. app.," *IRE Trans. Inform. Theory*, vol. IT-6, pp. 59-267, April 1960.

[8] C.W. Helstrom, *Statistical Theory of Signal Detection.* New York,NewYork: Pergamon, 1968.

APPENDIX A

A CHANGE OF VARIABLES FROM THE NON-CENTRAL CHI-SQUARE DISTRIBUTION TO THE Q-FUNCTION

The Q-function is related to the cummulative distribution function (cdf) and complementary cdf of the non-central chi-square [6]. Let $z = 1/n(X'^2)$ where X'^2 is non-central X^2 with n^0 of freedom and non-centrality parameter λ.

$$X'^2 = nz$$

$$\Pr\{0 \le z \le s\} = \Pr\{0 \le X'^2(\lambda)/n \le s\}$$

$$= \Pr\{0 \le {X_n}'^2(\lambda) \le ns\}[6]$$

$$r = \lambda/n$$

$$1 - Q_m(\alpha, \beta) = \Pr\{0 \le {X_{2m}}'^2(\alpha^2) \le \beta^2\}$$

therefore

$$\beta^2 = ns$$

$$\alpha^2 = \lambda$$

$$2m = n,$$

for P_D give m, α, $\beta = \sqrt{\gamma}$.

From above

$$n = 2m \tag{A.1}$$

$$s = \beta^2/n = \gamma/2m \tag{A.2}$$

$$r = \lambda/n = \alpha^2/2m. \tag{A.3}$$

APPENDIX B

DERIVATION OF RICE TAIL SERIES USING A POLYNOMIAL EXPANSION AND A TAYLOR SERIES EXPANSION

Rice [3], [p. 1990] mentions that for certain parameters, his algorithm must be expanded in a Taylor series since the expressions become essentially small differences between large numbers. Since the derivation was not given in his paper, a simple case will be derived to show how a good approximation for this saddle point can be made. The first term in the Rice tail series is given by

$$p_{01} = 1/v_1[(v_1 t_1^{(1)})/t_1 - 1]. \qquad (B.1)$$

It remains to be shown the this is equivalent to

$$p_{01} = (a_1/2)t_1^{(1)} + O(t_1), \qquad (B.2)$$

where O(t_1) are important higher order terms dominated by terms of t_1 [3], [p. 1990]. By setting $t = 0$ after expanding $h(t)$ about $t = t_1$ leads to a series for $-h_1$ which can be used to get v_1^{-2j-1} as $[t_1{}^{(1)}/t_1]^{2j+1}]$ times a power series in t_1 [3], [pp. 1990,1991]. Expanding $h(t)$

$$h(t) = h(t_1) + h'(t_1)(t - t_1) + h^{(2)}(t_1)/2!(t - t_1)^2 + \cdots$$

$$-h_1 = h_1^{(2)}(t_1)^2/2! - h_1^{(3)}(t_1)^3/3! + \cdots \qquad (B.3)$$

Substitute this series into (B.1)

$$p_{01} = [1/(t_1)[2/h_1^{(2)}]^{\frac{1}{2}} - (|t_1|/t_1)[h_1^{(2)}(t_1)^2/2! - h_1^{(3)}(t_1)^3/3! + O(t_1{}^4)]^{-\frac{1}{2}} \qquad (B.4)$$

$$p_{01} = [1/(t_1)[2/h_1^{(2)}]^{\frac{1}{2}} - (1/t_1)[h_1^{(2)}/2! - h_1^{(3)}(t_1)/3! + O(t_1{}^2)]^{-\frac{1}{2}} \qquad (B.5)$$

Now factor out $[h_1^{(2)}/2]^{\frac{1}{2}}$

$$p_{01} = [1/(t_1)[2/h_1^{(2)}]^{\frac{1}{2}}\{1 - [1 - h_1^{(3)}/h_1^{(2)}(t_1)/3 + O(t_1{}^2)]^{-\frac{1}{2}}\}. \qquad (B.6)$$

But this can be simplified by substitution [3], [p. 1989 eq.(71)] in terms of a_k

$$p_{01} = [1/(t_1)[2/h_1^{(2)}]^{\frac{1}{2}}\{1 - [1 + a_1(t_1) - a_2(t_1)^2 + O(t_1{}^2)]^{-\frac{1}{2}}\}. \qquad (B.7)$$

Let X equal the series terms under the denominator of the second term in brackets. Taking a binomial expansion

$$(1 + X)^{-\frac{3}{2}} = \sum_{k=0}^{2} \binom{-\frac{3}{2}}{k} X^k = 1 - (X/2) + (3/8)X^2 + O(X^3).$$

Thus, substitute $t_1^{(1)}$ for $[2/h_1^{(2)}]^{\frac{1}{2}}$ and some terms of t_1 cancel

$$p_{01} = t_1^{(1)}[a_1 - t_1(a_2 + (3/4)a_1{}^2)] + O(t_1{}^2). \qquad (B.8)$$

Note that the above expression is different from (B.2) since higher order terms of t_1 must be included to get the required accuracy. The terms of $(t_1)^2$ and up can be dropped.

In general where $|t_1|$ is very small,

$$p_{j1} = (-1)^j\left(\frac{1}{2}\right)_j (t_1{}^{(1)})^{2j+1}[1 - \sum_{k=0}^{2j+2} \binom{-\frac{2j+1}{2}}{k} X^k], \qquad (B.9)$$

