Computing Exp (A) and Its Integral

Källström, Claes

1973

COMPUTING EXP(A) AND $\int$ EXP(As) ds

CLAES KÄLLSTRÖM

Report 7309 March 1973
Lund Institute of Technology
Division of Automatic Control

COMPUTING EXP(A) AND ∫ EXP(As)ds

C. Källström

ABSTRACT

This report describes one algorithm to compute exp(A) and one algorithm to compute both exp(A) and ∫exp(As)ds. The method used in the two algorithms is finite series approximation, where the matrix is scaled before the expansion. The choice of the number of terms in the series expansion is discussed in detail, and results of numerical investigations performed on the PDP 15/35 computer are presented. The conclusion is that nine terms is a suitable choice on a computer with a floating point accuracy of 7 - 8 decimal digits.

This work has been supported by the Swedish Board for Technical Development under Contract No. 72-202/U 137.

TABLE OF CONTENTS

1. INTRODUCTION

1.1. Linear Systems

Consider the continuous linear time-invariant system

$$\frac{dx}{dt} = Ax(t) + Bu(t) \tag{1.1}$$

$$y(t) = Cx(t) + Du(t) \tag{1.2}$$

with initial condition

$$x(t_o) = x_o \tag{1.3}$$

where

$x(t)$ is an n-dimensional state vector
$u(t)$ is an r-dimensional control vector
$y(t)$ is a p-dimensional vector of outputs

A, B, C and D are time-invariant matrices of dimension n x n, n x r, p x n and p x r respectively.

The solution of (1.1) with initial condition (1.3) is of the form (see e.g.[23]):

$$x(t) = e^{A(t-t_o)}x_o + \int_{t_o}^{t} e^{A(t-s)}Bu(s)ds \tag{1.4}$$

where the matrix exponential is defined by

$$e^M = I + M + M^2/2! + \ldots = \sum_{i=0}^{\infty} M^i/i! \tag{1.5}$$

Now assume that we observe the output vector at the discrete times t_o, $t_o + \tau$, $t_o + 2\tau, \ldots$, where τ is the sampling interval, and that the control vector is constant over the intervals $(t_o, t_o+\tau)$, $(t_o+\tau, t_o+2\tau), \ldots$, then (1.4) implies that the continuous system (1.1) - (1.2) can be described at the sampling events by following discrete system:

$$x(t+\tau) = e^{A\tau}\ x(t) + \int_{0}^{\tau} e^{As}\ ds\ Bu(t) \tag{1.6}$$

$$y(t) = Cx(t) + Du(t) \tag{1.7}$$

$$t = t_o,\ t_o + \tau,\ t_o + 2\tau, \ldots$$

Introduce the notations

$$\phi = e^{A\tau} \tag{1.8}$$

$$\Gamma = \int_{0}^{\tau} e^{As}\ ds\ B \tag{1.9}$$

where ϕ and Γ obviously are time-invariant matrices of dimension n x n and n x r respectively.

With these notations the discrete system (1.6) - (1.7) can be written:

$$x(t+\tau) = \phi x(t) + \Gamma u(t) \tag{1.10}$$

$$y(t) = Cx(t) + Du(t) \tag{1.11}$$

$$t = t_o,\ t_o + \tau,\ t_o + 2\tau, \ldots$$

Obviously the evaluation of e^A, where A is a square matrix, and the computation of the discrete system matrices ϕ and Γ play an important part in linear system theory.

If the definition of matrix exponential (1.5) is inserted in (1.9), we obtain following series expansion of Γ:

$$\Gamma = \left(I + A\tau/2! + A^2\tau^2/3! + \ldots\right) B\tau = .$$

$$= \left[\sum_{i=0}^{\infty} (A\tau)^i/(i+1)!\right] B\tau \tag{1.12}$$

Multiply (1.12) from left by A and add the matrix B to obtain

$$A\Gamma + B = e^{A\tau} B \tag{1.13}$$

and then

$$A\Gamma = (e^{A\tau} - I)B \tag{1.14}$$

If the inverse A^{-1} exists, it is possible to obtain Γ from $\phi = e^{A\tau}$ by solving the linear equations system (1.14). As the solution of (1.14) is time-consuming and not successful when A is singular, a special method to obtain ϕ and Γ, not using $e^{A\tau}$, is introduced in Chapter 3.

Following two algorithms are treated in this paper:

(1) evaluation of e^A (algorithm 1)
(2) computation of ϕ and Γ (algorithm 2)

1.2. Different Methods of Evaluating Exp(A)

Several methods to evaluate e^A have been tried. The main methods are:

1. Inverse Laplace transformation
2. The Cayley-Hamilton method
3. The eigenvalue/eigenvector method
4. Finite series approximation

Methods 1, 2 and 3 require the computation of the eigenvalues of the matrix A, which may be a difficult computational problem (see e.g. 6). Method 1 is suitable to use for handcomputation. An algorithm using method 2 has been proposed by Fath [7]. If the eigenvalues and eigenvectors of A are desired for other reasons, method 3 may be suitable.

The finite series approximation method seems generally to be the most suitable one for digital computations. The method was first used in the IBM Research Laboratories in the early sixties. R.W. Koepcke wrote an IBM 7090 assembler program which was modified and incorporated in Kalman-Englar's ASP package [10]. A FORTRAN version is found in Åström's Linear Quadratic Package described in [22]. The finite series approximation method is used in algorithm 1 to evaluate e^A, and a modification of the method is used in algorithm 2 to compute ϕ and Γ. The details are discussed in Chapters 2 and 3.

2. EVALUATION OF EXP(A)

2.1. Method of Finite Series Approximation in Evaluating Exp(A)

The matrix exponential e^A is defined by the series

$$e^A = I + A + A^2/2! + \dots = \sum_{i=0}^{\infty} A^i/i! \tag{2.1}$$

In the method of finite series approximation we estimate e^A with

$$\widehat{e^A} = I + A + A^2/2! + \dots + A^N/N! = \sum_{i=0}^{N} A^i/i! \tag{2.2}$$

where N is determined so that the truncation error matrix

$$T = \sum_{i=N+1}^{\infty} A^i/i! \tag{2.3}$$

has suitable small elements.

If A has large elements a large N must be determined. This means a long computing time and a bad accuracy because of the round-off errors introduced by the great number of matrix operations.

To overcome these difficulties we choose a positive integer N and a non-negative integer k so that

$$T = \sum_{i=N+1}^{\infty} (A/2^k)^i/i! \tag{2.4}$$

has suitable small elements.

Now we compute

$$\widehat{e^{A/2^k}} = \sum_{i=0}^{N} (A/2^k)^i/i! \tag{2.5}$$

and then

$$\widehat{e^A} = \left(\cdots\left(\left(\widehat{e^{A/2^k}}\right)^2\right)^2\cdots\right)^2 \tag{2.6}$$

where the number of powers is k.

The difficult problem of choosing suitable N and k will be discussed in detail in Chapter 4.

The convergence of (2.1) is easy to prove. Introduce the vector space $L_n = \{X \mid X \text{ an } n \times n \text{ matrix}\}$. Then L_n is a Banach-space, and the series (2.1) is convergent if it is convergent in the norm.

We have

$$||A^i/i!|| \leq ||A||^i/i! \tag{2.7}$$

The ordinary positive series

$$\sum_{i=0}^{\infty} ||A||^i/i! \tag{2.8}$$

converges for all $||A||$ towards the scalar $e^{||A||}$.

From (2.7) we have that the series

$$\sum_{i=0}^{\infty} ||A^i/i!|| \tag{2.9}$$

is dominated by the series (2.8), and then (2.9) is convergent and finally (2.1) is convergent for all quadratic matrices A. See e.g. [1] and [18].

The convergence of the method (2.5) and (2.6) is discussed in [20]. An analysis of roundoff and truncation errors is given in [21].

2.2. Series Expansion

According to (2.5) we estimate $e^{A/2^k}$ with

$$\widehat{e^{A/2^k}} = \sum_{i=0}^{N} (A/2^k)^i/i! \tag{2.10}$$

We introduce the notation $\overline{A} = A/2^k$. A straightforward computation of (2.10) requires a memory storage of 4 matrices of dimension n x n, namely

1. A

2.

1. $\bar{A}$

2. $\sum_{i=0}^{\ell} \bar{A}^i/i!$

3. $\bar{A}^{\ell}/\ell!$

4. The matrix product $(\bar{A}^{\ell}/\ell!)\ \bar{A}$

where $\ell = 1,\dots, N$

In fact, it is possible to reduce matrix 4 to a vector, where the rows of the resulting matrix $(\bar{A}^{\ell}/\ell!)\bar{A}$ temporarily are stored before they are put into matrix 3.

The number of matrix operations involved in a straightforward computation of (2.10) is:

(N-1) matrix multiplications
(N-1) matrix additions
1 addition of the unity matrix

Another way to evaluate (2.10) is now described. Define matrices B_{ℓ}, $\ell = 0,\dots, N$ as follows:

$$B_o = I \tag{2.11}$$
$$B_{\ell} = \bar{A}B_{\ell-1}/(N-\ell+1) + I$$

Obviously we have

$$\widehat{e^{\bar{A}}} = B_N \tag{2.12}$$

Memory storage of 3 matrices of dimensions n x n is required, namely

1. $\bar{A}$
2. B_{ℓ}
3. The matrix product $\bar{A}B_{\ell}$

As in the straightforward computation it is possible to reduce matrix 3 to a vector.

The number of matrix operations involved is:

(N-1) matrix multiplications
N additions of the unity matrix

We conclude that the computing time is less in this case, which has been confirmed by numerical investigations.

Additions of the last terms in the straightforward computation of (2.10) seems to be ill-conditioned numerical operations. The method (2.11) - - (2.12) does not include additions of matrices with such small elements, and the accuracy should be better. This conclusion has been confirmed by numerical investigations.

The last method is obviously to prefer with regard to accuracy, computing time and memory storage.

In fact following modification of (2.11) and (2.12) will decrease the execution time further:

$$\begin{aligned} B_o &= I/N! \\ B_\ell &= \bar{A}B_{\ell-1} + I/(N-\ell)! \\ e^{\bar{A}} &= B_N = \bar{A}B_{N-1} + I \\ \ell &= 1, 2, .., N-1 \end{aligned} \qquad (2.13)$$

However, it is necessary to compute N! explicite, and if this value is too large to be stored in a fix point variable, the computation may be less accurate than (2.11) and (2.12)

2.3. Origin Shift

If the eigenvalues of A are concentrated around a point far away from the origin, it seems reasonable to shift the origin to the centre of mass of the eigenvalues before the evaluation of e^A to get shorter computing time and higher accuracy. See [18]. The sum of the eigenvalues is equal to tr A (trace of A), and consequently the centre is tr(A)/n. We then have

$$A = (A - (tr(A)/n)I) + (tr(A)/n)I \qquad (2.14)$$

and

$$e^A = e^{(A - (tr(A)/n)I) + (tr(A)/n)I} \qquad (2.15)$$

But the matrices

$A - (tr(A)/n)I$

and

$(tr(A)/n)I$

commutate, which implies that (2.15) can be written

$$e^{A} = e^{(A - (tr(A)/n)I)}\, e^{(tr(A)/n)I} \tag{2.16}$$

We have

$$e^{(tr(A)/n)I} = e^{tr(A)/n} I \tag{2.17}$$

and (2.16) can be written

$$e^{A} = (e^{tr(A)/n})\, e^{(A - (tr(A)/n)I)} \tag{2.18}$$

where $e^{tr(A)/n}$ is a scalar.

The origin shift requires, of course, that it is possible to obtain an accurate value of e^{a}, where a is a scalar.

The origin shift seems to be a reasonable way of increasing the accuracy and decreasing the computing time. In some applications, however, the matrix A is known to have its eigenvalues centered around the origin. It is therefore desirable for the user to be able to control if origin shift will be performed or not.

The difference in accuracy and computing time between using origin shift and not is further discussed in Chapter 6.

2.4. Algorithm 1

Now we sum up the conclusions we have made in sections 2.1, 2.2 and 2.3 in an algorithm to evaluate e^{A}:

1. Put S = A if no origin shift or compute tr(A)/n and put S = A - - (tr(A)/n)I if origin shift.

2. Choose N and k so that the truncation error

$$T = \sum_{i=N+1}^{\infty} (S/2^{k})^{i}/i!$$

is suitable small.

3. Compute $\bar{S} = S/2^{k}$.

4. Put $B_{o} = I$ and compute

$$B_{1} = \bar{S}\, B_{o}/N + I$$
$$\vdots$$
$$B_{\ell} = \bar{S}\, B_{\ell-1}/(N-\ell+1) + I$$
$$\vdots$$
$$\widehat{e^{\bar{S}}} = B_{N} = \bar{S}\, B_{N-1} + I$$

5. Compute

$$\widehat{e^{S}} = \left(\cdots\left(\left(\widehat{e^{\bar{S}}}\right)^{2}\right)^{2}\cdots\right)^{2}$$

where the square is taken k times.

6. Put

$\widehat{e^{A}} = \widehat{e^{S}}$ if no origin shift

or compute

$\widehat{e^{A}} = e^{tr(A)/n}\, \widehat{e^{S}}$

if origin shift.

The difficult problem of choosing suitable N and k is discussed in Chapter 4.

3. SAMPLING A LINEAR SYSTEM

In section 1.1 we have introduced the definitions ((1.8) and (1.9)):

$$\phi = e^{A\tau} \tag{3.1}$$

$$\Gamma = \int_0^\tau e^{As}\,ds\cdot B \tag{3.2}$$

where ϕ and Γ are matrices of dimension n x n and n x r respectively, and τ is the sampling interval. As shown in section 1.1 it would be possible to compute $\phi = e^{A\tau}$ by algorithm 1 given in section 2.4, and then solve (1.14) to get Γ. As this method is time-consuming and does not work if A is singular, another approach of computing ϕ and Γ is introduced in this chapter.