and

$$X = a_1(t_1) - a_2(t_1)^2 + a_3(t_1)^3 - \cdots - a_j(t_1)^{2j+2}$$

$$(c)_o = 1, (c)_n = c(c+1)\cdots(c+n+1).$$

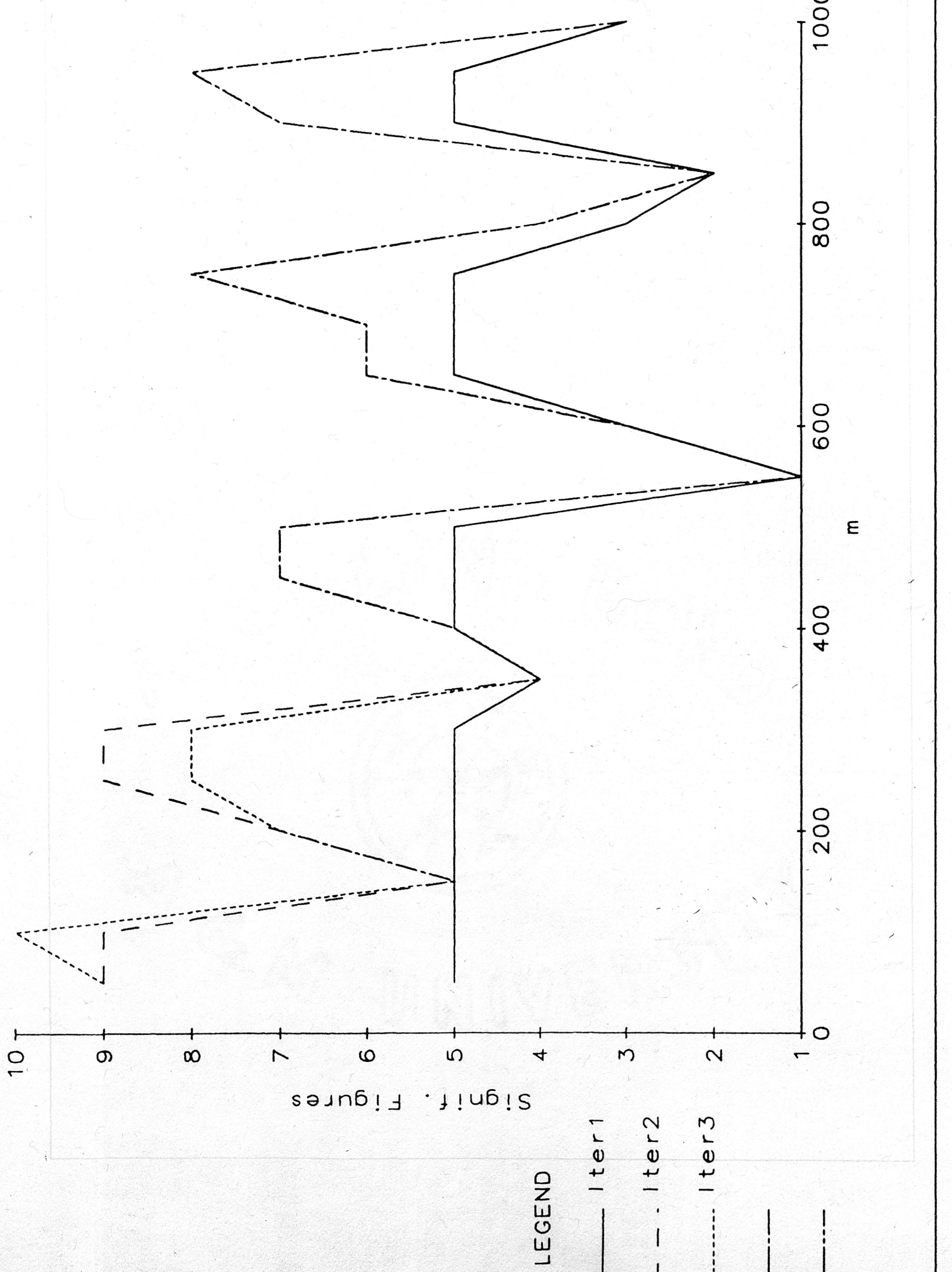


FIG 7 REAL*8/NOG/G CHERNOFF 10^-24

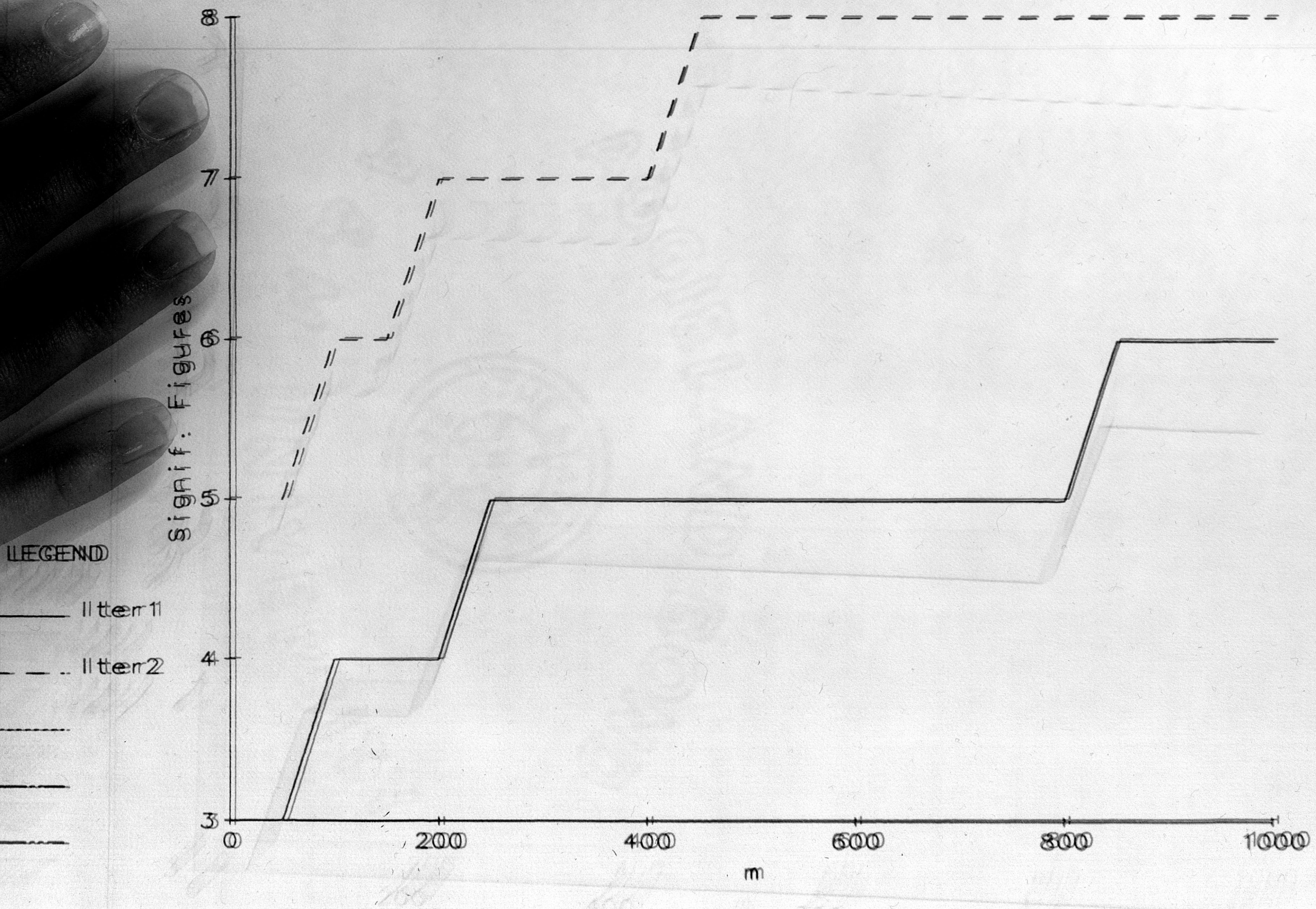


FIG 8 REAL*8/NOG/G PDET

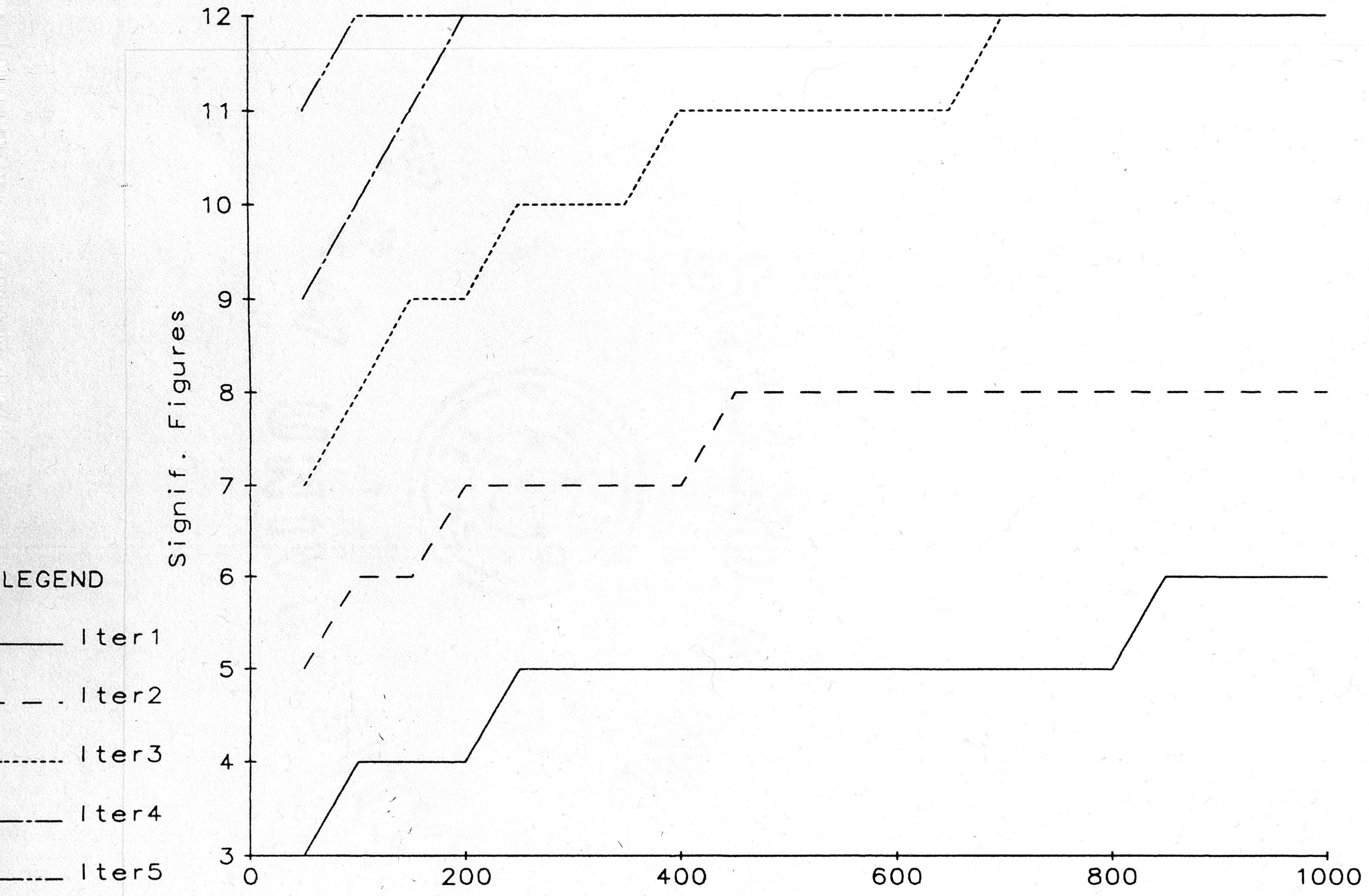


FIG 9 REAL*16/H FLOATING PDET

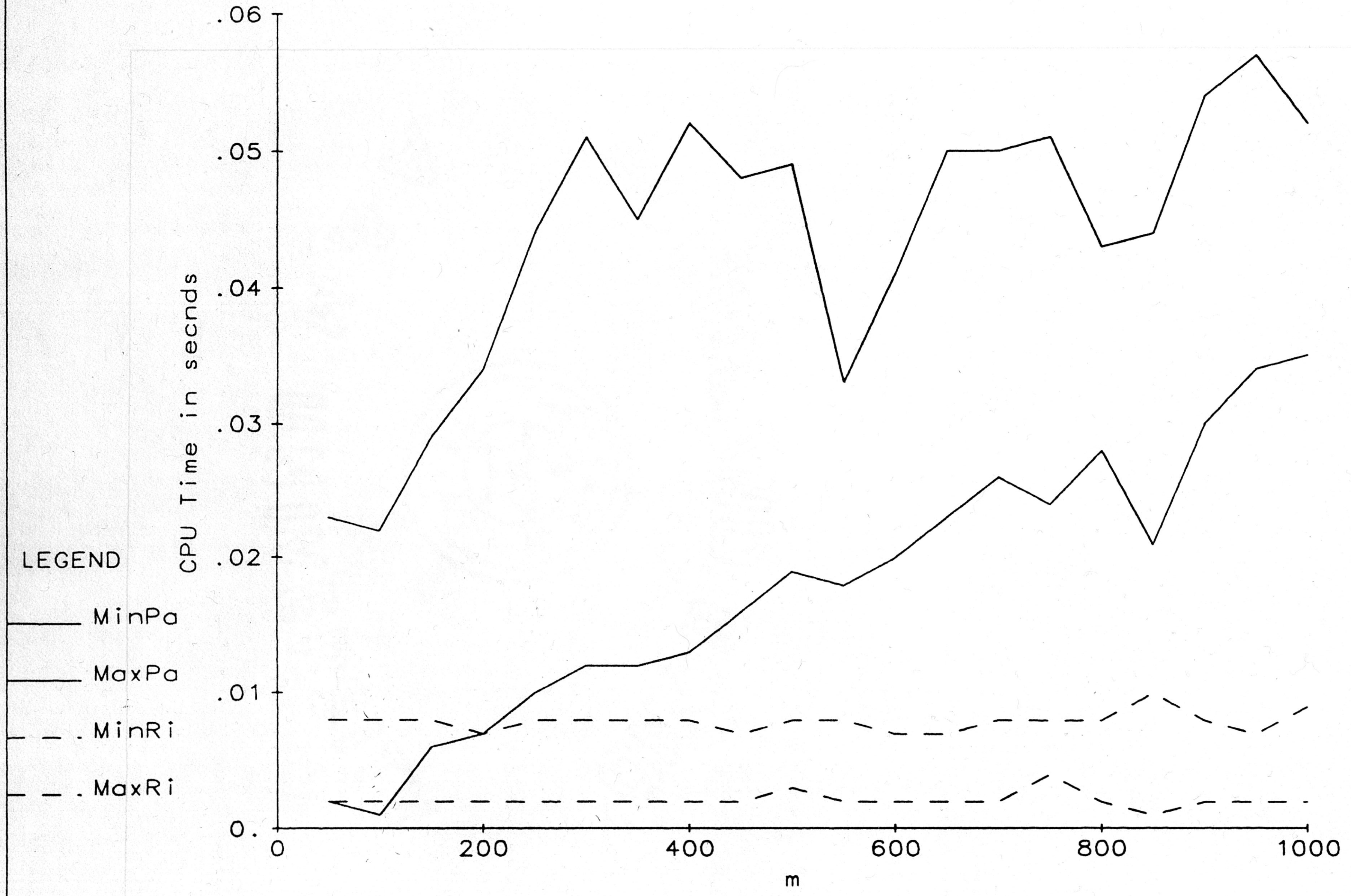


FIG **10** REAL*8/NOG/G CHERNOFF 10^-24 ITER 1

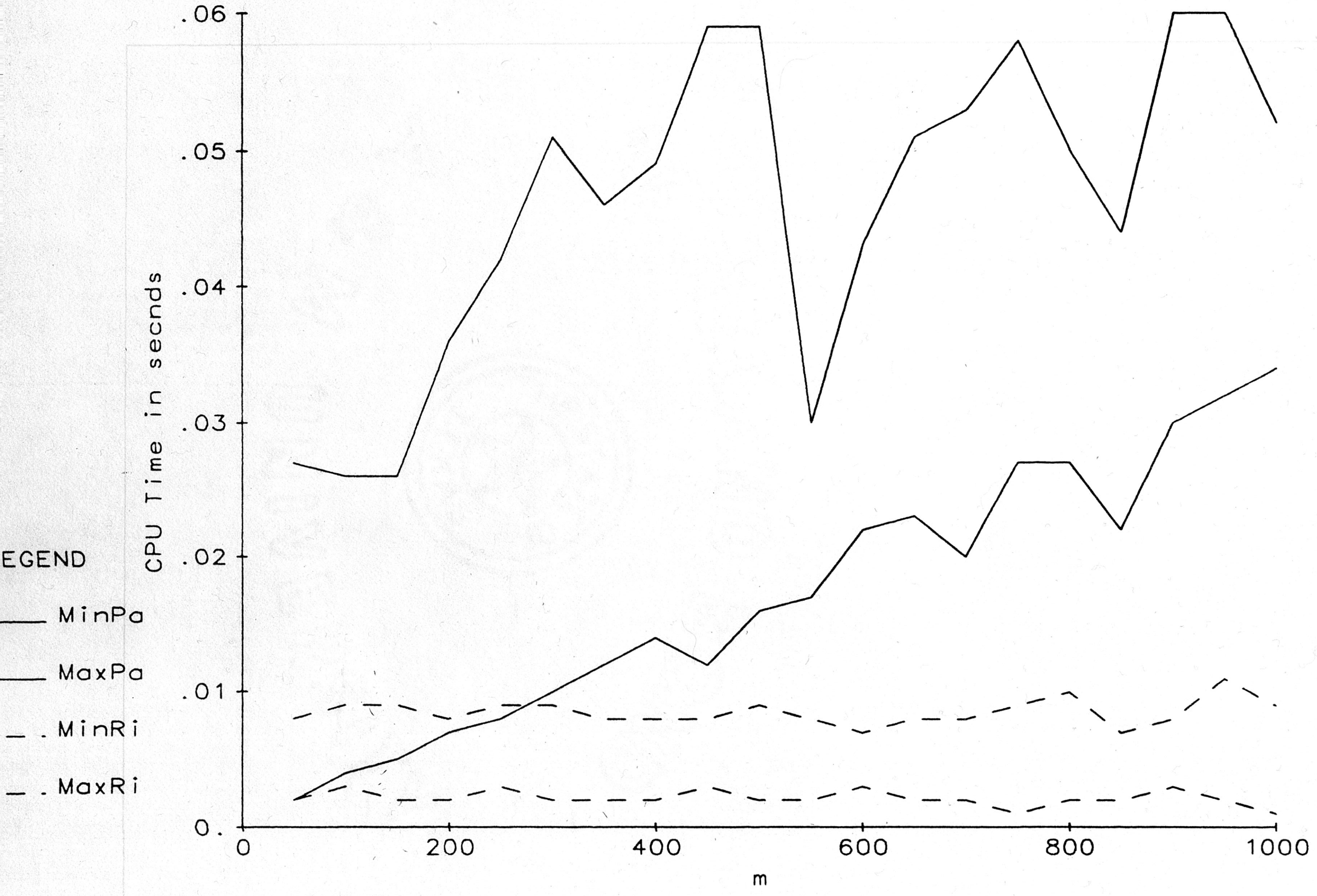


FIG 11 REAL*8/G CHERNOFF 10^-24 ITER 2

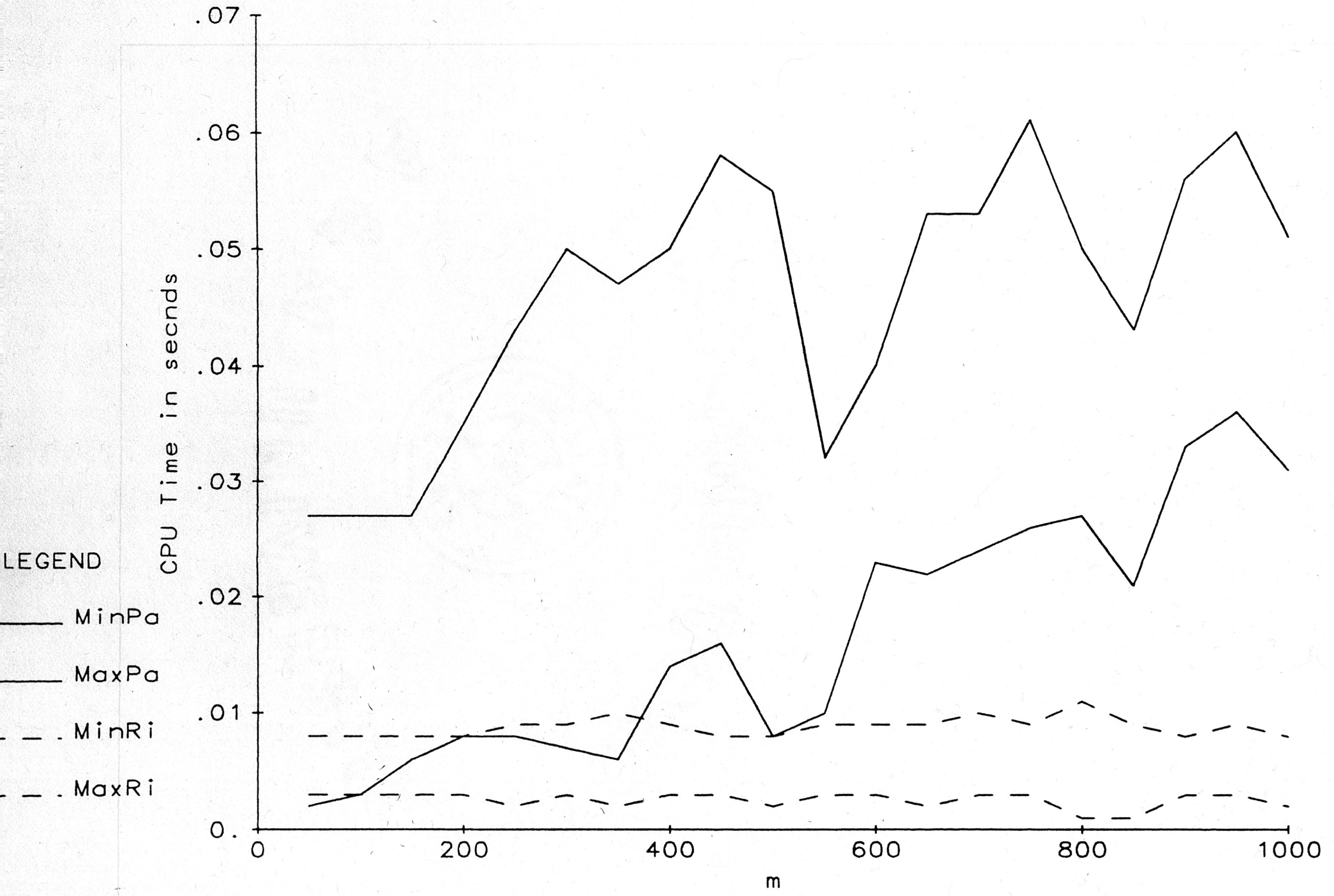


FIG 1**2** REAL*8/G CHERNOFF 10^-24 ITER 3

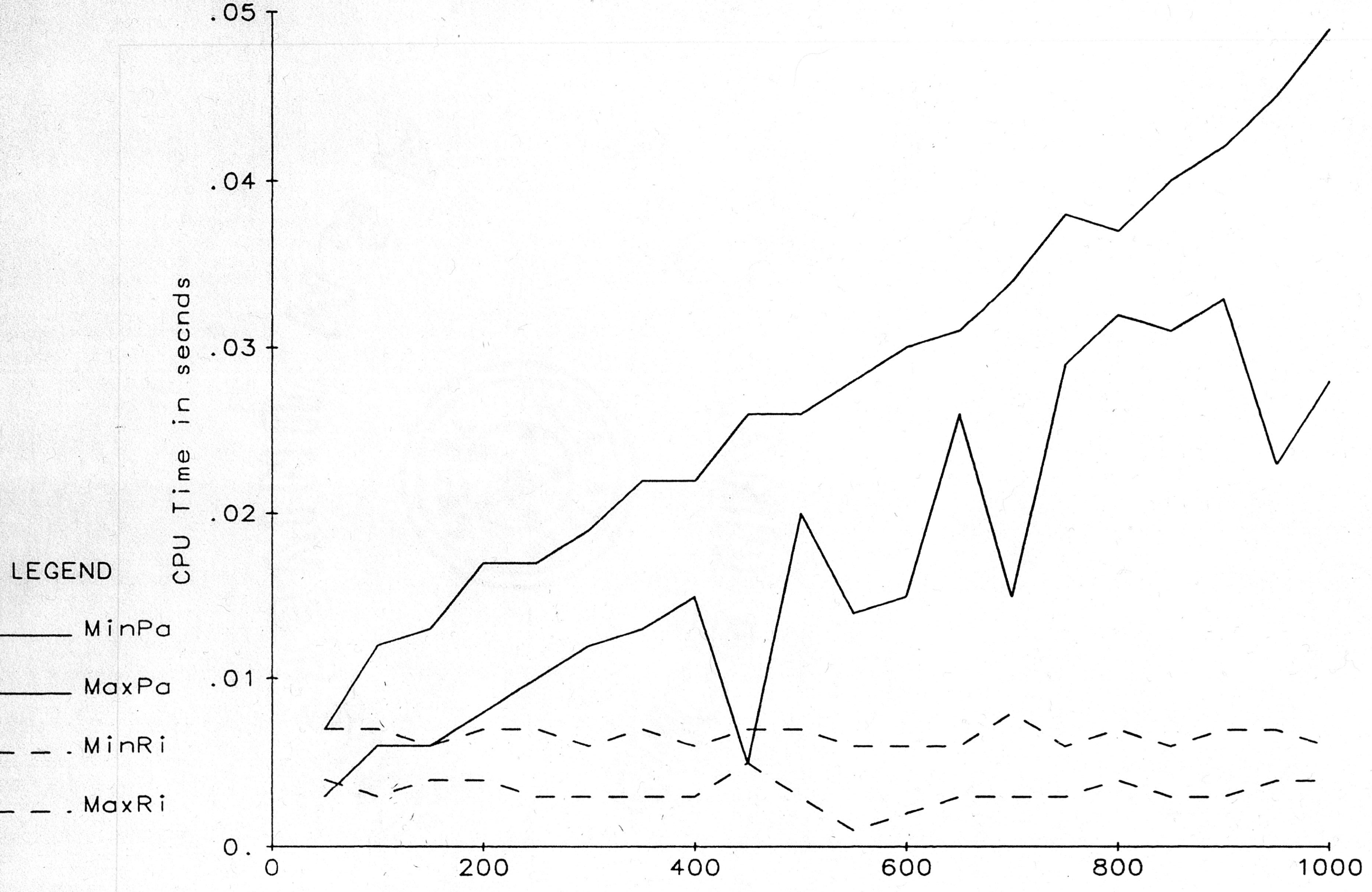


FIG 13 REAL*8/G PDET ITER 1

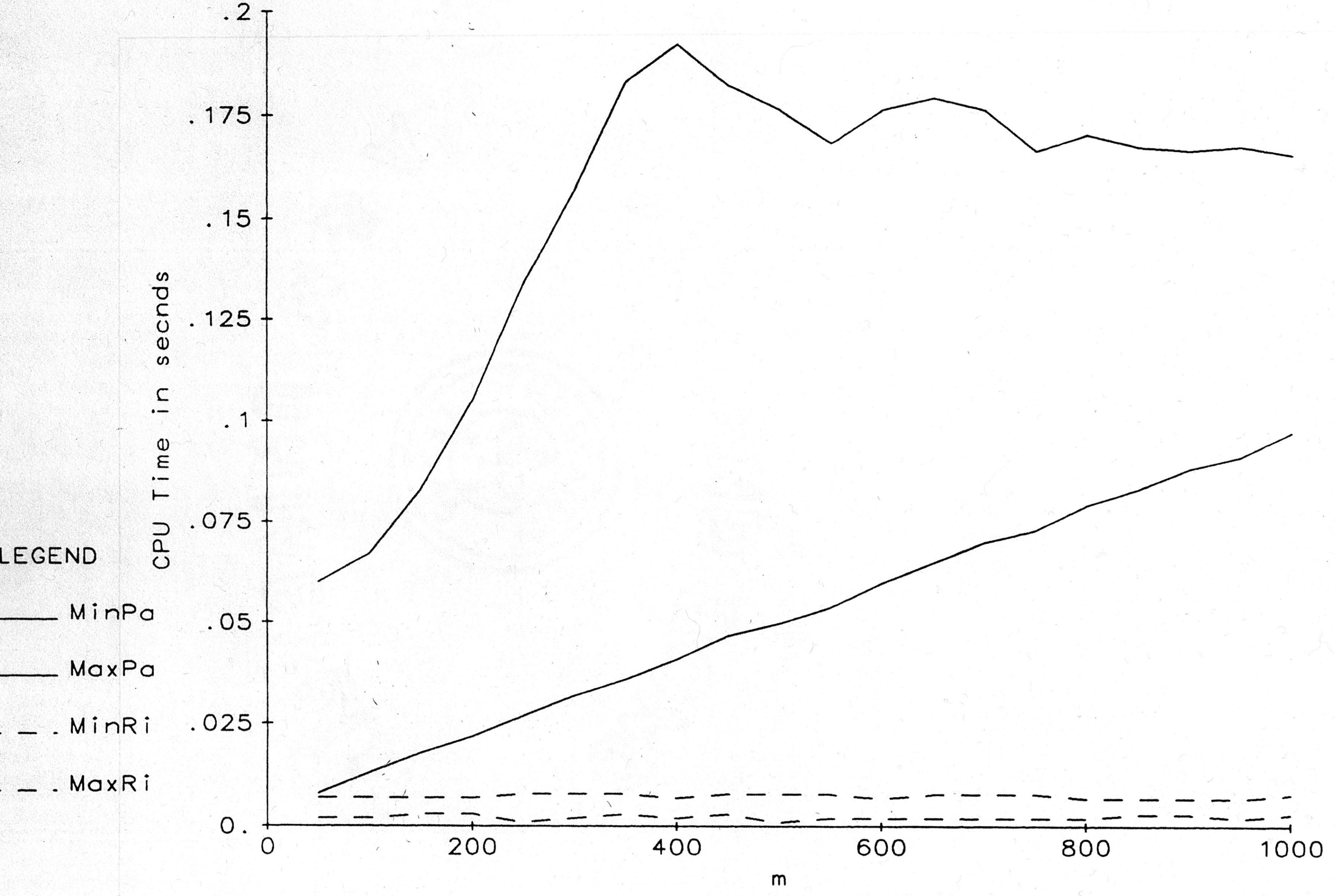


FIG 15 REAL*16/H CHERNOFF 10^-24 ITER 2

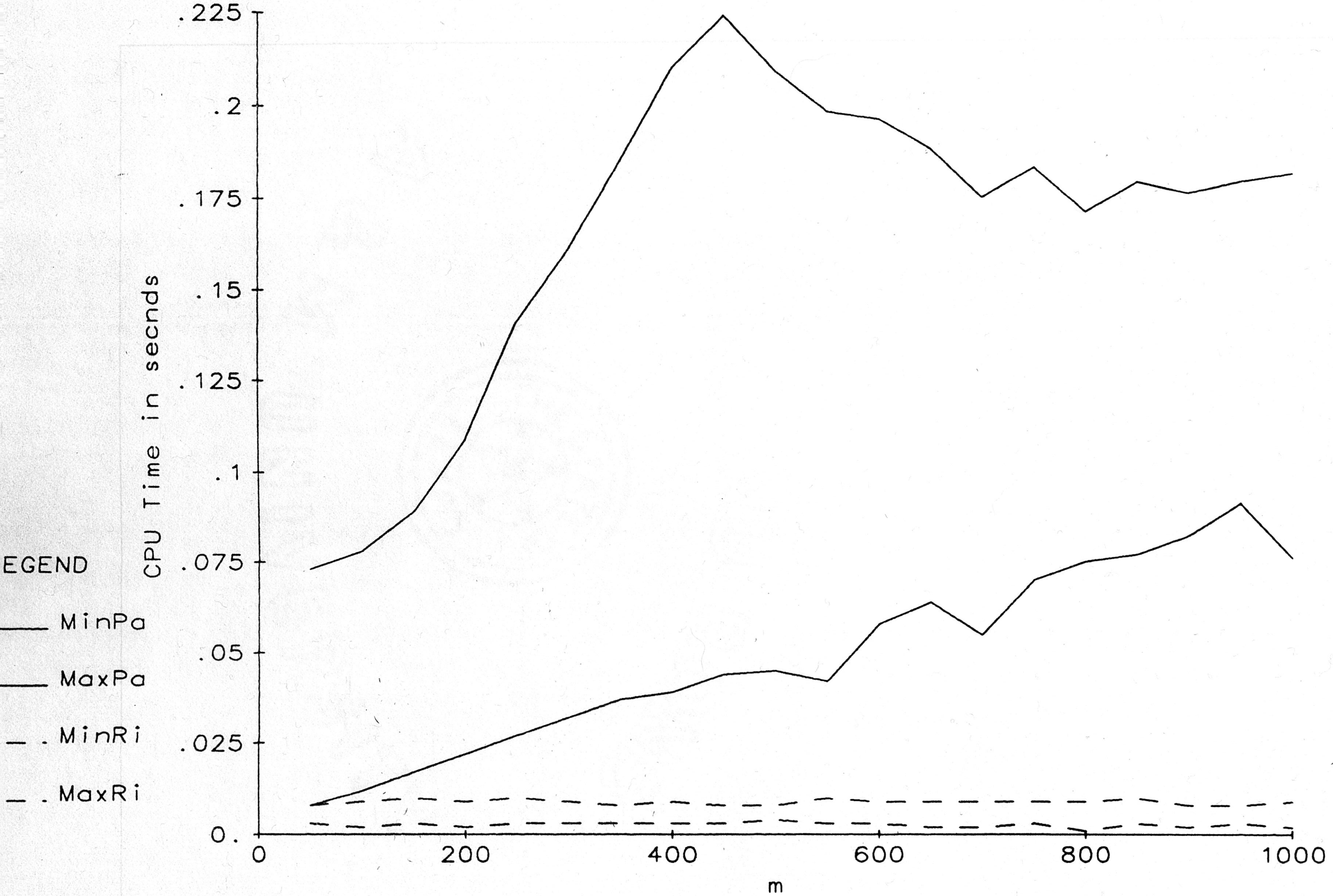


FIG 16 REAL*16/H CHERNOFF 10^-24 ITER 3

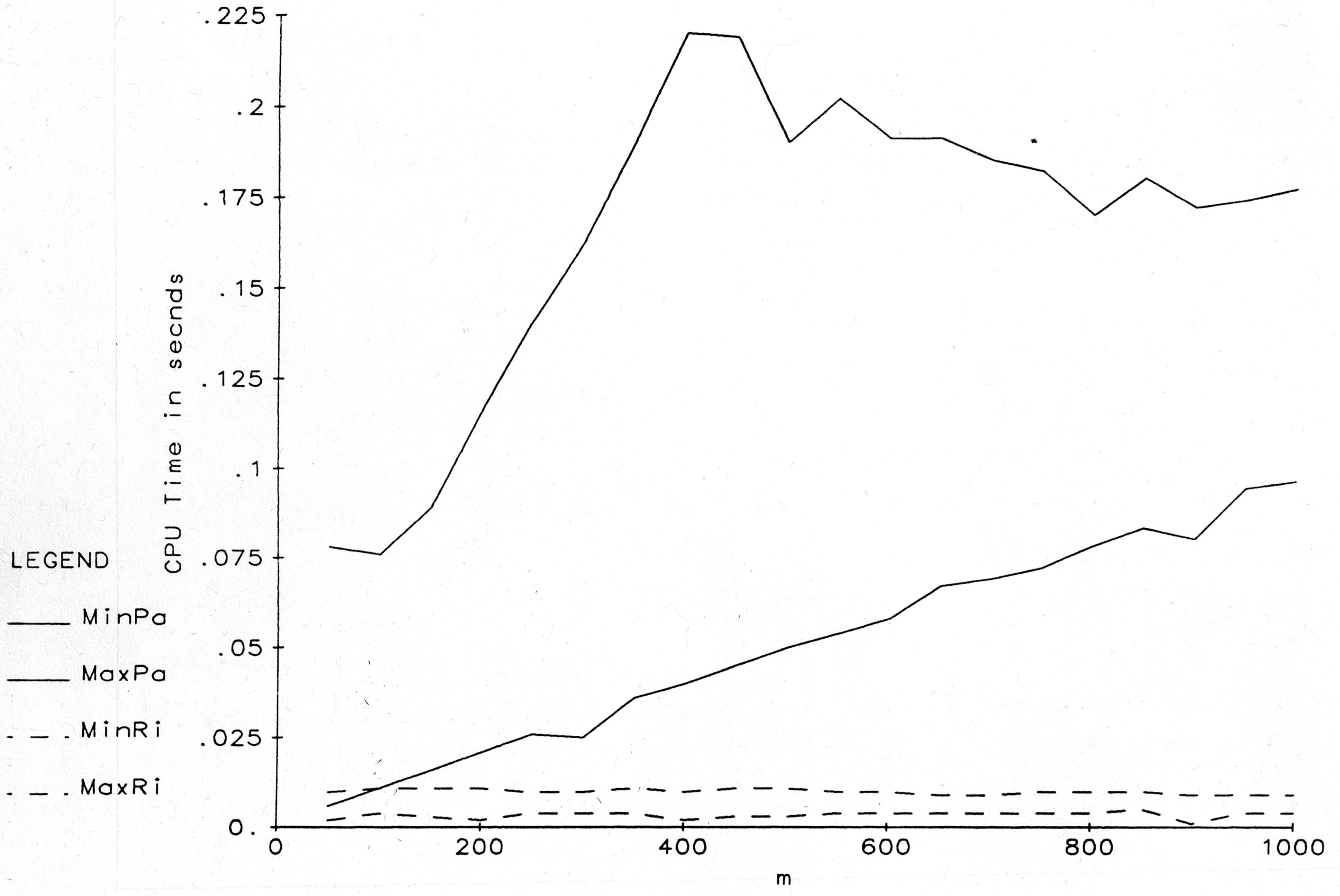


FIG 17 REAL*16/H CHERNOFF 10^-24 ITER 4

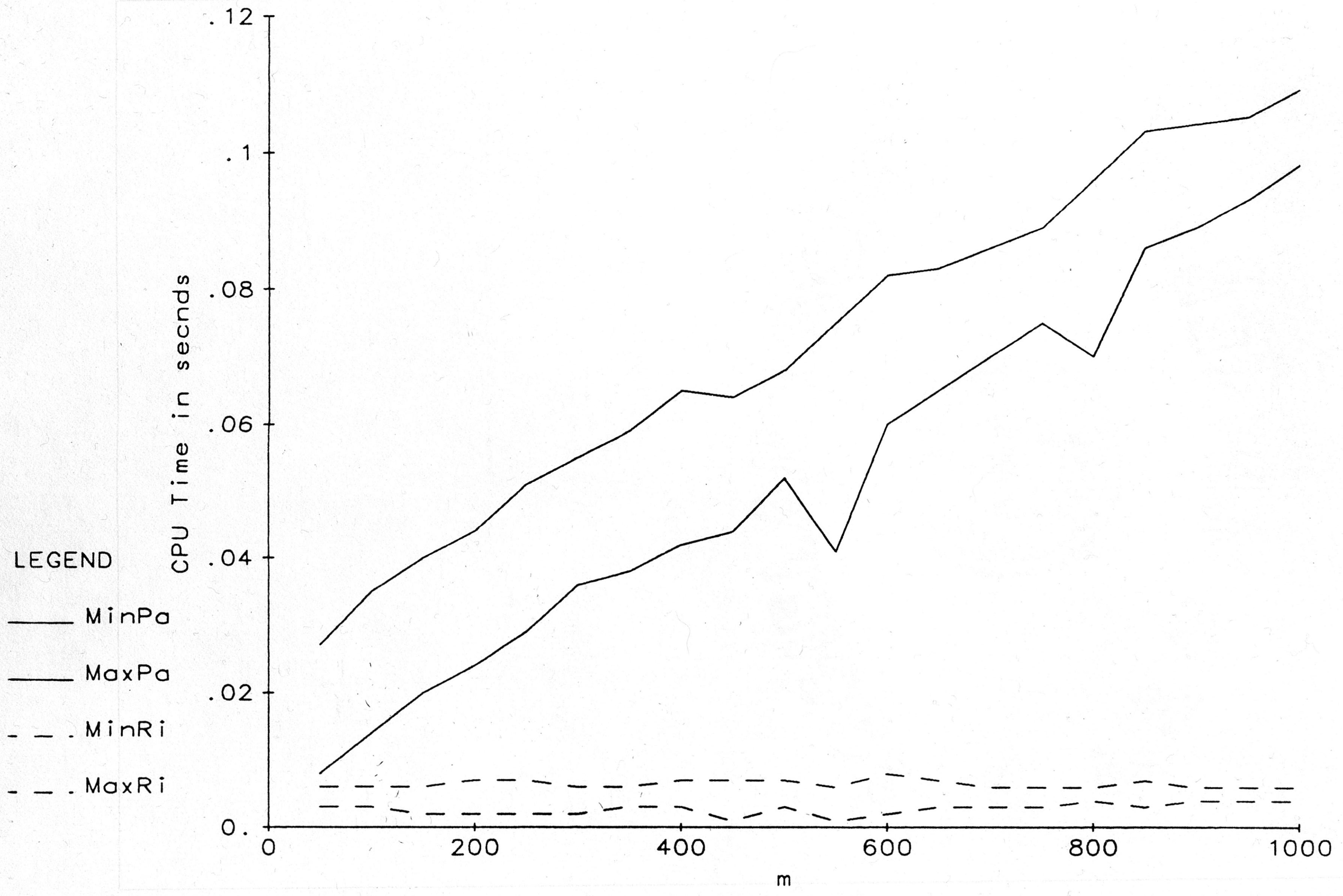


FIG 18 REAL*16/H PDET ITER 1

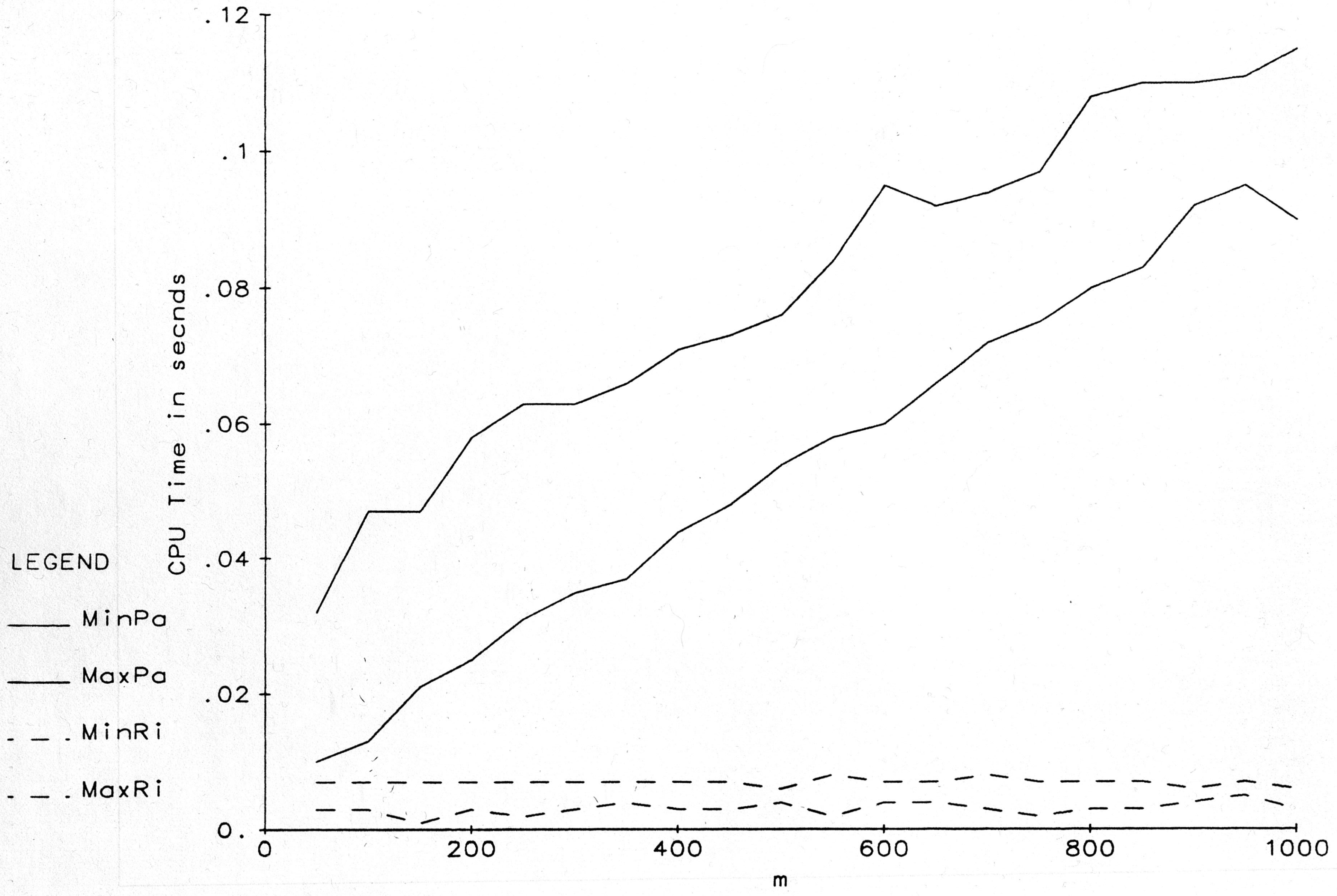


FIG 19 REAL*16/H PDET ITER 2

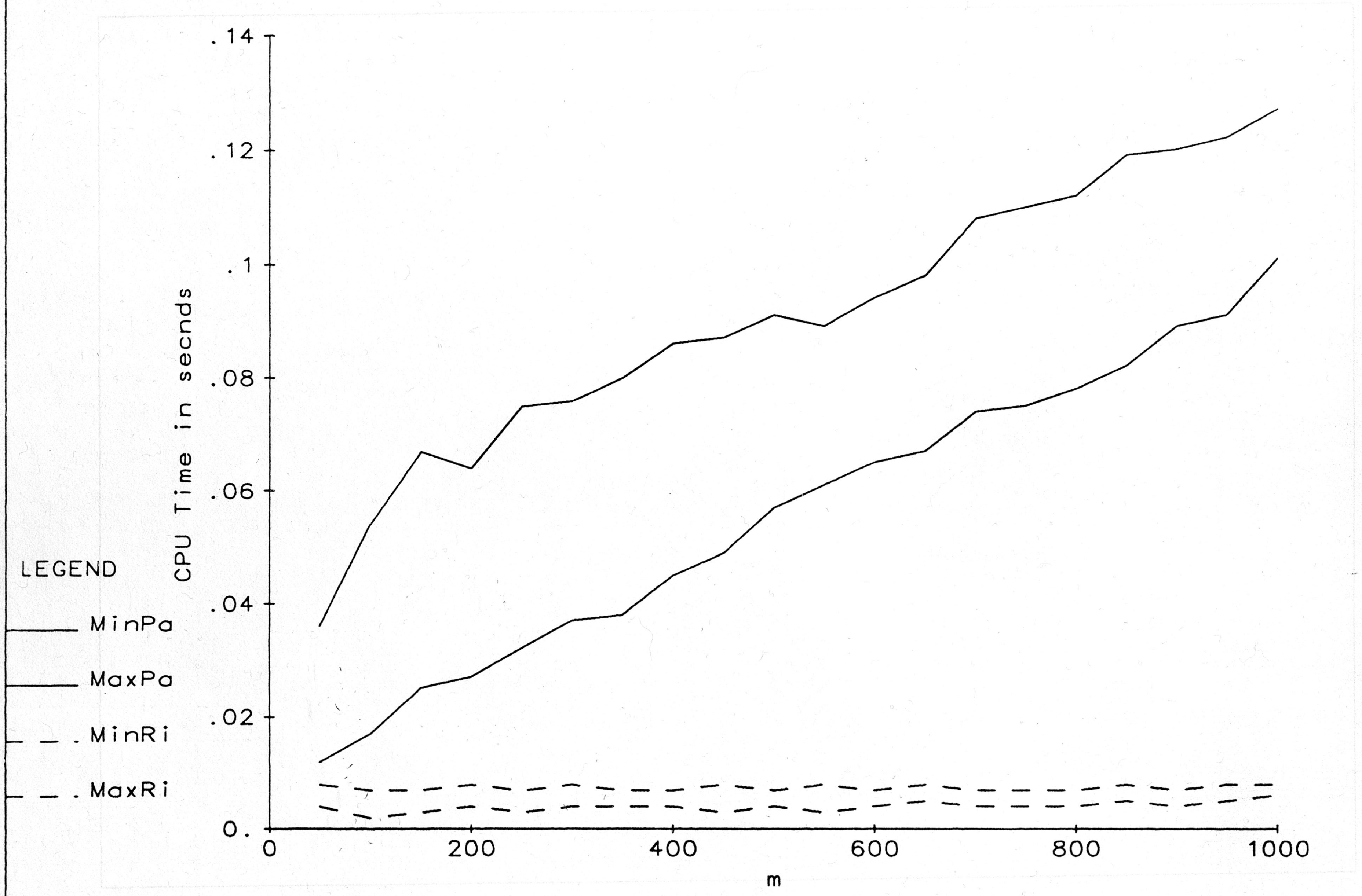


FIG 20 REAL*16/H PDET ITER 3

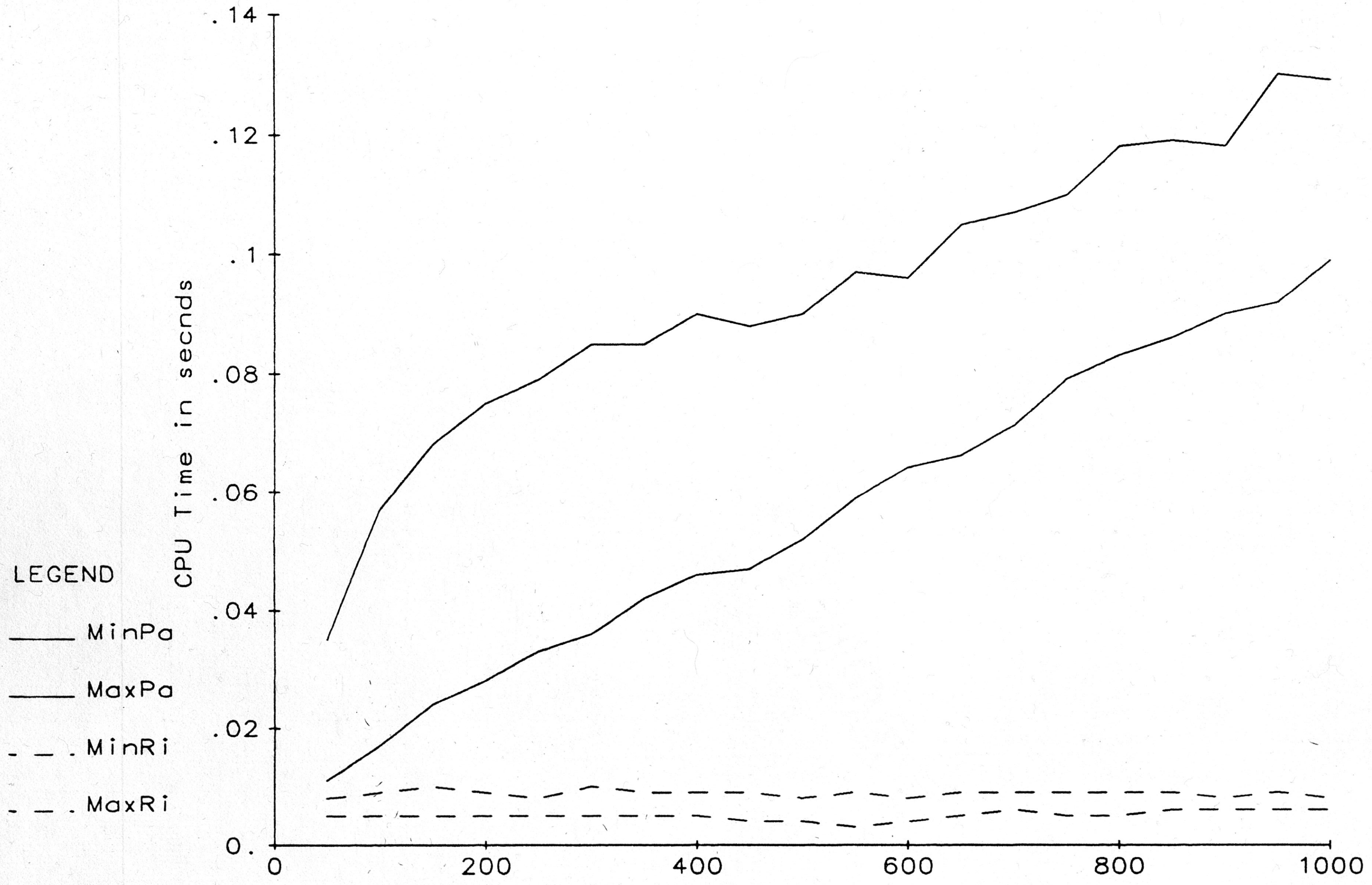


FIG 21 REAL*16/H PDET ITER 4

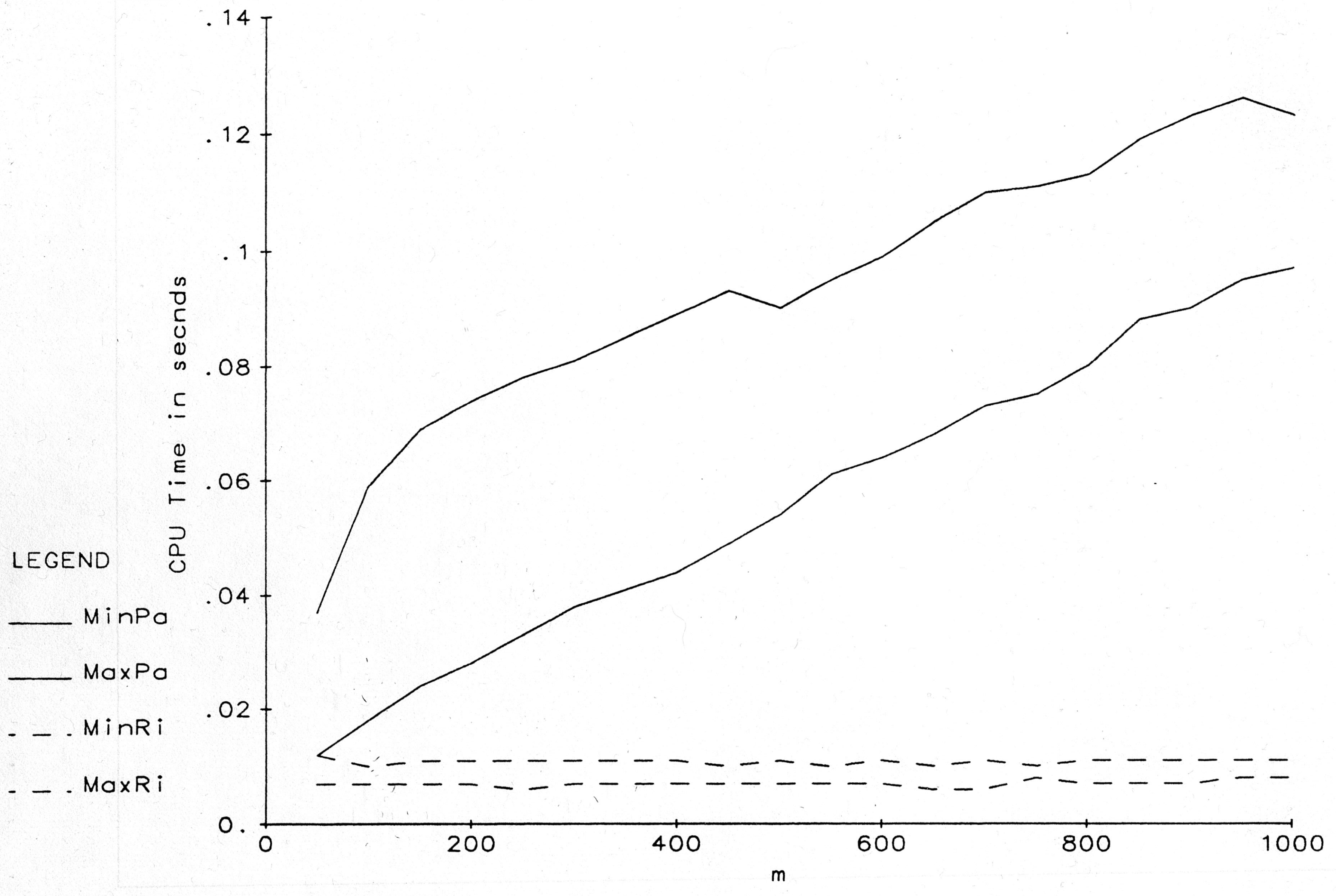


FIG 22 REAL*16/H PDET ITER 5

APPENDIX D

```
C               PROGRAM LISTING:  IMPLEMENTATION OF RICE ALGORITHM

C This is the program implementation of the Rice algorithm.
C It is now configured for REAL*16 operation.  If you want to run it on
C REAL*8/G__floating or /NOG__floating, just change the variable declaration
C that has "Real*16 x,s,r,t1..." to "Real*8 x,s,r,t1...".
C Also change the line that is about 10 lines down that reads,
C "QLOG(1.D0 + T1)..." to "DLOG(1.D0 + T1)...".
C
C
C Inputs:
C
C          H        = m  Order of Q-function
C          Beta     = Second argument in Qm(Alpha,Beta)
C          Alpha    = First argument in Qm(Alpha,Beta)
C          Limit    = Number of iterations for the Rice algorithm
C


C
C Outputs:
C
C          RiceD    = Generalized Q-function
C          RiceCD   = 1 - Qm(Alpha,Beta)


C
C Subroutines called:
C                              ERF      The error function approximation
C                                       subroutine.
C


          Real*8 FUNCTION RiceCD(H,Beta,Alpha,Limit,RiceD)

          Parameter (nmax=101)
          Integer*4 H,Limit,IT,NUM,N,M,K,PTEMP4,EXPAND
          Real*8 Alpha,Beta,RiceD
          Real*16 x,s,r,t1,v1,t11,P,C,PTEMP0,P0,P1,P2,P3,
     1    PTEMP1,PTEMP2,BTEMP1,BTEMP2,BTEMP3,B(0:nmax,0:nmax),
     2    A(0:nmax),SUMC1,SUM1,SUM2,PI,PTEMP3,PT,ERF

C
C Accurately compute PI for use at the end of the program
C

          PI = DACOS(-1.D0)

C
C Refer to Appendix A for explanation of x,s,r
C
C Also see [3, pp. 1989-1991]
C
```

```
                    x = 2.D0 * H
                    s = Beta * Beta / x
                    r = Alpha * Alpha / x
                    t1 = -1.D0 + ((1.D0 + SQRT(1.D0 + 4.D0 * r * s)) /
      2             (2.D0 * s))
                    v1 = (t1 / (ABS(t1))) * SQRT(-0.5D0 * (s * t1 -
      2  QLOG(1.D0 + T1) + r / (1.D0 + t1) - r))
                    t11=2.D0 * SQRT(((1.D0 + t1)**3)/(2.D0 * r + 1.D0 + t1))
                    P = 0.D0
C
C Compute all Ak values
C
                    Do 510 K=1,15
                           A(K) = (2.D0 * ((-1)**(K + 1))) /
      2             ((K + 2.D0) * (1.D0 + t1)**K) * (((K + 2.D0) *
      3             r + 1.D0 + t1) / (2.D0 * r + 1.D0 + t1))
510                 Continue
C
C Flag to check for small t1
C
                IF (ABS(SQRT(ABS(X*(S-1))) - ALPHA) .LE. 0.3) THEN
C
C Refer to Appendix B for explanation of this approximation
C
                    P0=((A(1)-t1*(A(2)+3.D0/4.D0*(A(1)**2)))/2.D0)*t11
                    P1=3.D0/2.D0*(A(3)-A(4)* t1)
                    P1=P1-(15.D0/8.D0*((-2.D0*A(1)*A(2))+(2.D0*A(1)*A(3)+
      2             A(2)**2)*t1))
                    P1=P1+(35.D0/16.D0*((A(1)**3)-3.D0*(A(1)**2)*A(2)*t1))
                    P1=P1-(945.D0/384.D0*(A(1)**4)*t1)
                    P1=(-.5D0*(t11**3))*P1 * (X**-1)
                    P2=(5.D0/2.D0*(A(5)-A(6)*t1))
                    P2=P2-(35.D0/8.D0*(-2.D0*A(1)*A(4)-2.D0*A(2)*A(3)+
      2             t1*((A(3)**2)+2.D0*A(1)*A(5)+2.D0*A(2)*A(4))))