3.1. Method of Finite Series Approximation in Sampling a Linear System

In section 1.1 we have shown (1.12):

$$\Gamma = \left(\sum_{i=0}^{\infty} (A\tau)^i/(i+1)!\right) B\tau \tag{3.3}$$

We now define the series expansion

$$\Psi(A) = \sum_{i=0}^{\infty} A^i/(i+1)! \tag{3.4}$$

and write (3.3) as:

$$\Gamma = \Psi(A\tau)B\tau \tag{3.5}$$

The convergence of (3.4) is proved analogous to the convergence of (2.1) in section 2.1. Only note that

$$||A^i/(i+1)!|| \leq ||A||^i/(i+1)! \leq ||A||^i/i! \tag{3.6}$$

and the convergence of

$$\sum_{i=0}^{\infty} ||A||^i/i! \tag{3.7}$$

implies that (3.4) is convergent.

From the definition of matrix exponential (1.5) and (3.4) we can state:

$$e^{A} = A\Psi(A) + I = \Psi(A)A + I \tag{3.8}$$

The method is now obvious. Evaluate $\Psi(A\tau)$ according to (3.4) and then compute:

$$\phi = (A\tau)\Psi(A\tau) + I \tag{3.9}$$

$$\Gamma = \Psi(A\tau)(B\tau) \tag{3.10}$$

As in section 2.1 we choose a positive integer N and a non-negative integer k so that

$$T = \sum_{i=N+1}^{\infty} (A/2^{k})^{i}/(i+1)! \tag{3.11}$$

has suitable small elements, where T is the truncation error matrix.

Then we estimate $\Psi(A/2^{k})$ with

$$\hat{\Psi}(A/2^{k}) = \sum_{i=0}^{N} (A/2^{k})^{i}/(i+1)! \tag{3.12}$$

To obtain $\hat{\Psi}(A)$ from $\hat{\Psi}(A/2^{k})$ is somewhat more complicated than in the exponential case.

Put A/2 into equation (3.8) to obtain

$$e^{A/2} = (A/2)\Psi(A/2) + I \tag{3.13}$$

Square (3.13) and use (3.8):

$$e^{A} = \left((A/2)\Psi(A/2) + I\right)^{2} = A\Psi(A) + I \, , \tag{3.14}$$

Develope the square and use that A/2 and $\Psi(A/2)$ commutate to get:

$$A\left(\Psi(A) - (A/4)\Psi^{2}(A/2) - \Psi(A/2)\right) = 0 \tag{3.15}$$

This equality has to be satisfied for any matrix A, and we obtain:

$$\Psi(A) = \left((A/4)\Psi(A/2) + I\right)\Psi(A/2) \tag{3.16}$$

We can obtain $\hat{\Psi}(A)$ from $\hat{\Psi}(A/2^k)$ by writing (3.16) as

$$\hat{\Psi}(A/2^{\ell-1}) = \left(\frac{1}{2}(A/2^{\ell})\hat{\Psi}(A/2^{\ell}) + I\right)\hat{\Psi}(A/2^{\ell}) \tag{3.17}$$

and use this relation for $\ell = k, k-1, \ldots, 1$.

To compute $\hat{\Psi}(A/2^{\ell-1})$ from $\hat{\Psi}(A/2^{\ell})$ requires

2. matrix multiplications
1 addition of the unity matrix

while the corresponding relation to (3.17) in the exponential case, (2.6), only requires one matrix multiplication.

3.2. Series Expansion

According to (3.12) we estimate $\Psi(A/2^k)$ with

$$\hat{\Psi}(A/2^k) = \sum_{i=0}^{N} (A/2^k)^i/(i+1)! \tag{3.18}$$

Again we use $\bar{A} = A/2^k$. It is possible to evaluate (3.18) in a straightforward way, but analogous to section 2.2 we use another approach.

Define matrices C_ℓ, $\ell = 0, 1, \ldots, N$ as follows:

$$\begin{aligned} C_o &= I \\ C_\ell &= \bar{A}C_{\ell-1}/(N-\ell+2) + I \end{aligned} \tag{3.19}$$

Obviously we have

$$\hat{\Psi}(\bar{A}) = C_N \tag{3.20}$$

This method is quite analogous to the method (2.11) - (2.12) described in section 2.2, and the advantages are quite the same.

3.3. Algorithm 2

In section 2.3 is the origin shift in evaluating e^A described. Application of the same origin shift in evaluating $\Psi(A)$ implies the solving of a linear equations system, and the advantages of the shift thus have disappeared.

We now sum up sections 3.1 and 3.2 in an algorithm to compute ϕ and Γ:

1. Choose N and k so that the truncation error matrix

$$T = \sum_{i=N+1}^{\infty} (A\tau/2^k)^i/(i+1)!$$

has suitable small elements.

2. Compute $\bar{A} = A\tau/2^k$.

3. Put $C_o = I$ and compute

$$C_1 = \bar{A}C_o/(N+1) + I$$

$$\vdots$$

$$C_\ell = \bar{A}C_{\ell-1}/(N-\ell+2) + I$$

$$\vdots$$

$$\hat{\Psi}(\bar{A}) = C_N = \bar{A}C_{N-1}/2 + I$$

4. Compute $\hat{\Psi}(A\tau)$ by using

$$\hat{\Psi}(A\tau/2^{\ell-1}) = \left(\frac{1}{2}\,(A\tau/2^\ell)\hat{\Psi}(A\tau/2^\ell) + I\right)\hat{\Psi}(A\tau/2^\ell)$$

for $\ell = k, k-1, .., 1$.

5. Compute

$$\phi = (A\tau)\hat{\Psi}(A\tau) + I$$
$$\Gamma = \hat{\Psi}(A\tau)(B\tau)$$

The difficult problem of choosing suitable N and k is discussed in next chapter.

4. CHOICE OF THE NUMBER OF TERMS AND THE SCALE FACTOR

In this chapter the choice of the number of terms and the scale factor $1/2^k$ in the two algorithms given in Sections 2.4 and 3.3 will be discussed.

It would be desirable to find an N with regard to the accuracy of the computer only, and then choose a k dependent on the actual matrix A. Numerical investigations will show that this is a reasonable approach to obtain efficient algorithms.

A simple consideration of the number of terms for a computer with eight significant decimal digits is given in Kalman-Englar's ASP package [10]. They decided to use N = 36, but the results in this chapter will show that this number is too large.

The investigations are performed on the computer PDP 15/35. The mantissa of a real single precision variable occupies 27 bits including a sign bit, which means that the accuracy is between 7 and 8 decimal digits. Matrix multiplications, however, are partly performed in double precision. The double precision accuracy is about 10 decimal digits.

4.1. Algorithm 1

Define the relative error of $\widehat{e^A}$ as

$$\delta(\widehat{e^A}) = \frac{||\widehat{e^A} - e^A||}{||e^A||} \tag{4.1}$$

where the norm of A is taken as

$$||A|| = \min\{\max_i \sum_{j=1}^{n} |a_{ij}|, \max_j \sum_{i=1}^{n} |a_{ij}|\} \tag{4.2}$$

See 3].

The relative error of $\widehat{e^A}$ as a function of N and k has been computed on PDP 15/35 for four kinds of test matrices.

Testmatrix 1:

$A = a$

where a is a scalar.

Obviously we get

$e^A = e^a$

Testmatrix 2:

$$A = \begin{bmatrix} 0 & a \\ -a & 0 \end{bmatrix}$$

We obtain (see e.g. [3]):

$$e^A = \begin{bmatrix} \cos a & \sin a \\ -\sin a & \cos a \end{bmatrix} \qquad (4.3)$$

Testmatrix 3:

If Λ is an n by n matrix and T is any nonsingular n by n matrix, then following can be proved (see e.g. [3]):

$$e^{T\Lambda T^{-1}} = Te^{\Lambda} T^{-1} \qquad (4.4)$$

We choose

$$T = \begin{bmatrix} n & n-1 & n-2 & \dots & 2 & 1 \\ n-1 & n-1 & n-2 & \dots & 2 & 1 \\ n-2 & n-2 & n-2 & \dots & 2 & 1 \\ & & \vdots & & & \\ 2 & 2 & 2 & \dots & 2 & 1 \\ 1 & 1 & 1 & \dots & 1 & 1 \end{bmatrix} \qquad (4.5)$$

and then we get (see [8]):

$$T^{-1} = \begin{bmatrix} 1 & -1 & & & & & \\ -1 & 2 & -1 & & 0 & & \\ & -1 & 2 & -1 & & & \\ & & & \cdots & & & \\ & 0 & & & -1 & 2 & -1 \\ & & & & & -1 & 2 \end{bmatrix} \qquad (4.6)$$

Λ is chosen as the diagonal matrix:

$$\Lambda = \begin{bmatrix} 1 & & & & 0 \\ & 2 & & & \\ & & 3 & & \\ & & & \ddots & \\ 0 & & & & n \end{bmatrix} \qquad (4.7)$$

Obviously we obtain from the definition of matrix exponentials:

$$e^{\Lambda} = \begin{bmatrix} e & & & & 0 \\ & e^2 & & & \\ & & e^3 & & \\ & & & \ddots & \\ 0 & & & & e^n \end{bmatrix} \qquad (4.8)$$

Now testmatrices

$$A = T\Lambda T^{-1} \qquad (4.9)$$

and exponential matrices

$$e^A = Te^{\Lambda}T^{-1} \qquad (4.10)$$

easily can be generated with various order n.

<u>Testmatrix 4</u>:

Define the n by n matrix

$$E = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ 1 & 1 & \cdots & 1 \\ & & \vdots & \\ 1 & 1 & \cdots & 1 \end{bmatrix} \qquad (4.11)$$

If we put

$$T = aE + bI \tag{4.12}$$

it is easy to show that

$$T^{-1} = cE + dI \tag{4.13}$$

where

$$c = - \frac{a}{b} \quad \frac{1}{an + b} \tag{4.14}$$

$$d = \frac{1}{b} \tag{4.15}$$

Put a = b and choose Λ according to (4.7), and then testmatrices of various order can be generated analoguous to (4.9) and (4.10). It appears that the value of the parameter a = b does not influence the matrices A and e^A.

Some results of the numerical investigations are shown in Tables 4.1 - 4.5. The computational effort to evaluate e^A can roughly be estimated by the number of matrix multiplications M. We obviously have

$$M = N + k -1 \tag{4.16}$$

which means, of course, that the same number of matrix multiplications M can be obtained by several combinations of N and k. For each M = 1,2,3,.. the minimum relative error is underlined in the Tables 4.1 - 4.5.

```
MATRIX A
  -8.0




RELATIVE ERROR OF EXP(A)

 *
  * N    2         3         4         5         6         7         8         9         10        11        12        13        14
K  ******************************************************************************************************************************
   *
0  *   0.7E+05 0.2E+06 0.3E+06 0.5E+06 0.6E+06 0.6E+06 0.6E+06 0.5E+06 0.4E+06 0.3E+06 0.2E+06 0.1E+06 0.5E+05
1  *   0.7E+05 0.1E+06 0.7E+05 0.4E+05 0.1E+05 0.4E+04 0.8E+03 0.1E+03 0.3E+02 0.8E+00 0.1E+01 0.2E+00 0.7E-01
2  *   0.3E+04 0.4E+02 0.4E+02 0.9E+00 0.7E+00 0.1E+00 0.4E-01 0.7E-02 0.1E-02 0.2E-03 0.3E-04 0.5E-05 0.1E-05
3  *   0.1E+02 0.5E+00 0.2E+00 0.3E-01 0.4E-02 0.5E-03 0.5E-04 0.6E-05 0.5E-06 0.2E-06 0.2E-06 0.2E-06 0.2E-06
4  *   0.6E+00 0.6E-01 0.6E-02 0.5E-03 0.4E-04 0.2E-05 0.1E-06 0.1E-06 0.1E-06 0.1E-06 0.1E-06 0.1E-06 0.1E-06
5  *   0.1E+00 0.6E-02 0.3E-03 0.1E-04 0.5E-06 0.4E-06 0.4E-06 0.4E-06 0.4E-06 0.4E-06 0.4E-06 0.4E-06 0.4E-06
6  *   0.2E-01 0.7E-03 0.2E-04 0.2E-05 0.4E-06 0.4E-06 0.4E-06 0.4E-06 0.4E-06 0.4E-06 0.4E-06 0.4E-06 0.4E-06
7  *   0.5E-02 0.9E-04 0.4E-06 0.3E-05 0.4E-06 0.4E-06 0.4E-06 0.4E-06 0.4E-06 0.4E-06 0.4E-06 0.4E-06 0.4E-06
8  *   0.1E-02 0.1E-04 0.5E-05 0.5E-05 0.5E-05 0.5E-05 0.5E-05 0.5E-05 0.5E-05 0.5E-05 0.5E-05 0.5E-05 0.5E-05
9  *   0.3E-03 0.7E-05 0.4E-06 0.4E-06 0.4E-06 0.4E-06 0.4E-06 0.4E-06 0.4E-06 0.4E-06 0.4E-06 0.4E-06 0.4E-06
```

Table 4.1

Testmatrix 1 with a = -8

```
MATRIX A
   6.4



RELATIVE ERROR OF EXP(A)

 *
  * N    2         3         4         5         6         7         8         9        10        11        12        13        14
K  *******************************************************************************************************************************
   *
0  *  0.1E+01  0.9E+00  0.8E+00  0.6E+00  0.5E+00  0.3E+00  0.2E+00  0.1E+00  0.6E-01  0.3E-01  0.1E-01  0.6E-02  0.3E-02
1  *  0.9E+00  0.6E+00  0.4E+00  0.2E+00  0.9E-01  0.3E-01  0.1E-01  0.4E-02  0.1E-02  0.3E-03  0.6E-04  0.1E-04  0.3E-05
2  *  0.6E+00  0.3E+00  0.9E-01  0.2E-01  0.5E-02  0.1E-02  0.2E-03  0.3E-04  0.4E-05  0.5E-06  0.8E-07  0.8E-07  0.8E-07
3  *  0.3E+00  0.7E-01  0.1E-01  0.1E-02  0.2E-03  0.2E-04  0.2E-05  0.8E-07  0.8E-07  0.8E-07  0.8E-07  0.8E-07  0.8E-07
4  *  0.1E+00  0.1E-01  0.1E-02  0.6E-04  0.4E-05  0.3E-06  0.3E-06  0.3E-06  0.3E-06  0.3E-06  0.3E-06  0.3E-06  0.3E-06
5  *  0.4E-01  0.2E-02  0.7E-04  0.3E-05  0.3E-06  0.3E-06  0.3E-06  0.3E-06  0.3E-06  0.3E-06  0.3E-06  0.3E-06  0.3E-06
6  *  0.1E-01  0.2E-03  0.5E-05  0.2E-05  0.2E-05  0.2E-05  0.2E-05  0.2E-05  0.2E-05  0.2E-05  0.2E-05  0.2E-05  0.2E-05
7  *  0.3E-02  0.3E-04  0.3E-05  0.6E-06  0.6E-06  0.6E-06  0.6E-06  0.6E-06  0.6E-06  0.6E-06  0.6E-06  0.6E-06  0.6E-06
8  *  0.7E-03  0.7E-05  0.6E-06  0.6E-06  0.6E-06  0.6E-06  0.6E-06  0.6E-06  0.6E-06  0.6E-06  0.6E-06  0.6E-06  0.6E-06
9  *  0.2E-03  0.7E-05  0.7E-05  0.7E-05  0.7E-05  0.7E-05  0.7E-05  0.7E-05  0.7E-05  0.7E-05  0.7E-05  0.7E-05  0.7E-05
```

Table 4.2

Testmatrix 1 with a = 6.4

```
MATRIX A
   0.0        8.0
  -8.0        0.0




RELATIVE ERROR OF EXP(A)

 *
  * N    2       3       4       5       6       7       8       9      10      11      12      13      14
K *******************************************************************************************************************
   *
0  *   0.3E+02 0.1E+03 0.2E+03 0.3E+03 0.4E+03 0.4E+03 0.4E+03 0.3E+03 0.2E+03 0.2E+03 0.9E+02 0.5E+02 0.3E+02
1  *   0.8E+02 0.9E+02 0.7E+02 0.2E+02 0.8E+01 0.6E+01 0.2E+01 0.5E+00 0.2E+00 0.8E-01 0.2E-01 0.7E-02 0.2E-02
2  *   0.3E+02 0.3E+01 0.7E+00 0.3E+00 0.9E-01 0.2E-01 0.5E-02 0.1E-02 0.2E-03 0.3E-04 0.5E-05 0.7E-06 0.4E-07
3  *   0.2E+01 0.4E+00 0.8E-01 0.1E-01 0.2E-02 0.2E-03 0.3E-04 0.3E-05 0.2E-06 0.4E-07 0.4E-07 0.4E-07 0.4E-07
4  *   0.3E+00 0.4E-01 0.5E-02 0.4E-03 0.3E-04 0.2E-05 0.2E-06 0.4E-07 0.4E-07 0.4E-07 0.4E-07 0.4E-07 0.4E-07
5  *   0.7E-01 0.5E-02 0.2E-03 0.1E-04 0.5E-06 0.3E-06 0.3E-06 0.3E-06 0.3E-06 0.3E-06 0.3E-06 0.3E-06 0.3E-06
6  *   0.2E-01 0.6E-03 0.1E-04 0.1E-06 0.6E-06 0.6E-06 0.6E-06 0.6E-06 0.6E-06 0.6E-06 0.6E-06 0.6E-06 0.6E-06
7  *   0.5E-02 0.8E-04 0.9E-06 0.6E-06 0.6E-06 0.6E-06 0.6E-06 0.6E-06 0.6E-06 0.6E-06 0.6E-06 0.6E-06 0.6E-06
8  *   0.1E-02 0.1E-04 0.8E-06 0.6E-06 0.6E-06 0.6E-06 0.6E-06 0.6E-06 0.6E-06 0.6E-06 0.6E-06 0.6E-06 0.6E-06
9  *   0.3E-03 0.4E-05 0.4E-05 0.4E-05 0.4E-05 0.4E-05 0.4E-05 0.4E-05 0.4E-05 0.4E-05 0.4E-05 0.4E-05 0.4E-05
```

~~Table 4.3~~

Testmatrix 2 with a = 8

```
MATRIX A
 -10.0     2.0     2.0     2.0     2.0     2.0     2.0     2.0     2.0     2.0     2.0     2.0
 -11.0     3.0     2.0     2.0     2.0     2.0     2.0     2.0     2.0     2.0     2.0     2.0
 -10.0     0.0     4.0     2.0     2.0     2.0     2.0     2.0     2.0     2.0     2.0     2.0
  -9.0     0.0     0.0     5.0     2.0     2.0     2.0     2.0     2.0     2.0     2.0     2.0
  -8.0     0.0     0.0     0.0     6.0     2.0     2.0     2.0     2.0     2.0     2.0     2.0
  -7.0     0.0     0.0     0.0     0.0     7.0     2.0     2.0     2.0     2.0     2.0     2.0
  -6.0     0.0     0.0     0.0     0.0     0.0     8.0     2.0     2.0     2.0     2.0     2.0
  -5.0     0.0     0.0     0.0     0.0     0.0     0.0     9.0     2.0     2.0     2.0     2.0
  -4.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0    10.0     2.0     2.0     2.0
  -3.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0    11.0     2.0     2.0
  -2.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0    12.0     2.0
  -1.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0    13.0
```

```
RELATIVE ERROR OF EXP(A)

 *
  * N     2         3         4         5         6         7         8         9        10        11        12        13        14
 K ********************************************************************************************************************************
   *
 1 *   0.1E+01  0.1E+01  0.9E+00  0.8E+00  0.7E+00  0.5E+00  0.3E+00  0.2E+00  0.9E-01  0.5E-01  0.2E-01  0.9E-02  0.3E-02
 2 *   0.1E+01  0.8E+00  0.6E+00  0.3E+00  0.1E+00  0.5E-01  0.2E-01  0.5E-02  0.1E-02  0.3E-03  0.8E-04  0.2E-04  0.3E-05
 3 *   0.8E+00  0.4E+00  0.2E+00  0.4E-01  0.8E-02  0.2E-02  0.3E-03  0.4E-04  0.5E-05  0.7E-06  0.1E-06  0.1E-06  0.1E-06
 4 *   0.5E+00  0.1E+00  0.2E-01  0.2E-02  0.3E-03  0.2E-04  0.2E-05  0.1E-06  0.1E-06  0.2E-06  0.2E-06  0.2E-06  0.2E-06
 5 *   0.2E+00  0.2E-01  0.2E-02  0.1E-03  0.6E-05  0.5E-06  0.5E-06  0.5E-06  0.5E-06  0.5E-06  0.5E-06  0.5E-06  0.5E-06
 6 *   0.6E-01  0.3E-02  0.1E-03  0.4E-05  0.6E-06  0.5E-06  0.5E-06  0.5E-06  0.5E-06  0.5E-06  0.5E-06  0.5E-06  0.5E-06
 7 *   0.2E-01  0.4E-03  0.1E-04  0.3E-05  0.2E-05  0.2E-05  0.2E-05  0.2E-05  0.2E-05  0.2E-05  0.2E-05  0.2E-05  0.2E-05
 8 *   0.5E-02  0.6E-04  0.5E-05  0.5E-05  0.5E-05  0.5E-05  0.5E-05  0.5E-05  0.5E-05  0.5E-05  0.5E-05  0.5E-05  0.5E-05
 9 *   0.1E-02  0.1E-04  0.1E-04  0.1E-04  0.1E-04  0.1E-04  0.1E-04  0.1E-04  0.1E-04  0.1E-04  0.1E-04  0.1E-04  0.1E-04
10 *   0.3E-03  0.3E-04  0.3E-04  0.3E-04  0.3E-04  0.3E-04  0.3E-04  0.3E-04  0.3E-04  0.3E-04  0.3E-04  0.3E-04  0.3E-04
```

Table 4.4

Testmatrix 3 with n = 12

```
MATRIX A

   -.21111111+01  -.21111111+01  -.11111111+01  -.11111111+00   .88888889+00   .18888889+01
   -.32222222+01  -.22222221+00  -.12222222+01  -.22222222+00   .77777778+00   .17777778+01
   -.33333333+01  -.23333333+01   .16666667+01  -.33333333+00   .66666666+00   .16666667+01
   -.34444444+01  -.24444444+01  -.14444444+01   .35555556+01   .55555555+00   .15555556+01
   -.35555556+01  -.25555556+01  -.15555556+01  -.55555555+00   .54444444+01   .14444444+01
   -.36666667+01  -.26666667+01  -.16666667+01  -.66666666+00   .33333333+00   .73333333+01
   -.37777778+01  -.27777778+01  -.17777778+01  -.77777778+00   .22222222+00   .12222222+01
   -.38888889+01  -.28888889+01  -.18888889+01  -.88888889+00   .11111111+00   .11111111+01

    .28888889+01   .38888889+01
    .27777778+01   .37777778+01
    .26666667+01   .36666667+01
    .25555556+01   .35555556+01
    .24444444+01   .34444444+01
    .23333333+01   .33333333+01
    .92222222+01   .32222222+01
    .21111111+01   .11111111+02
```

```
RELATIVE ERROR OF EXP(A)

 *
  * N    2       3       4       5       6       7       8       9       10      11      12      13      14
 K ****************************************************************************************************************
   *
 1 *  0.9E+00 0.8E+00 0.5E+00 0.4E+00 0.2E+00 0.1E+00 0.4E-01 0.2E-01 0.6E-02 0.2E-02 0.5E-03 0.2E-03 0.4E-04
 2 *  0.9E+00 0.5E+00 0.2E+00 0.6E-01 0.2E-01 0.4E-02 0.9E-03 0.2E-03 0.3E-04 0.5E-05 0.5E-06 0.3E-06 0.2E-06
 3 *  0.5E+00 0.1E+00 0.3E-01 0.5E-02 0.7E-03 0.8E-04 0.9E-05 0.7E-06 0.2E-06 0.4E-06 0.4E-06 0.4E-06 0.4E-06
 4 *  0.2E+00 0.3E-01 0.3E-02 0.2E-03 0.2E-04 0.6E-06 0.2E-06 0.4E-06 0.4E-06 0.4E-06 0.4E-06 0.4E-06 0.4E-06
 5 *  0.7E-01 0.4E-02 0.2E-03 0.9E-05 0.6E-06 0.2E-06 0.1E-06 0.1E-06 0.1E-06 0.1E-06 0.1E-06 0.1E-06 0.1E-06
 6 *  0.2E-01 0.6E-03 0.1E-04 0.5E-06 0.8E-06 0.8E-06 0.8E-06 0.8E-06 0.8E-06 0.8E-06 0.8E-06 0.8E-06 0.8E-06
 7 *  0.5E-02 0.8E-04 0.2E-05 0.4E-05 0.4E-05 0.4E-05 0.4E-05 0.4E-05 0.4E-05 0.4E-05 0.4E-05 0.4E-05 0.4E-05
 8 *  0.1E-02 0.9E-05 0.2E-05 0.2E-05 0.2E-05 0.2E-05 0.2E-05 0.2E-05 0.2E-05 0.2E-05 0.2E-05 0.2E-05 0.2E-05
 9 *  0.3E-03 0.1E-04 0.1E-04 0.1E-04 0.1E-04 0.1E-04 0.1E-04 0.1E-04 0.1E-04 0.1E-04 0.1E-04 0.1E-04 0.1E-04
10 *  0.1E-03 0.9E-05 0.9E-05 0.9E-05 0.9E-05 0.9E-05 0.9E-05 0.9E-05 0.9E-05 0.9E-05 0.9E-05 0.9E-05 0.9E-05
```

Table 4.5

Testmatrix 4 with n = 8

The minimum relative error δ as a function of M for the five matrices in Tables 4.1 - 4.5 are shown in Fig. 4.1 and 4.2. The optimal number of matrix products, M_{opt}, is easily found for each matrix from the plots. All plots of investigated testmatrices have the same characteristic appearance. The anomaly of the plot in Fig. 4.1 (c) can be explained by going back to Table 4.3. The minimum value of δ for M = 10 is obtained by the combination N = 5 and k = 6. By looking at δ for k = 6 and N = = 6,7,8,.. we can conclude that the small value of δ for N = 5 and k = 6 only is a "lucky hit".

When M_{opt} has been determined from the plot it is quite easy to go back to the table to find the optimal combination of N and k. The results are summarized in Table 4.6.