                    P2=P2+(105.D0/16.D0*(3.D0*A(1)*((A(2)**2)+
      2             A(1)*A(3))-(3.D0*A(1)*t1*(A(1)*A(4)+2.D0*A(2)*A(3)))))
                    P2=P2-(3465.D0/384.D0*(-4.D0*(A(1)**3)*A(2)+2.D0*(A(1)**2)*
      2             t1*(3.D0*(A(2)**2)+2.D0*A(1)*A(3))))
                    P2=P2+(45045.D0/3840.D0*((A(1)**5)-5.D0*(A(1)**4)*A(2)*t1))
                    P2=P2-(675675.D0/46080.D0*(A(1)**6)*t1)
                    P2=(.75*(t11**5))*P2 * (X**-2)
                    P3=(7.D0/2.D0*A(7))
                    P3=P3+(63.D0/4.D0*(A(1)*A(6)+A(2)*A(5)+A(3)*A(4)))
                    P3=P3+(693.D0/16.D0*((A(1)**2)*A(5)+2.D0*A(1)*A(2)*A(4)+
      2                    A(1)*(A(3)**2)+(A(2)**2)*A(3)))
                    P3=P3+(9009.D0/96.D0*((A(1)**3)*A(4)+2.D0*(A(1)**2)*A(2)*A(3)+
      2                    A(1)*A(2)*A(3)+A(1)*(A(2)**3)))
         P3=P3+(135135.D0/768.D0*((A(1)**4)*A(3)+2.D0*(A(1)**3)*(A(2)**2)))
                    P3=P3+(2297295.D0/7680.D0*((A(1)**5)*A(2)))
                    P3=P3+(43648605.D0/645120.D0*(A(1)**7))
                    P3=(-15.D0/8.D0*(t11**7))*P3 * (X**-3)
         P4=9.D0/2.D0*(t1*(-a(10))+(a(9)))
         P4=P4-99.D0/8.D0*(t1*(2*a(1)*a(9)+2*a(2)*a(8)+2*a(3)*a(7)
      1  +2*a(4)*a(6)+a(5)^2)+(-2*a(1)*a(8)-2*a(2)*a(7)-2*a(3)*a(6)
      2  -2*a(4)*a(5)))
      P4=P4+1287.D0/36.D0*(t1*(-3*a(1)^2*a(8)-6*a(1)*a(2)*a(7)
     1  -6*a(1)*a(3)*a(6)-3*a(2)^2a(6)-6*a(1)*a(4)*a(5)-6*a(2)*a(3)*a(5)
     2  -3*a(2)*a(4)^2-3*a(3)^2*a(4))+(3*a(1)^2*a(7)+6*a(1)*a(2)*a(6)
     3  +6*a(1)*a(3)*a(5)+3*a(2)^2*a(5)+3*a(1)*a(4)^2+6*a(2)*a(3)*a(4)+a(3)^3))
      P4=P4-19305.D0/384.D0*(t1*(4*a(1)^3*a(7)+12*a(1)^2*a(2)*a(6)
     1  +12*a(1)^2*a(3)*a(5)+12*a(1)*a(2)^2*a(5)+6a(1)^2a(4)^2
     2  +24a(1)a(2)a(3)a(4)+4a(2)^3a(4)+4a(1)a(3)^3+6*a(2)^2*a(3)^2)+
     3  (-4*a(1)^3*a(6)-12*a(1)^2*a(2)*a(5)-12*a(1)^2*a(3)*a(4)
     4  -12*a(1)*a(2)^2*a(4)-12*a(1)*a(2)*a(3)^2-4*a(2)^3*a(3)))
      P4=P4+328185.D0/3840.D0*(t1*(-5*a(1)^4*a(6)-20*a(1)^3*a(2)*a(5)
     1  -20*a(1)^3*a(3)*a(4)-30*a(1)^2*a(2)^2*a(4)-30*a(1)^2*a(2)*a(3)^2
     2  -20*a(1)*a(2)^3*a(3)-a(2)^5)+(+5*a(1)^4*a(5)+20*a(1)^3*a(2)*a(4)