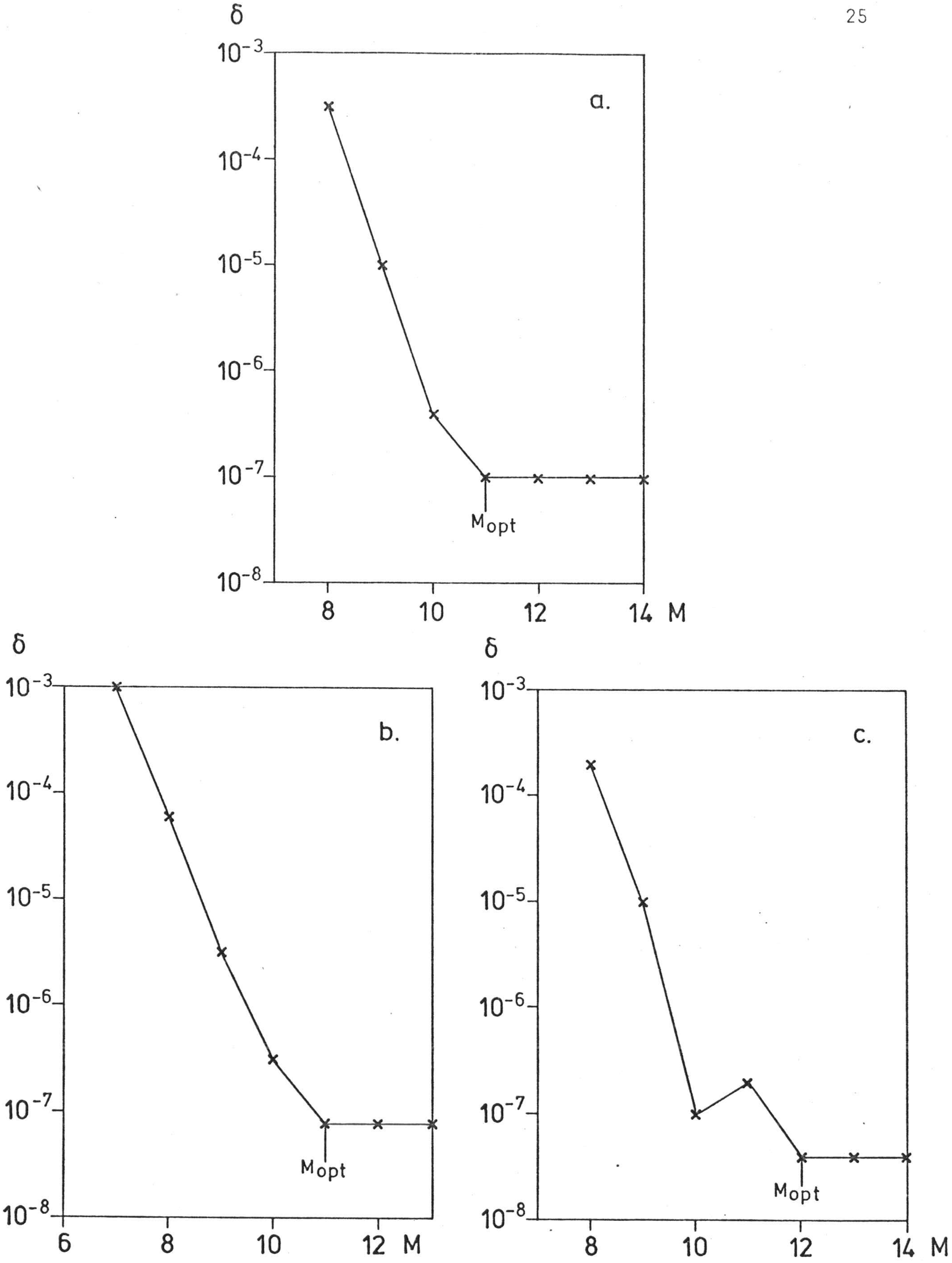


Fig. 4.1 - Minimum relative error δ as function of the number of matrix multiplications M. The δ-axis has logarithmic scale.

a. Test matrix 1 with a = -8.

b. Test matrix 1 with a = 6.4.

c. Test matrix 2 with a = 8.

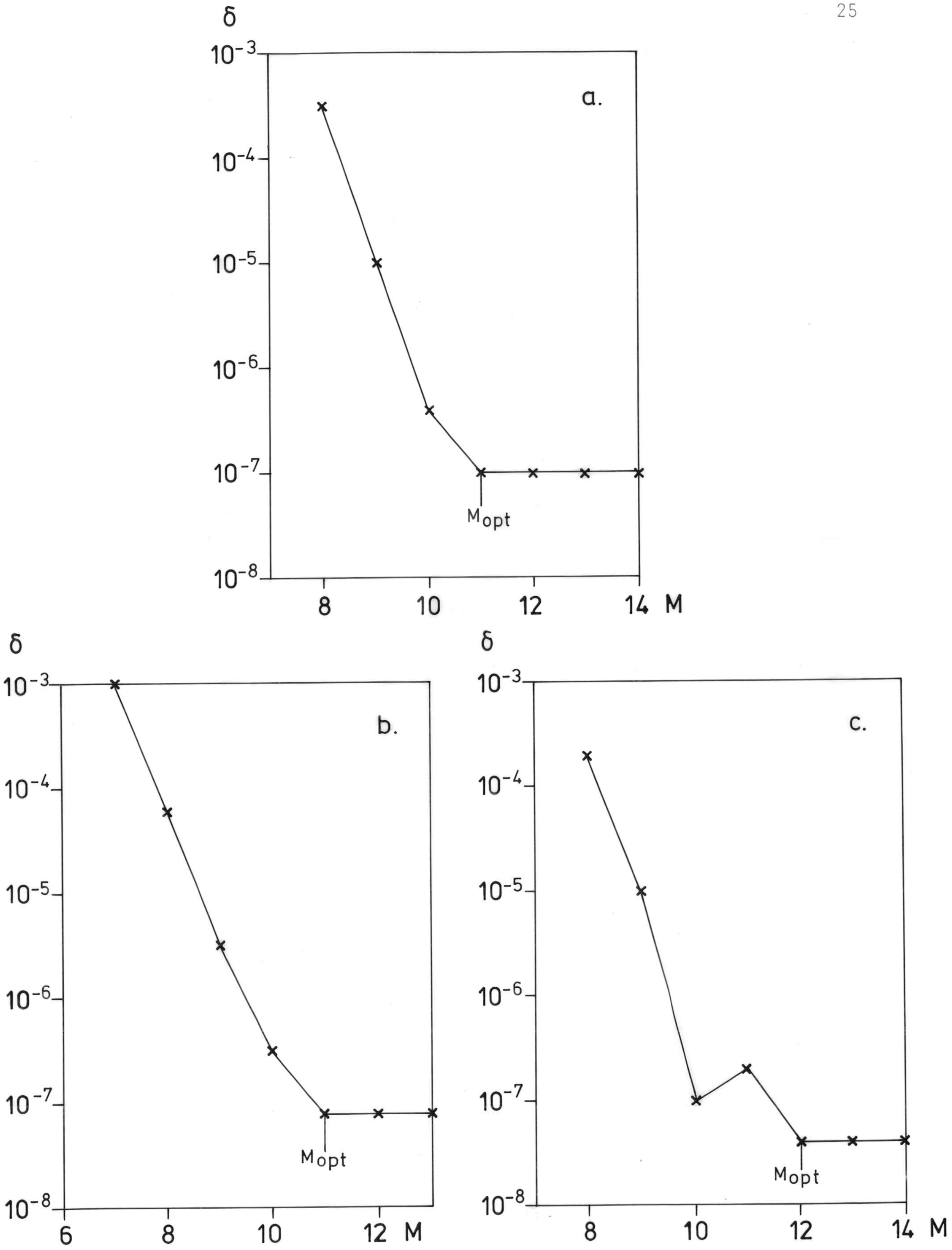


Fig. 4.1 - Minimum relative error δ as function of the number of matrix multiplications M. The δ-axis has logarithmic scale.

a. Test matrix 1 with a = -8.

b. Test matrix 1 with a = 6.4.

c. Test matrix 2 with a = 8.

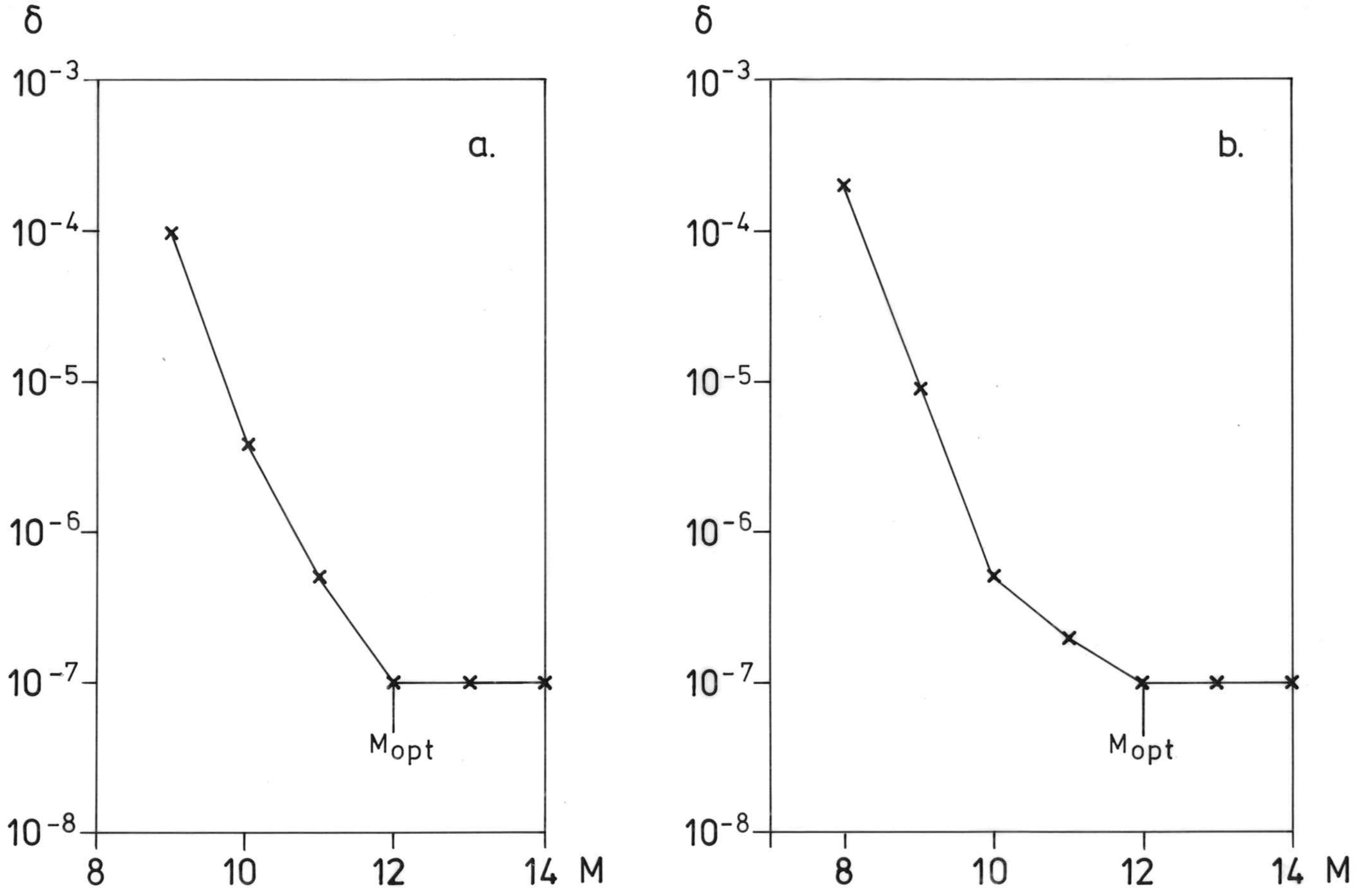


Fig. 4.2 - Minimum relative error δ as function of the number of matrix multiplications M. The δ-axis has logarithmic scale.

a. Test matrix 3 with n = 12.

b. Test matrix 4 with n = 8.

Testmatrix no.	M_{opt}	N_{opt}	k_{opt}	$\delta(\widehat{e^{A}})\cdot 10^{8}$
1: a = -8	11	8	4	10
1: a = 6.4	11	9	3	8
2: a = 8	12	9	4	4
3: n = 12	12	9	4	10
4: n = 8	12	8	5	10

Table 4.6
The optimal combination of N and k, and the obtained relative error for some test matrices.

We notice that the optimal choice of N_{opt} is either 8 or 9.

The considerations outlined above have been performed for a large number of test matrices with order between 2 and 13. The conclusion is that N_{opt} mainly takes the values 7, 8 or 9, with preference to 8.

The remaining problem is now how to determine k so that the truncation error matrix

$$T = \sum_{i=N+1}^{\infty} (A/2^{k})^{i}/i! \tag{4.17}$$

has suitable small elements.

We estimate T with

$$\hat{T} = (A/2^{k})^{N}/N! \tag{4.18}$$

Now choose k in such a way that

$$||(A/2^{k})^{N}/N!|| \leqslant \varepsilon \tag{4.19}$$

$$||(A/2^{k-1})^{N}/N!|| > \varepsilon$$

where the norm is given by (4.2) and ε is a small number dependent on the accuracy of the computer.

To avoid the computation of A^{N} in (4.18) it seems attractive to use the relation

$$||\hat{T}|| = ||(A/2^k)^N/N!|| \leqslant ||A||^N \frac{1}{2^{Nk}N!} \qquad (4.20)$$

and then choose such a k that

$$\begin{aligned} &||A||^N \frac{1}{2^{Nk}N!} \leqslant \varepsilon \\ &||A||^N \frac{1}{2^{N(k-1)}N!} > \varepsilon \end{aligned} \qquad (4.21)$$

There are, however, such matrices that (4.20) estimate $||\hat{T}||$ very badly, which implies that the value of k determined according to (4.21) is too large.

To obtain an accurate result it is necessary to compute A^N and use (4.19) to determine k. The number of required matrix multiplications to compute A^N is of course dependent on N:

N = 7: 4 matrix multiplications
N = 8: 3 matrix multiplications (4.22)
N = 9: 4 matrix multiplications

We now conclude that the most reasonable choice of N when implementing algorithm 1 on a computer with the same accuracy as PDP 15/35 is 8.