     3  +10*a(1)^3*a(3)^2+30*a(1)^2*a(2)^2*a(3)+5*a(1)*a(2)^4))
      P4=P4-6235515.D0/46080.D0*(t1*(6*a(1)^5*a(5)+30*a(1)^4*a(2)*a(4)
     1  +15*a(1)^4*a(3)^2+60*a(1)^3*a(2)^2*a(3)+15*a(1)^2*a(2)^4)
     2  +(-6*a1^5*a4-30*a1^4*a2*a3-20*a1^3*a2^3))
      P4=P4+130945815.D0/645120.D0*(t1*(-7*a(1)^6*a(4)-42*a(1)^5*a(2)*a(3)
     1  -35*a(1)^4*a(2)^3)
     2  +(7*a(1)^6*a(3)+21*a(1)^5*a(2)^2))
      P4=P4+3011753745.D0/10321920.D0*(t1*(8*a(1)^7*a(3)+28*a(1)^6*a(2)^2)
     1  -8*a(1)^7a(2))
      P4=P4-7.5293843E10/185794560.D0*(t1*(-9*a(1)^8*a(2))+(a(1)^9))
      P4=P4+2.0329337E12/3715891200.D0*t1*(a(1))^10
      P4=(105.D0/16.D0*(t11**9))*P4 * (X**-4)
      P5=11.D0/2.D0*(t1*(-a(12))+(a(11)))
      P5=P5-143.D0/8.D0*(t1*(2*a(3)*a(9)+2*a(4)*a(8)+2*a(5)*a(7)
     1  +a(6)^2+2*a(10)*a(2)+2*a(1)*a(11))+(-2*a(2)*a(9)-2*a(3)*a(8)
     2  -2*a(4)*a(7)-2*a(5)*a(6)-2a(1)a(10)))
      P5=P5+2145.D0/36.D0*(t1*(-6*a(1)*a(2)*a(9)-6*a(1)*a(3)*a(8)
     1  -3*a(2)^2*a(8)-6*a(1)*a(4)*a(7)-6*a(2)*a(3)*a(7)-6*a(1)*a(5)*a(6)
     2  -6*a(2)*a(4)*a(6)-3*a(3)^2*a(6)-3*a(2)*a(5)^2-6*a(3)*a(4)*a(5)
     3  -a(4)^3-3*a(1)^2*a(10))+(3*a(1)^2*a(9)-6*a(1)*a(2)*a(8)
     4  +6*a(1)*a(3)*a(7)+3*a(2)^2*a(7)+6*a(1)*a(4)*a(6)+6*a(2)*a(3)*a(6)
     5  +3*a(1)*a(5)^2+6*a(2)*a(4)*a(5)+3*a(3)^2*a(5)+3*a(3)*a(4)^2))
      P5=P5-36465.D0/384.D0*t1*(4*a(1)^3*a(9)+12*a(1)^2*a(2)*a(8)
     1  +12a(1)^2*a(3)*a(7)+12*a(1)*a(2)^2*a(7)+12*a(1)^2*a(4)*a(6)
     2  +24*a(1)*a(2)*a(3)*a(6)+4*a(2)^3*a(6)+6*a(1)^2*a(5)^2
     3  +24*a(1)*a(2)*a(4)*a(5)+12*a(1)*a(3)^2*a(5)+12*a(2)*a(3)^2*a(5)
     4  +12*a(1)*a(3)*a(4)^2+6*a(2)^2*a(4)^2+12*a(2)*a(3)^2*a(4)+a(3)^4)
      P5=P5+692835.D0/3840.D0*(t1*(-5*a(1)^4*a(8)-20*a(1)^3*a(2)*a(7)
     1  -20*a(1)^3*a(3)*a(6)-a(1)^2*a(2)^2*a(6)-20*a(1)^3*a(4)*a(5)
     2  -60*a(1)^2*a(2)*a(3)*a(5)-20*a(1)*a(2)^3*a(5)-30*a(1)^2*a(2)*a(4)^2
     3  -30*a(1)^2*a(3)^2*a(4)-60*a(1)*a(2)^2*a(3)*a(4)-5*a(2)^4*a(4)
     4  -20*a(1)*a(2)*a(3)^3-10*a(2)^3*a(3)^2)+(5*a(1)^4*a(7)
     5  +20*a(1)^3*a(2)*a(6)+20*a(1)^3*a(3)*a(5)+30*a(1)^2*a(2)^2*a(5)
     6  +10*a(1)^3*a(4)^2+60*a(1)^2*a(2)*a(3)*a(4)+20*a(1)*a(2)^3*a(4)
     7  +10*a(1)^2*a(3)^3+30*a(1)*a(2)^2*a(3)^2+5*a(2)^4*a(3)))
      P5=P5-14549535.D0/46080.D0*(t1*(6*a(1)^5*a(7)+30*a(1)^4*a(2)*a(6)
     1  +30*a(1)^4*a(3)*a(5)+60*a(1)^3*a(2)^2*a(5)+15*a(1)^4*a(4)^2
     2  +120*a(1)^3*a(2)*a(3)*a(4)+60*a(1)^2*a(2)^3*a(4)+20*a(1)^3*a(3)^3
     3  +90*a(1)^2*a(2)^2*a(3)^2+30*a(1)*a(2)^4*a(3)+a(2)^6)+
     4  (-6*a(1)^5*a(6)-30*a(1)^4*a(2)*a(5)-30*a(1)^4*a(3)*a(4)
     5  -60*a(1)^3*a(2)^2*a(4)-60*a(1)^3*a(2)*a(3)^2-60*a(1)^2*a(2)^3*a(3)
     6  -6*a(1)*a(2)^5))
      P5=P5+334639305.D0/645120.D0*(t1*(-7*a(1)^6*a(6)-42*a(1)^5*a(2)*a(5)
     1   -42*a(1)^5*a(3)*a(4)-105*a(1)^4*a(2)^2*a(4)-105*a(1)^4*a(2)*a(3)^2
     2   -140*a(1)^3*a(2)^3*a(3)-21*a(1)^2*a(2)^5)+(7*a(1)^6*a(5)
     3   +42*a(1)^5*a(2)*a(4)+21*a(1)^5*a(3)^2
     4   +105*a(1)^4*a(2)^2*a(3)+35*a(1)^3*a(2)^4))
         P5=P5-8365982625.D0/10321920.D0*(t1*(8*a(1)^7*a(5)+56*a(1)^6*a(2)*a(4)
     1   +28*a(1)^6*a(3)^2+168*a(1)^5*a(2)^2*a(3)+70*a(1)^4*a(2)^4)+
     2   (-8*a(1)^7*a(4)-56*a(1)^6*a(2)*a(3)-56*a(1)^5*a(2)^3))
         P5=P5+2.2588153E11/185794560*(t1*(-9*a(1)^8*a(4)-72*a1^7*a2*a3
     1   -84*a1^6*a2^3)+(9*a1^8*a3+36*a1^7*a2^2))
         P5=P5-6.5505643E12/3715891200*(t1*(10*a(1)^9*a(3)+45*a(1)^8*a(2)^2)+
     1   (-10*a(1)^9*a(2)))
         P5=P5+2.0306749E14/8.1749606E10*(t1*(-11*a(1)^10*a(2))+(a(1)^11))
         P5=P5-6.7012273E15/1.9619905E12*t1*(a(1))^12
         P5=-(945.D0/32.D0*(t11**11))*P5 * (X**-5)
                 P=P0 + P1 + P3 + P4 + P5
                 GOTO 580
             ENDIF
                 PTEMP3 = v1 * t11 / t1
                 Do 520 IT=0,LIMIT
                 C = 1.D0
                 PTEMP4 = 2 * IT + 1
                 Do 530 NUM=0,IT
                         IF (NUM .EQ. 0) Go To 530
                         C = C * (0.5 + NUM - 1)
530              Continue
                 PTEMP0 = 0.D0
                 PTEMP1 = 0.D0
                 Do 540 N=0,2*IT
                   BTEMP3 = 0.D0
                   B(0,N) = 0.D0
                   B(0,0) = 1.D0
                   Do 550 M=0,N
                     BTEMP1=0.D0