Going back to the Tables 4.1 - 4.5 it is now possible to determine the optimal k for each matrix when N = 8. Then by computing

$$||\hat{T}|| = ||(A/2^k)^8/8!|| \qquad (4.23)$$

for $k = k_{opt} - 1$ and $k = k_{opt}$ we can make a consideration how to choose ε in (4.19). See Table 4.7.

Testmatrix no.	k_{opt} when N=8	$\|\|(A/2^k)^8/8!\|\|$	
		k=k_{opt}-1	k=k_{opt}
1: a=-8	4	$2\cdot10^{-5}$	$1\cdot10^{-7}$
1: a=6.4	4	$4\cdot10^{-6}$	$2\cdot10^{-8}$
	5	$2\cdot10^{-8}$	$6\cdot10^{-11}$
2: a=8	4	$2\cdot10^{-5}$	$1\cdot10^{-7}$
3: n=12	5	$5\cdot10^{-6}$	$2\cdot10^{-8}$
	6	$2\cdot10^{-8}$	$8\cdot10^{-11}$
4: n=8	5	$3\cdot10^{-7}$	$1\cdot10^{-9}$

Table 4.7

The estimated truncation error when k = k_{opt}-1 and k=k_{opt} for some test matrices when N = 8.

If we choose $\varepsilon = 2\cdot10^{-7}$, the value of k obtained by (4.19) with N=8 will be the same as k_{opt} for the 5 matrices in Table 4.7. When considering the entire set of testmatrices the optimal value of ε is modified to $5\cdot10^{-7}$.

Point 2 of algorithm 1 in section 2.4 now can be described as:

2.a Compute $v = ||S^N||/N!$

b. Choose k in such a way that $\frac{v}{2^{Nk}} \leq \varepsilon$ and $\frac{v}{2^{N(k-1)}} > \varepsilon$

When implementing algorithm 1 on a computer with the same accuracy as PDP 15/35 a suitable choice of N is 8 and a suitable choice of ε is $5\cdot10^{-7}$.

4.2. Algorithm 2

Define the relative error of $\hat{\Psi}(A)$ as

$$\delta(\hat{\Psi}(A)) = \frac{||\hat{\Psi}(A) - \Psi(A)||}{||\Psi(A)||} \tag{4.24}$$

where the norm is given by (4.2).

The relative error of $\hat{\Psi}(A)$ as a function of N and k has been computed on PDP 15/35 for two kinds of test matrices.

Testmatrix 1:

$A = a$

where a is a scalar.

From (3.8) we get

$$\Psi(A) = \frac{e^{a} - 1}{a} \tag{4.25}$$

Testmatrix 2:

$$A = \begin{bmatrix} 0 & a \\ -a & 0 \end{bmatrix}$$

From (3.8) and (4.3) we get

$$\Psi(A) = \begin{bmatrix} \frac{1}{a} \sin a & \frac{1}{a}(1 - \cos a) \\ -\frac{1}{a}(1 - \cos a) & \frac{1}{a} \sin a \end{bmatrix} \tag{4.26}$$

Some results of the numerical investigations are shown in Tables 4.8 - - 4.10. As in section 4.1 we estimate roughly the computational effort by the required number of matrix multiplications M. From (3.17) we conclude that the number of matrix multiplications due to the scaling is 2k. We then get

$$M = N + 2k - 1 \tag{4.27}$$

For each M = 1,2,3,.. the minimum relative error is underlined in the Tables 4.8 - 4.10.

The minimum relative error δ as a function of M for the three matrices in Tables 4.8 - 4.10 is shown in Fig. 4.3. The optimal number of matrix products, M_{opt}, is easily found for each matrix.

```
MATRIX A
    6.4





RELATIVE ERROR OF  PSI (A)

*
 * N     2         3         4         5         6         7         8         9         10        11        12        13        14
K *****************************************************************************************************************************************
  *
0 *  0.9E+00  0.8E+00  0.6E+00  0.5E+00  0.3E+00  0.2E+00  0.1E+00  0.6E-01  0.3E-01  0.1E-01  0.6E-02  0.3E-02  0.1E-02
1 *  0.6E+00  0.4E+00  0.2E+00  0.9E-01  0.3E-01  0.1E-01  0.4E-02  0.1E-02  0.3E-03  0.6E-04  0.1E-04  0.3E-05  0.6E-06
2 *  0.3E+00  0.9E-01  0.2E-01  0.5E-02  0.1E-02  0.2E-03  0.3E-04  0.4E-05  0.5E-06  0.2E-07  0.2E-07  0.2E-07  0.2E-07
3 *  0.7E-01  0.1E-01  0.1E-02  0.2E-03  0.2E-04  0.2E-05  0.1E-06  0.2E-07  0.2E-07  0.2E-07  0.2E-07  0.2E-07  0.2E-07
4 *  0.1E-01  0.1E-02  0.6E-04  0.4E-05  0.3E-06  0.2E-07  0.2E-07  0.2E-07  0.2E-07  0.2E-07  0.2E-07  0.2E-07  0.2E-07
5 *  0.2E-02  0.7E-04  0.3E-05  0.2E-07  0.2E-07  0.2E-07  0.2E-07  0.2E-07  0.2E-07  0.2E-07  0.2E-07  0.2E-07  0.2E-07
6 *  0.2E-03  0.5E-05  0.3E-06  0.3E-06  0.3E-06  0.3E-06  0.3E-06  0.3E-06  0.3E-06  0.3E-06  0.3E-06  0.3E-06  0.3E-06
7 *  0.3E-04  0.7E-06  0.4E-06  0.4E-06  0.4E-06  0.4E-06  0.4E-06  0.4E-06  0.4E-06  0.4E-06  0.4E-06  0.4E-06  0.4E-06
8 *  0.4E-05  0.4E-06  0.4E-06  0.4E-06  0.4E-06  0.4E-06  0.4E-06  0.4E-06  0.4E-06  0.4E-06  0.4E-06  0.4E-06  0.4E-06
9 *  0.1E-05  0.7E-06  0.7E-06  0.7E-06  0.7E-06  0.7E-06  0.7E-06  0.7E-06  0.7E-06  0.7E-06  0.7E-06  0.7E-06  0.7E-06
```

Table 4.8

Testmatrix 1 with a = 6.4

MATRIX A
-6.4

```
RELATIVE ERROR OF  PSI  (A)

 *
  * N      2         3         4         5         6         7         8         9        10        11        12        13        14
K  ***************************************************************************************************************************************
   *
0  *   0.3E+02  0.4E+02  0.5E+02  0.5E+02  0.4E+02  0.3E+02  0.2E+02  0.1E+02  0.7E+01  0.3E+01  0.2E+01  0.7E+00  0.3E+00
1  *   0.6E+01  0.3E+01  0.9E+00  0.3E+00  0.2E-01  0.1E-01  0.1E-02  0.6E-03  0.2E-03  0.4E-04  0.9E-05  0.2E-05  0.4E-06
2  *   0.2E-02  0.4E-02  0.5E-03  0.2E-03  0.3E-04  0.5E-05  0.9E-06  0.1E-06  0.5E-07  0.2E-07  0.2E-07  0.2E-07  0.2E-07
3  *   0.4E-03  0.7E-04  0.1E-04  0.1E-05  0.1E-06  0.2E-07  0.2E-07  0.2E-07  0.2E-07  0.2E-07  0.2E-07  0.2E-07  0.2E-07
4  *   0.4E-04  0.3E-05  0.2E-06  0.2E-07  0.2E-07  0.2E-07  0.2E-07  0.2E-07  0.2E-07  0.2E-07  0.2E-07  0.2E-07  0.2E-07
5  *   0.4E-05  0.1E-06  0.5E-07  0.2E-07  0.2E-07  0.2E-07  0.2E-07  0.2E-07  0.2E-07  0.2E-07  0.2E-07  0.2E-07  0.2E-07
6  *   0.5E-06  0.2E-07  0.2E-07  0.2E-07  0.2E-07  0.2E-07  0.2E-07  0.2E-07  0.2E-07  0.2E-07  0.2E-07  0.2E-07  0.2E-07
7  *   0.7E-07  0.2E-07  0.2E-07  0.2E-07  0.2E-07  0.2E-07  0.2E-07  0.2E-07  0.2E-07  0.2E-07  0.2E-07  0.2E-07  0.2E-07
8  *   0.5E-07  0.2E-07  0.2E-07  0.2E-07  0.2E-07  0.2E-07  0.2E-07  0.2E-07  0.2E-07  0.2E-07  0.2E-07  0.2E-07  0.2E-07
9  *   0.2E-07  0.2E-07  0.2E-07  0.2E-07  0.2E-07  0.2E-07  0.2E-07  0.2E-07  0.2E-07  0.2E-07  0.2E-07  0.2E-07  0.2E-07
```

Table 4.9

Testmatrix 1 with a = - 6.4

```
MATRIX A
   0.0        6.4
  -6.4        0.0
```

```
RELATIVE ERROR OF  PSI  (A)

*
 * N     2       3       4       5       6       7       8       9       10      11      12      13      14
K  *****************************************************************************************************************
   *
0  *  0.5E+03 0.7E+03 0.8E+03 0.8E+03 0.7E+03 0.5E+03 0.3E+03 0.2E+03 0.1E+03 0.5E+02 0.2E+02 0.9E+01 0.4E+01
1  *  0.2E+03 0.6E+02 0.1E+02 0.1E+02 0.6E+01 0.2E+01 0.6E+00 0.2E+00 0.5E-01 0.1E-01 0.3E-02 0.6E-03 0.1E-03
2  *  0.1E+02 0.3E+01 0.9E+00 0.2E+00 0.4E-01 0.7E-02 0.1E-02 0.2E-03 0.2E-04 0.3E-05 0.2E-06 0.2E-06 0.2E-06
3  *  0.1E+01 0.2E+00 0.3E-01 0.4E-02 0.4E-03 0.3E-04 0.2E-05 0.2E-06 0.9E-06 0.9E-06 0.9E-06 0.9E-06 0.9E-06
4  *  0.2E+00 0.1E-01 0.9E-03 0.5E-04 0.3E-05 0.9E-06 0.9E-06 0.9E-06 0.9E-06 0.9E-06 0.9E-06 0.9E-06 0.9E-06
5  *  0.2E-01 0.7E-03 0.2E-04 0.9E-06 0.9E-06 0.9E-06 0.9E-06 0.9E-06 0.9E-06 0.9E-06 0.9E-06 0.9E-06 0.9E-06
6  *  0.2E-02 0.4E-04 0.1E-05 0.9E-06 0.9E-06 0.9E-06 0.9E-06 0.9E-06 0.9E-06 0.9E-06 0.9E-06 0.9E-06 0.9E-06
7  *  0.3E-03 0.4E-05 0.2E-05 0.2E-05 0.2E-05 0.2E-05 0.2E-05 0.2E-05 0.2E-05 0.2E-05 0.2E-05 0.2E-05 0.2E-05
8  *  0.4E-04 0.2E-05 0.2E-05 0.2E-05 0.2E-05 0.2E-05 0.2E-05 0.2E-05 0.2E-05 0.2E-05 0.2E-05 0.2E-05 0.2E-05
9  *  0.7E-05 0.3E-05 0.3E-05 0.3E-05 0.3E-05 0.3E-05 0.3E-05 0.3E-05 0.3E-05 0.3E-05 0.3E-05 0.3E-05 0.3E-05
```

Table 4.10

Testmatrix 2 with a = 6.4

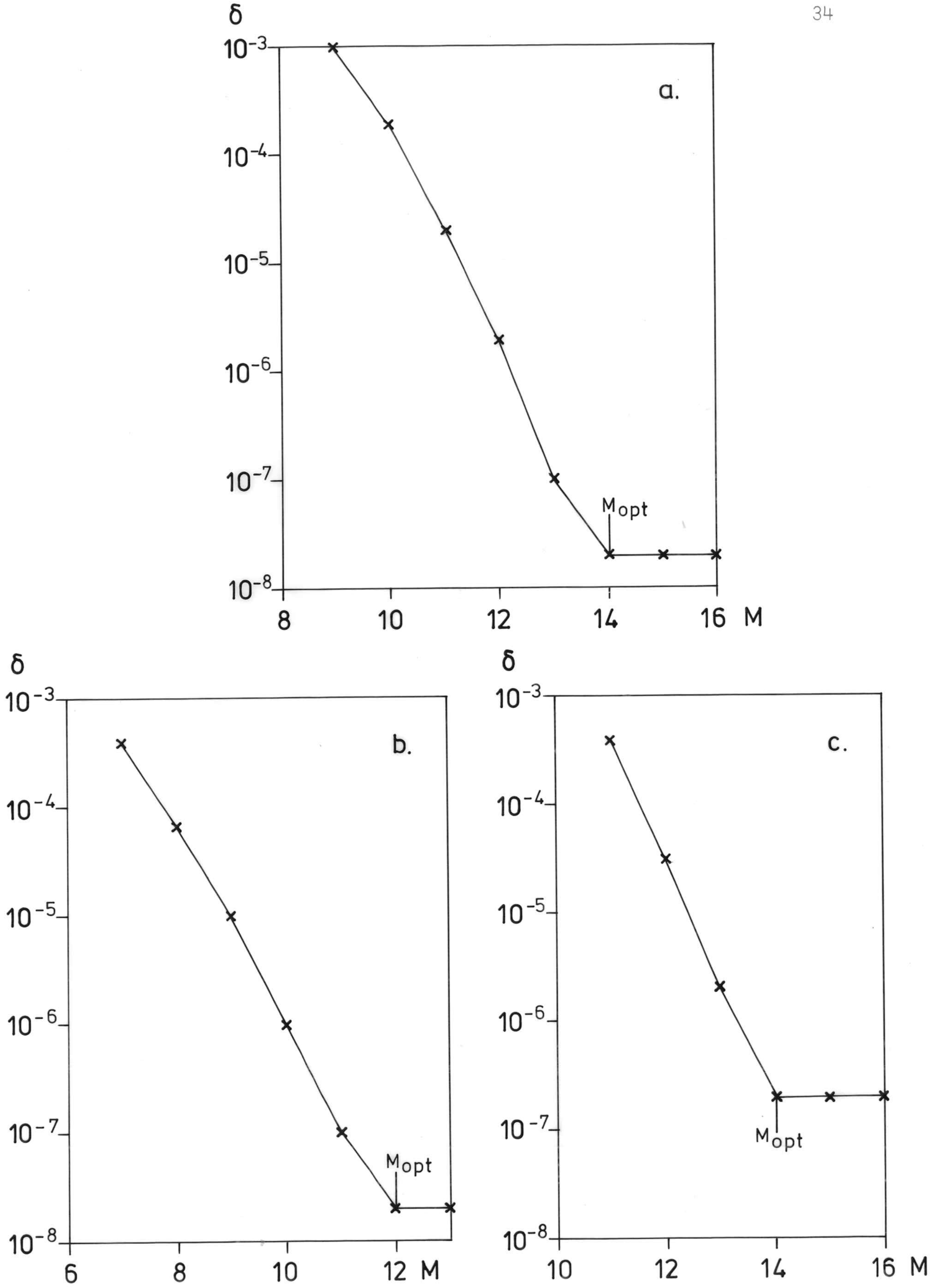


Fig. 4.3 - Minimum relative error δ as function of the number of matrix multiplications M. The δ-axis has logarithmic scale.

a. Test matrix 1 with a = 6.4

b. Test matrix 1 with a = -6.4

c. Test matrix 2 with a = 6.4

If we compare Fig. 4.3(a) to Fig. 4.1(b) we can notice that $M_{opt}(\Psi(6.4)) = 14$ and $M_{opt}(e^{6.4}) = 11$. It is often true that the optimal number of matrix multiplications when evaluating $\Psi(A)$ is greater than the number when evaluating e^{A}.