                     Do 560 K=1,N-M+1
                         BTEMP1 = BTEMP1 + K * A(K) * B(M,N-K+1)
560                  Continue

                   B(M+1,N+1) = (1.D0 / (N + 1)) * BTEMP1
                   BTEMP2 = 1.D0
                   Do 570 NUM=0,M
                         IF (NUM .EQ. 0) Go To 570
                         BTEMP2 = BTEMP2 * (IT + 0.5 + NUM - 1)
570                Continue

                   BTEMP3 = BTEMP3 + B(M,N) * BTEMP2
550              Continue
                 PTEMP0 = PTEMP0 + ((-t1)**N) * BTEMP3
540              Continue
                 PTEMP1 = (PTEMP3**PTEMP4) * PTEMP0
                 PT = ((((-1)**IT) * C) / (v1**PTEMP4))
                 PT = PT * PTEMP1 - PT
                 PT = PT * (x**(-IT))
                 P = P + PT
520              Continue
580              PTEMP2 = ERF(ABS(v1 * SQRT(x)))
```

```
            IF (V1 .EQ. 0) THEN EXPAND=1
            PTEMP2 = (v1 / ABS(v1)) * PTEMP2
            SUMC1 = 0.5D0 * (1.D0 - PTEMP2)
            SUM1 = 0.5D0 * (1.D0 + PTEMP2)
            SUM2 = (1.0 / (2 * SQRT(PI * X))) * EXP(-X *
     2                (v1**2)) * P
            RiceCD = SUMC1 + SUM2
            RiceD = SUM1 - SUM2
         RETURN
       END
```

```
C                       APPENDIX E
C
C      PROGRAM LISTING:  IMPLEMENTATION OF ERROR FUNCTION APPROXIMATION
C
C This program uses Chebyshev Expansions for calculating the
C Error function.  See reference by:
C Schonfelder, J.L.,Chebyshev Expansions for the Error and
C                    ----------------------------------
C      Related Functions, Mathematics of Computation,Vol. 32,
C      ---------------
C      Number 144, October 1978, Page 1232-1240.
C
      REAL*16 FUNCTION ERF(D)
      IMPLICIT NONE
      REAL*16 D
      REAL*16 A(72),Y,T,X,TR
      INTEGER*4 I
C
C CONSTANTS FOR 0 <= X <= 2
C
      A(1)    =     +1.483110564084803581889448079057E+0
      A(2)    =      -3.01071073386594942470731046311E-1
      A(3)    =       +6.8994830689831566246603180718E-2
      A(4)    =       -1.3916271264722187682546525687E-2
      A(5)    =        +2.420799522433463662891678239E-3
      A(6)    =         -3.65863968584808644649382577E-4