When M_{opt} has been determined from the plot we go back to the table to find the optimal combination (or combinations) of N and k. The results are summarized in Table 4.11.

Testmatrix no.	M_{opt}	N_{opt}	k_{opt}	$\delta(\hat{\Psi}(A))\cdot 10^{8}$
1: a=6.4	14	5	5	2
		7	4	
		9	3	
		11	2	
1: a= -6.4	12	5	4	2
		7	3	
2: a= 6.4	14	9	3	20

Table 4.11
The optimal combination or combinations of N and k, and the obtained relative error for some test matrices.

When all test matrices are considered it appears that the most common values of N_{opt} are 7,8 and 9.

As in section 4.1 we estimate the truncation error matrix

$$T = \sum_{i=N+1}^{\infty} (A/2^{k})^{i}/(i+1)! \qquad (4.28)$$

with

$$\hat{T} = (A/2^{k})^{N}/(N+1)! \qquad (4.29)$$

We now choose k in such a way that

$$\begin{aligned} &\|(A/2^{k})^{N}/(N+1)!\| \leq \varepsilon \\ &\|(A/2^{k-1})^{N}/(N+1)!\| > \varepsilon \end{aligned} \qquad (4.30)$$

where the norm is given by (4.2) and ε is a small number dependent on the accuracy of the computer. The number of matrix multiplications to compute A^N is given by (4.22). Our conclusion now is that the most reasonable choice of N when implementing algorithm 2 on a computer with the same accuracy as PDP 15/35 is 8.

If we make the same considerations as in section 4.1, we obtain that $\varepsilon = 5 \cdot 10^{-7}$ is a suitable choice on PDP 15/35.

Point 1 of algorithm 2 in section 3.3 now can be described as:

1.a. Compute $v = ||(A\tau)^N||/(N+1)!$
   b. Choose k in such a way that $\frac{v}{2^{Nk}} \leqslant \varepsilon$ and $\frac{v}{2^{N(k-1)}} > \varepsilon$

When implementing algorithm 2 on a computer with the same accuracy as PDP 15/35 a suitable choice of N is 8 and a suitable choice of ε is $5 \cdot 10^{-7}$.

5. ORGANIZATION OF THE CODES FOR THE TWO ALGORITHMS

The two algorithms given in section 2.4 and 3.3 have been implemented on the computer PDP 15/35. The programming language is FORTRAN IV and the compiler F4S V12D has been used. The computer uses software real arithmetic.

Both in algorithm 1 and 2 a series has to be evaluated. The difference between the two series expansions is quite small, and it seems reasonable to make a subroutine which can be called by both algorithm 1 and 2 to perform the series evaluations. The program head of this Fortran subroutine EXPAN is shown in Appendix B. The computations are organized as described in the second method of section 2.2 and in section 3.2. The subroutine needs one internal matrix, which is stored in a common block SLASK of fixed length 1 k used for dummy matrices.

This internal matrix could in fact be saved by making the code and computing time longer. Cf section 2.2. The maximum order of the input matrix of subroutine EXPAN is restricted to 13 by the internal matrix. The required number of cells of the code and the internal matrix is shown in Table 5.1. Note that a real variable on PDP 15/35 is stored in two cells.

Algorithm 1 is implemented by subroutine MEXP. The program head is shown in Appendix C. Subroutine MEXP calls subroutine EXPAN and the two subroutines NORM and MMULT in the Program Library [19]. Subroutine NORM computes the norm given by (4.2) and subroutine MMULT performs matrix multiplications. See Fig 5.1.

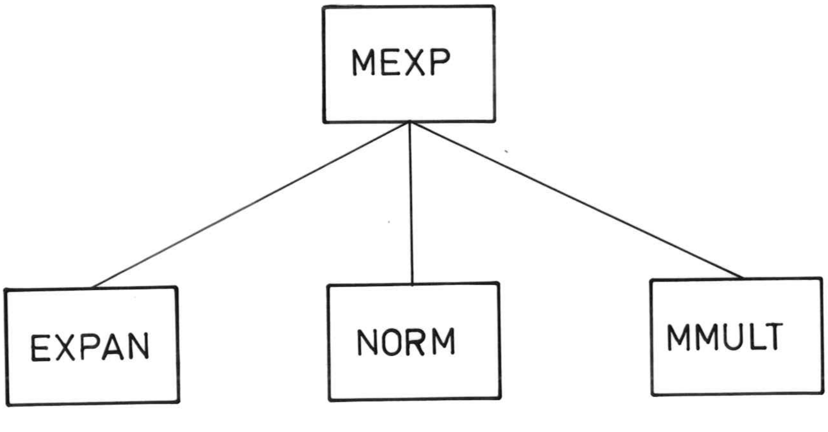


Fig. 5.1. The subroutine structure of MEXP.

The computations of MEXP are organized as described in section 2.4 and section 4.1. If the origin shift (see section 2.3) is to be performed or not is controlled by the argument NOTRAC. Subroutine MEXP needs two internal matrices, which are stored in the common block SLASK. Note that one of these matrices is shared with EXPAN. The required number of cells of MEXP is shown in Table 5.1.

Finally algorithm 2 is implemented by subroutine COSA. The program head is given in Appendix D. The computations are organized as described in section 3.3 and 4.2. The subroutine calls EXPAN, NORM and MMULT as shown in Fig. 5.2. COSA needs three internal matrices, stored in the common block SLASK. One of these is shared with subroutine EXPAN. The required number of cells is shown in Table 5.1.

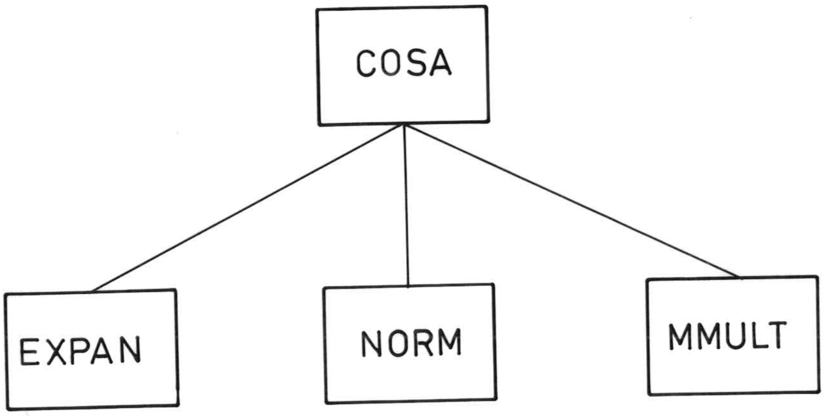


Fig 5.2

The subroutine structure of COSA.

	Number of cells			
	Code without subroutines	Code with sub- subroutines	Internal matrices	Totally
EXPAN	223	1 256	338	1 594
MEXP	463	2 251	676	2 927
COSA	498	2 197	1 014	3 211

Table 5.1
The required number of cells of subroutines EXPAN, MEXP and COSA on PDP 15/35. The maximum order of the input matrices is restricted to 13 by internal matrices stored in the common block SLASK. Note that a real variable is stored in two cells. The notation "subroutines" in the table head means for EXPAN system subroutines, mainly consisting of the real arithmetic package, and for MEXP and COSA furthermore subroutines EXPAN, NORM and MMULT.

It should be pointed out that subroutines MEXP and COSA in fact compute A^8 twice, the first time to determine the scale factor $1/2^k$ and the second time when evaluating the series. If the main memory is large enough to store the seven matrices A^2, $A^3, \dots,$ A^8 it seems possible to decrease the computing time by only computing A^8 once. The saving in computing time is roughly the three matrix multiplications $A*A$, A^2*A^2 and A^4*A^4 when determining the scale factor $1/2^k$, but probably the expanded programming administration will decrease the saving. In our case, however, there is no possibility to store seven matrices.

6. NUMERICAL EXAMPLES

The two Fortran subroutines MEXP and COSA, described in chapter 5, have been tested on PDP 15/35. Some of these test examples will be given in this chapter.

The execution time is measured with the accuracy of 0.1 sec..When a time is equal to 0.0 sec, this means that the real execution time is less than 0.1 sec.. Some of the characteristic times of PDP 15/35 are listed in Appendix A.

The relative error is computed from the internal representation of the numbers in the computer. The accuracy of a real floating point number is between 7 and 8 digits. The resulting matrices, however, are printed with 7 digits, which means that it is possible for the elements of a computed matrix exp(A) to have 7 correct digits but the relative error is not zero.

It is possible to compute exp(A) in three ways:

o MEXP without origin shift (NOTRAC = 0)
o MEXP with origin shift (NOTRAC = 1)
o COSA

The total number of matrix multiplications M is obtained from the number of scalings k by

$$M = k + 7 \tag{6.1}$$

for MEXP and

$$M = 2k + 8 \tag{6.2}$$

for COSA. Note that the number of matrix multiplications to compute $\Psi(A)$ is 2k+7, but another matrix multiplication is required to obtain exp(A). COSA also performs a matrix-vector-multiplication to obtain, in this case, a dummy vector .

The first example is test matrix 1 (see sec. 4.1) with a = 3. See Table 6.1. In this case, the call of MEXP with NOTRAC = 1 only is a way to let the Fortran Library Function EXP perform the computation.

```
MATRIX A

  0.3000000E+01

NORM(A)=    0.3000000E+01


EXACT MATRIX EXP(A)

  0.2008554E+02

NORM(EXP(A))=    0.2008554E+02




EXP(A) COMPUTED BY MEXP
NOTRAC=0

  0.2008554E+02


NUMBER OF SCALINGS (K) =   3
RELATIVE ERROR =  0.0E+00
EXECUTION TIME =     0.0 SEC




EXP(A) COMPUTED BY MEXP
NOTRAC=1

  0.2008554E+02


NUMBER OF SCALINGS (K) =   0
RELATIVE ERROR =  0.7E-07
EXECUTION TIME =     0.0 SEC




EXP(A) COMPUTED BY COSA

  0.2008554E+02


NUMBER OF SCALINGS (K) =   2
RELATIVE ERROR =  0.7E-07
EXECUTION TIME =     0.0 SEC
```

Table 6.1.

Test matrix 1 with a = 3.

The next example is the matrix

$$A = \begin{bmatrix} 100 & 0 \\ 0 & 1 \end{bmatrix} \tag{6.3}$$

with

$$e^A = \begin{bmatrix} e^{100} & 0 \\ 0 & e \end{bmatrix} \tag{6.4}$$

The result is shown in Table 6.2.