      A(7)    =          +4.8620984432319048282887568E-5
      A(8)    =           -5.749256558035684835054215E-6
      A(9)    =            +6.11324357843476469706758E-7
      A(10)   =             -5.8991015312958434390846E-8
      A(11)   =              +5.207009092068648240455E-9
      A(12)   =              -4.23297587996554326810E-10
      A(13)   =               +3.1881135066491749748E-11
      A(14)   =                -2.236155018832684273E-12
      A(15)   =                 +1.46732984799108492E-13
      A(16)   =                    -9.044001985381747E-15
      A(17)   =                     +5.25481371547092E-16
      A(18)   =                      -2.8874261222849E-17
      A(19)   =                       +1.504785187558E-18
      A(20)   =                         -7.4572892821E-20
      A(21)   =                          +3.522563810E-21
      A(22)   =                           -1.58944644E-22
      A(23)   =                             +6.864365E-24
      A(24)   =                              -2.84257E-25
      A(25)   =                               +1.1306E-26
      A(26)   =                                 -4.33E-28
      A(27)   =                                  +1.6E-29
      A(28)   =                                  -1.0E-30
C
C CONSTANTS FOR X > 2
C
      A(29)   =     +1.077977852072383151168335910348E+0
      A(30)   =       -2.6559890409148673372146500904E-2
      A(31)   =        -1.487073146698099509605046333E-3
      A(32)   =         -1.38040145414143859607708920E-4
```

```
              DO 20 I=1, 28
                   TR = QCOS((I-1)*T)
                   Y = Y + A(I)*TR
                   IF (I .EQ. 1) Y=Y/2
20            CONTINUE
              ERF = X * Y
         ELSE
              T = QACOS((10.5 - X*X) / (2.5 + X*X))
              Y = 0
              DO 30 I=29, 72
                   TR = QCOS((I-29)*T)
                   Y = Y + A(I)*TR
                   IF (I .EQ. 29) Y=Y/2
30            CONTINUE
              ERF = 1.D0 - (EXP(-X**2) * Y / X)
         ENDIF
!       WRITE (5,*) Y
!       GOTO 5
         RETURN
         END
```