Lion and Alderson [15] have suggested following matrix:

$$A = \begin{bmatrix} -2 & 4 \\ 3 & -6 \end{bmatrix} \tau \tag{6.5}$$

with

$$e^{A\tau} = \begin{bmatrix} \frac{1}{4}(3 + e^{-8\tau}) & \frac{1}{2}(1 - e^{-8\tau}) \\ \frac{3}{8}(1 - e^{-8\tau}) & \frac{1}{4}(1 + 3e^{-8\tau}) \end{bmatrix} \tag{6.6}$$

Matrix A has the eigenvalues 0 and -8τ. The sampling interval τ is chosen as 1/80, 1/8, 1, 10, 100 and 1000. The results are shown in Tables 6.3 - - 6.8. A summary of the results is given in Table 6.9.

```
MATRIX A

  0.1000000E+03    0.0000000E+00
  0.0000000E+00    0.1000000E+01

NORM(A)=    0.1000000E+03
```

```
EXACT MATRIX EXP(A)

  0.2688117E+44    0.0000000E+00
  0.0000000E+00    0.2718282E+01

NORM(EXP(A))=    0.2688117E+44
```

```
EXP(A) COMPUTED BY MEXP
NOTRAC=0

  0.2688113E+44    0.0000000E+00
  0.0000000E+00    0.2718273E+01


NUMBER OF SCALINGS (K) =  8
EXECUTION TIME =    0.1 SEC


EXP(A) COMPUTED BY MEXP
NOTRAC=1

  0.2688113E+44    0.0000000E+00
  0.0000000E+00    0.2718287E+01


NUMBER OF SCALINGS (K) =  7
EXECUTION TIME =    0.1 SEC


EXP(A) COMPUTED BY COSA

  0.2688112E+44    0.0000000E+00
  0.0000000E+00    0.2718282E+01


NUMBER OF SCALINGS (K) =  7
EXECUTION TIME =    0.1 SEC
```

Table 6.2.

```
MATRIX A

 -0.2500000E-01    0.5000000E-01
  0.3750000E-01   -0.7500000E-01

NORM(A)=    0.1125000E+00


EXACT MATRIX EXP(A)

  0.9762094E+00    0.4758129E-01
  0.3568597E-01    0.9286281E+00

NORM(EXP(A))=    0.1011895E+01



EXP(A) COMPUTED BY MEXP
NOTRAC=0

  0.9762094E+00    0.4758129E-01
  0.3568597E-01    0.9286281E+00


NUMBER OF SCALINGS (K) =   0
RELATIVE ERROR =  0.1E-07
EXECUTION TIME =     0.0 SEC



EXP(A) COMPUTED BY MEXP
NOTRAC=1

  0.9762094E+00    0.4758129E-01
  0.3568597E-01    0.9286281E+00


NUMBER OF SCALINGS (K) =   0
RELATIVE ERROR =  0.2E-07
EXECUTION TIME =     0.0 SEC



EXP(A) COMPUTED BY COSA

  0.9762094E+00    0.4758129E-01
  0.3568597E-01    0.9286281E+00


NUMBER OF SCALINGS (K) =   0
RELATIVE ERROR =  0.1E-07
EXECUTION TIME =     0.0 SEC
```

Table 6.3.

$$A = \begin{bmatrix} -2 & 4 \\ 3 & -6 \end{bmatrix}\tau, \text{ where } \tau = \frac{1}{80}$$

```
MATRIX A

 -0.2500000E+00    0.5000000E+00
  0.3750000E+00   -0.7500000E+00

NORM(A)=     0.1125000E+01


EXACT MATRIX EXP(A)

   0.8419699E+00    0.3160603E+00
   0.2370452E+00    0.5259096E+00

NORM(EXP(A))=     0.1079015E+01


EXP(A) COMPUTED BY MEXP
NOTRAC=0

   0.8419698E+00    0.3160603E+00
   0.2370452E+00    0.5259096E+00


NUMBER OF SCALINGS (K) =   1
RELATIVE ERROR =  0.3E-08
EXECUTION TIME =     0.1 SEC


EXP(A) COMPUTED BY MEXP
NOTRAC=1

   0.8419699E+00    0.3160603E+00
   0.2370452E+00    0.5259096E+00


NUMBER OF SCALINGS (K) =   0
RELATIVE ERROR =  0.5E-07
EXECUTION TIME =     0.0 SEC


EXP(A) COMPUTED BY COSA

   0.8419699E+00    0.3160603E+00
   0.2370452E+00    0.5259096E+00


NUMBER OF SCALINGS (K) =   1
RELATIVE ERROR =  0.2E-07
EXECUTION TIME =     0.1 SEC
```

Table 6.4.

$$A = \begin{bmatrix} -2 & 4 \\ 3 & -6 \end{bmatrix} \tau, \text{ where } \tau = \frac{1}{8}$$

```
MATRIX A

 -0.2000000E+01    0.4000000E+01
  0.3000000E+01   -0.6000000E+01

NORM(A)=    0.9000000E+01


EXACT MATRIX EXP(A)

  0.7500839E+00    0.4998323E+00
  0.3748742E+00    0.2502516E+00

NORM(EXP(A))=    0.1124958E+01



EXP(A) COMPUTED BY MEXP
NOTRAC=0

  0.7500837E+00    0.4998322E+00
  0.3748741E+00    0.2502515E+00


NUMBER OF SCALINGS (K) =  4
RELATIVE ERROR =  0.2E-06
EXECUTION TIME =     0.1 SEC



EXP(A) COMPUTED BY MEXP
NOTRAC=1

  0.7500837E+00    0.4998322E+00
  0.3748741E+00    0.2502516E+00


NUMBER OF SCALINGS (K) =  3
RELATIVE ERROR =  0.2E-06
EXECUTION TIME =     0.1 SEC



EXP(A) COMPUTED BY COSA

  0.7500839E+00    0.4998322E+00
  0.3748742E+00    0.2502516E+00


NUMBER OF SCALINGS (K) =  4
RELATIVE ERROR =  0.5E-07
EXECUTION TIME =     0.1 SEC
```

Table 6.5

$$A = \begin{bmatrix} -2 & 4 \\ 3 & -6 \end{bmatrix} \tau \text{ , where } \tau = 1$$

```
MATRIX A

 -0.2000000E+02    0.4000000E+02
  0.3000000E+02   -0.6000000E+02

NORM(A)=    0.9000000E+02


EXACT MATRIX EXP(A)

   0.7500000E+00    0.5000000E+00
   0.3750000E+00    0.2500000E+00

NORM(EXP(A))=    0.1125000E+01



EXP(A) COMPUTED BY MEXP
NOTRAC=0

  0.7499989E+00    0.4999992E+00
  0.3749994E+00    0.2499996E+00


NUMBER OF SCALINGS (K) =   8
RELATIVE ERROR =  0.1E-05
EXECUTION TIME =     0.1 SEC



EXP(A) COMPUTED BY MEXP
NOTRAC=1

  0.7499993E+00    0.4999996E+00
  0.3749997E+00    0.2499998E+00


NUMBER OF SCALINGS (K) =   7
RELATIVE ERROR =  0.9E-06
EXECUTION TIME =     0.1 SEC



EXP(A) COMPUTED BY COSA
  0.7500002E+00    0.4999995E+00
  0.3750001E+00    0.2499998E+00


NUMBER OF SCALINGS (K) =   7
RELATIVE ERROR =  0.6E-06
EXECUTION TIME =     0.1 SEC
```

Table 6.6

$$A = \begin{bmatrix} -2 & 4 \\ 3 & -6 \end{bmatrix} \tau \text{ , where } \tau = 10$$

```
MATRIX A

 -0.2000000E+03    0.4000000E+03
  0.3000000E+03   -0.6000000E+03

NORM(A)=    0.9000000E+03


EXACT MATRIX EXP(A)

  0.7500000E+00    0.5000000E+00
  0.3750000E+00    0.2500000E+00

NORM(EXP(A))=    0.1125000E+01


EXP(A) COMPUTED BY MEXP
NOTRAC=0

  0.7499847E+00    0.4999898E+00
  0.3749924E+00    0.2499949E+00


NUMBER OF SCALINGS (K) = 11
RELATIVE ERROR =  0.2E-04
EXECUTION TIME =     0.1 SEC


EXP(A) COMPUTED BY MEXP
NOTRAC=1

  0.7499847E+00    0.4999898E+00
  0.3749924E+00    0.2499949E+00


NUMBER OF SCALINGS (K) = 11
RELATIVE ERROR =  0.2E-04
EXECUTION TIME =     0.1 SEC


EXP(A) COMPUTED BY COSA

  0.7499995E+00    0.5000010E+00
  0.3750004E+00    0.2499992E+00


NUMBER OF SCALINGS (K) = 10
RELATIVE ERROR =  0.1E-05
EXECUTION TIME =     0.2 SEC
```

Table 6.7

$$A = \begin{bmatrix} -2 & 4 \\ 3 & -6 \end{bmatrix} \tau, \text{ where } \tau = 100$$

```
MATRIX A

 -0.2000000E+04    0.4000000E+04
  0.3000000E+04   -0.6000000E+04

NORM(A)=     0.9000000E+04


EXACT MATRIX EXP(A)

  0.7500000E+00    0.5000000E+00
  0.3750000E+00    0.2500000E+00

NORM(EXP(A))=    0.1125000E+01


EXP(A) COMPUTED BY MEXP
NOTRAC=0

  0.7499050E+00    0.4999367E+00
  0.3749525E+00    0.2499683E+00


NUMBER OF SCALINGS (K) = 14
RELATIVE ERROR =  0.1E-03
EXECUTION TIME =     0.1 SEC


EXP(A) COMPUTED BY MEXP
NOTRAC=1

  0.7499050E+00    0.4999367E+00
  0.3749525E+00    0.2499683E+00


NUMBER OF SCALINGS (K) = 14
RELATIVE ERROR =  0.1E-03
EXECUTION TIME =     0.1 SEC


EXP(A) COMPUTED BY COSA

  0.7500181E+00    0.4999638E+00
  0.3749951E+00    0.2500098E+00


NUMBER OF SCALINGS (K) = 14
RELATIVE ERROR =  0.4E-04
EXECUTION TIME =     0.2 SEC
```

Table 6.8

$$A = \begin{bmatrix} -2 & 4 \\ 3 & -6 \end{bmatrix}\tau, \text{ where } \tau = 1000$$

	MEXP,NOTRAC=0				MEXP,NOTRAC=1				COSA			
Sampling interval τ	k	M	Exec. time sec	$\delta \cdot 10^8$	k	M	Exec. time sec	$\delta \cdot 10^8$	k	M	Exec. time sec	$\delta \cdot 10^8$
$\frac{1}{80}$	0	7	<0.1	1	0	7	0.1	2	0	8	<0.1	1
$\frac{1}{8}$	1	8	0.1	0.3	0	7	0.1	5	1	10	0.1	2
1	4	11	0.1	20	3	10	0.1	20	4	16	0.1	5
10	8	15	0.1	100	7	14	0.1	90	7	22	0.1	60
100	11	18	0.1	2000	11	18	0.1	2000	10	28	0.2	100
1000	14	21	0.1	10000	14	21	0.1	10000	14	36	0.2	4000

Table 6.9

Number of scalings k, total number of matrix multiplications M, execution time and relative error δ when computing

$$\exp \begin{bmatrix} -2 & 4 \\ 3 & -6 \end{bmatrix} \tau$$

with MEXP (NOTRAC = 0 and NOTRAC = 1) and COSA. When $\tau = 100$ and $\tau = 1000$ there is no difference in computation in MEXP between NOTRAC = 0 and NOTRAC = 1, for tr(A)/n is greater than 170.

See Appendix C.

Two examples generated by testmatrix 3 given in section 4.1 are shown in Table 6.10 and 6.11. The order n is 5 resp. 6.

A summary of the results of testmatrix 3, when the order n is varied from 2 to 13, is shown in Table 6.12.

The results of testmatrix 4 of section 4.1, when the order n is 4 and 5, are given in Tables 6.13 and 6.14. A summary of the results when n is varied from 2 to 13 is shown in Table 6.15.

```
MATRIX A

 -0.3000000E+01   0.2000000E+01   0.2000000E+01   0.2000000E+01   0.2000000E+01
 -0.4000000E+01   0.3000000E+01   0.2000000E+01   0.2000000E+01   0.2000000E+01
 -0.3000000E+01   0.0000000E+00   0.4000000E+01   0.2000000E+01   0.2000000E+01
 -0.2000000E+01   0.0000000E+00   0.0000000E+00   0.5000000E+01   0.2000000E+01
 -0.1000000E+01   0.0000000E+00   0.0000000E+00   0.0000000E+00   0.6000000E+01

NORM(A)=    0.1300000E+02


EXACT MATRIX EXP(A)

 -0.1596482E+02  -0.1473557E+02  -0.1823930E+02   0.9722830E+01   0.1876300E+03
 -0.1868310E+02  -0.1201729E+02  -0.1823930E+02   0.9722830E+01   0.1876300E+03
 -0.1401232E+02  -0.2407712E+02  -0.1085025E+02   0.9722830E+01   0.1876300E+03
 -0.9341548E+01  -0.1605141E+02  -0.4363226E+02   0.2980837E+02   0.1876300E+03
 -0.4670774E+01  -0.8025707E+01  -0.2181613E+02  -0.5930240E+02   0.2422282E+03

NORM(EXP(A))=    0.3360432E+03



EXP(A) COMPUTED BY MEXP
NOTRAC=0

 -0.1596481E+02  -0.1473557E+02  -0.1823930E+02   0.9722833E+01   0.1876300E+03
 -0.1868309E+02  -0.1201729E+02  -0.1823930E+02   0.9722831E+01   0.1876300E+03
 -0.1401232E+02  -0.2407712E+02  -0.1085025E+02   0.9722833E+01   0.1876300E+03
 -0.9341551E+01  -0.1605141E+02  -0.4363226E+02   0.2980838E+02   0.1876300E+03
 -0.4670774E+01  -0.8025705E+01  -0.2181613E+02  -0.5930239E+02   0.2422281E+03


NUMBER OF SCALINGS (K) =   4
RELATIVE ERROR =  0.1E-06
EXECUTION TIME =     0.6 SEC
```

Table 6.10 (cont. next page)

Testmatrix 3 with n = 5

```
EXP(A) COMPUTED BY MEXP
NOTRAC=1

 -0.1596482E+02  -0.1473557E+02  -0.1823930E+02   0.9722827E+01   0.1876300E+03
 -0.1868310E+02  -0.1201729E+02  -0.1823930E+02   0.9722827E+01   0.1876300E+03
 -0.1401232E+02  -0.2407712E+02  -0.1085025E+02   0.9722827E+01   0.1876300E+03
 -0.9341548E+01  -0.1605141E+02  -0.4363226E+02   0.2980836E+02   0.1876300E+03
 -0.4670775E+01  -0.8025707E+01  -0.2181613E+02  -0.5930240E+02   0.2422282E+03

NUMBER OF SCALINGS (K) =  3
RELATIVE ERROR =  0.5E-07
EXECUTION TIME =    0.6 SEC



EXP(A) COMPUTED BY COSA

 -0.1596481E+02  -0.1473557E+02  -0.1823930E+02   0.9722831E+01   0.1876300E+03
 -0.1868310E+02  -0.1201729E+02  -0.1823930E+02   0.9722832E+01   0.1876300E+03
 -0.1401232E+02  -0.2407712E+02  -0.1085025E+02   0.9722832E+01   0.1876300E+03
 -0.9341547E+01  -0.1605141E+02  -0.4363226E+02   0.2980837E+02   0.1876300E+03
 -0.4670774E+01  -0.8025707E+01  -0.2181613E+02  -0.5930239E+02   0.2422282E+03

NUMBER OF SCALINGS (K) =  3
RELATIVE ERROR =  0.5E-07
EXECUTION TIME =    0.7 SEC
```

Table 6.10 (cont)

```
MATRIX A

 -0.4000000E+01   0.2000000E+01   0.2000000E+01   0.2000000E+01   0.2000000E+01   0.2000000E+01
 -0.5000000E+01   0.3000000E+01   0.2000000E+01   0.2000000E+01   0.2000000E+01   0.2000000E+01
 -0.4000000E+01   0.0000000E+00   0.4000000E+01   0.2000000E+01   0.2000000E+01   0.2000000E+01
 -0.3000000E+01   0.0000000E+00   0.0000000E+00   0.5000000E+01   0.2000000E+01   0.2000000E+01
 -0.2000000E+01   0.0000000E+00   0.0000000E+00   0.0000000E+00   0.6000000E+01   0.2000000E+01
 -0.1000000E+01   0.0000000E+00   0.0000000E+00   0.0000000E+00   0.0000000E+00   0.7000000E+01

NORM(A)=    0.1600000E+02


EXACT MATRIX EXP(A)

 -0.2063559E+02  -0.2276128E+02  -0.4005544E+02  -0.4957957E+02   0.2642939E+02   0.5100313E+03
 -0.2335387E+02  -0.2004300E+02  -0.4005544E+02  -0.4957957E+02   0.2642939E+02   0.5100313E+03
 -0.1868310E+02  -0.3210283E+02  -0.3266638E+02  -0.4957957E+02   0.2642939E+02   0.5100313E+03
 -0.1401232E+02  -0.2407712E+02  -0.6544840E+02  -0.2949403E+02   0.2642939E+02   0.5100313E+03
 -0.9341548E+01  -0.1605141E+02  -0.4363226E+02  -0.1186048E+03   0.8102754E+02   0.5100313E+03
 -0.4670774E+01  -0.8025707E+01  -0.2181613E+02  -0.5930240E+02  -0.1612006E+03   0.6584444E+03

NORM(EXP(A))=    0.9134601E+03


EXP(A) COMPUTED BY MEXP
NOTRAC=0

 -0.2063559E+02  -0.2276127E+02  -0.4005543E+02  -0.4957956E+02   0.2642941E+02   0.5100312E+03
 -0.2335387E+02  -0.2004299E+02  -0.4005543E+02  -0.4957956E+02   0.2642941E+02   0.5100312E+03
 -0.1868309E+02  -0.3210282E+02  -0.3266638E+02  -0.4957956E+02   0.2642941E+02   0.5100312E+03
 -0.1401232E+02  -0.2407712E+02  -0.6544839E+02  -0.2949403E+02   0.2642941E+02   0.5100312E+03
 -0.9341548E+01  -0.1605141E+02  -0.4363226E+02  -0.1186048E+03   0.8102757E+02   0.5100312E+03
 -0.4670768E+01  -0.8025703E+01  -0.2181613E+02  -0.5930238E+02  -0.1612006E+03   0.6584442E+03


NUMBER OF SCALINGS (K) =   4
RELATIVE ERROR =  0.2E-06
EXECUTION TIME =     0.9 SEC
```

Table 6.11 (cont next page)

Testmatrix 3 with n = 6

```
EXP(A) COMPUTED BY MEXP
NOTRAC=1

 -0.2063559E+02  -0.2276128E+02  -0.4005543E+02  -0.4957957E+02   0.2642939E+02   0.5100313E+03
 -0.2335387E+02  -0.2004299E+02  -0.4005543E+02  -0.4957956E+02   0.2642939E+02   0.5100313E+03
 -0.1868309E+02  -0.3210282E+02  -0.3266638E+02  -0.4957957E+02   0.2642939E+02   0.5100313E+03
 -0.1401232E+02  -0.2407712E+02  -0.6544839E+02  -0.2949403E+02   0.2642940E+02   0.5100313E+03
 -0.9341548E+01  -0.1605141E+02  -0.4363226E+02  -0.1186048E+03   0.8102754E+02   0.5100313E+03
 -0.4670775E+01  -0.8025707E+01  -0.2181613E+02  -0.5930240E+02  -0.1612006E+03   0.6584445E+03

NUMBER OF SCALINGS (K) =  3
RELATIVE ERROR =  0.5E-07
EXECUTION TIME =     0.9 SEC



EXP(A) COMPUTED BY COSA

 -0.2063559E+02  -0.2276128E+02  -0.4005543E+02  -0.4957956E+02   0.2642939E+02   0.5100312E+03
 -0.2335387E+02  -0.2004299E+02  -0.4005543E+02  -0.4957956E+02   0.2642939E+02   0.5100312E+03
 -0.1868309E+02  -0.3210282E+02  -0.3266637E+02  -0.4957956E+02   0.2642939E+02   0.5100312E+03
 -0.1401232E+02  -0.2407712E+02  -0.6544839E+02  -0.2949402E+02   0.2642940E+02   0.5100312E+03
 -0.9341547E+01  -0.1605141E+02  -0.4363226E+02  -0.1186048E+03   0.8102753E+02   0.5100312E+03
 -0.4670774E+01  -0.8025706E+01  -0.2181613E+02  -0.5930239E+02  -0.1612006E+03   0.6584444E+03

NUMBER OF SCALINGS (K) =  4
RELATIVE ERROR =  0.9E-07
EXECUTION TIME =     1.2 SEC
```

Table 6.11 (cont)

Testmatrix	MEXP,NOTRAC=0				MEXP,NOTRAC=1				COSA			
Order n	k	M	Exec time sec	$\delta\cdot 10^8$	k	M	Exec time sec	$\delta\cdot 10^8$	k	M	Exec time sec	$\delta\cdot 10^8$
2	2	9	0.1	2	0	7	0.1	6	2	12	0.1	1
3	3	10	0.2	7	2	9	0.2	9	3	14	0.2	5
4	3	10	0.3	7	2	9	0.3	3	3	1.4	0.4	3
5	4	11	0.6	10	3	10	0.6	5	3	14	0.7	5
6	4	11	0.9	20	3	10	0.9	5	4	16	1.2	9
7	4	11	1.3	8	3	10	1.3	4	4	16	1.7	8
8	4	11	1.8	30	4	11	1.9	20	4	16	2.4	10
9	5	12	2.6	20	4	11	2.5	20	4	16	3.2	10
10	5	12	3.4	20	4	11	3.2	10	4	16	4.1	10
11	5	12	4.3	60	4	11	4.1	9	4	16	5.3	20
12	5	12	5.4	50	4	11	5.1	20	5	18	7.3	10
13	5	12	6.7	60	4	11	6.3	30	5	18	9.0	30

Table 6.12

Number of scalings k, total number of matrix multiplications M, execution time and relative error δ when computing exp(A), where A is testmatrix 3, with MEXP (NOTRAC = 0 and NOTRAC = 1) and COSA.

```
MATRIX A

 -0.2000000E+00   -0.2000000E+00    0.8000000E+00    0.1800000E+01
 -0.1400000E+01    0.1600000E+01    0.6000000E+00    0.1600000E+01
 -0.1600000E+01   -0.6000000E+00    0.3400000E+01    0.1400000E+01
 -0.1800000E+01   -0.8000000E+00    0.2000000E+00    0.5200000E+01

NORM(A)=    0.8000000E+01


EXACT MATRIX EXP(A)

 -0.1206530E+02   -0.1011281E+02    0.2583676E+01    0.3709629E+02
 -0.1571773E+02   -0.3657904E+01    0.1649521E+01    0.3616213E+02
 -0.1825703E+02   -0.1358626E+02    0.1919576E+02    0.3362284E+02
 -0.2515955E+02   -0.2048878E+02   -0.7792298E+01    0.8131846E+02

NORM(EXP(A))=    0.1347591E+03

EXP(A) COMPUTED BY MEXP
NOTRAC=0

 -0.1206530E+02   -0.1011280E+02    0.2583675E+01    0.3709629E+02
 -0.1571773E+02   -0.3657904E+01    0.1649521E+01    0.3616213E+02
 -0.1825703E+02   -0.1358625E+02    0.1919576E+02    0.3362283E+02
 -0.2515955E+02   -0.2048878E+02   -0.7792297E+01    0.8131846E+02


NUMBER OF SCALINGS (K) =  3
RELATIVE ERROR =  0.4E-07
EXECUTION TIME =     0.3 SEC


EXP(A) COMPUTED BY MEXP
NOTRAC=1

 -0.1206530E+02   -0.1011281E+02    0.2583675E+01    0.3709629E+02
 -0.1571773E+02   -0.3657904E+01    0.1649521E+01    0.3616214E+02
 -0.1825703E+02   -0.1358626E+02    0.1919576E+02    0.3362284E+02
 -0.2515955E+02   -0.2048878E+02   -0.7792298E+01    0.8131847E+02


NUMBER OF SCALINGS (K) =  2
RELATIVE ERROR =  0.7E-07
EXECUTION TIME =     0.4 SEC


EXP(A) COMPUTED BY COSA

 -0.1206530E+02   -0.1011280E+02    0.2583675E+01    0.3709629E+02
 -0.1571773E+02   -0.3657904E+01    0.1649521E+01    0.3616213E+02
 -0.1825703E+02   -0.1358626E+02    0.1919576E+02    0.3362283E+02
 -0.2515955E+02   -0.2048878E+02   -0.7792297E+01    0.8131846E+02


NUMBER OF SCALINGS (K) =  3
RELATIVE ERROR =  0.4E-07
EXECUTION TIME =     0.5 SEC
```

Table 6.13

Testmatrix 4 with n = 4

```
MATRIX A

 -0.6666667E+00  -0.6666667E+00   0.3333333E+00   0.1333333E+01   0.2333333E+01
 -0.1833333E+01   0.1166667E+01   0.1666667E+00   0.1166667E+01   0.2166667E+01
 -0.2000000E+01  -0.1000000E+01   0.3000000E+01   0.1000000E+01   0.2000000E+01
 -0.2166667E+01  -0.1166667E+01  -0.1666667E+00   0.4833333E+01   0.1833333E+01
 -0.2333333E+01  -0.1333333E+01  -0.3333333E+00   0.6666667E+00   0.6666667E+01

NORM(A)=    0.1133333E+02


EXACT MATRIX EXP(A)

 -0.3388385E+02  -0.3193135E+02  -0.1923487E+02   0.1527774E+02   0.1090927E+03
 -0.3738059E+02  -0.2532076E+02  -0.2001334E+02   0.1449928E+02   0.1083143E+03
 -0.3949667E+02  -0.3482590E+02  -0.2043880E+01   0.1238320E+02   0.1061982E+03
 -0.4524877E+02  -0.4057800E+02  -0.2788152E+02   0.6122924E+02   0.1004461E+03
 -0.6088461E+02  -0.5621383E+02  -0.4351735E+02  -0.9004740E+01   0.2332234E+03

NORM(EXP(A))=    0.4028440E+03



EXP(A) COMPUTED BY MEXP
NOTRAC=0

 -0.3388384E+02  -0.3193135E+02  -0.1923487E+02   0.1527774E+02   0.1090927E+03
 -0.3738058E+02  -0.2532076E+02  -0.2001333E+02   0.1449928E+02   0.1083143E+03
 -0.3949667E+02  -0.3482590E+02  -0.2043878E+01   0.1238320E+02   0.1061982E+03
 -0.4524877E+02  -0.4057799E+02  -0.2788152E+02   0.6122924E+02   0.1004461E+03
 -0.6088459E+02  -0.5621382E+02  -0.4351735E+02  -0.9004739E+01   0.2332234E+03


NUMBER OF SCALINGS (K) =   4
RELATIVE ERROR =   0.2E-06
EXECUTION TIME =     0.7 SEC
```

Table 6.14 (cont. next page)

Testmatrix 4 with n = 5

```
EXP(A) COMPUTED BY MEXP
NOTRAC=1

 -0.3388385E+02  -0.3193135E+02  -0.1923487E+02   0.1527774E+02   0.1090927E+03
 -0.3738059E+02  -0.2532076E+02  -0.2001334E+02   0.1449928E+02   0.1083143E+03
 -0.3949667E+02  -0.3482590E+02  -0.2043879E+01   0.1238320E+02   0.1061982E+03
 -0.4524877E+02  -0.4057800E+02  -0.2788152E+02   0.6122925E+02   0.1004461E+03
 -0.6088461E+02  -0.5621383E+02  -0.4351735E+02  -0.9004739E+01   0.2332234E+03

NUMBER OF SCALINGS (K) =  2
RELATIVE ERROR =  0.3E-07
EXECUTION TIME =     0.6 SEC


EXP(A) COMPUTED BY COSA

 -0.3388385E+02  -0.3193135E+02  -0.1923487E+02   0.1527774E+02   0.1090927E+03
 -0.3738059E+02  -0.2532076E+02  -0.2001334E+02   0.1449928E+02   0.1083143E+03
 -0.3949667E+02  -0.3482590E+02  -0.2043879E+01   0.1238320E+02   0.1061982E+03
 -0.4524877E+02  -0.4057800E+02  -0.2788152E+02   0.6122925E+02   0.1004461E+03
 -0.6088461E+02  -0.5621384E+02  -0.4351735E+02  -0.9004738E+01   0.2332234E+03

NUMBER OF SCALINGS (K) =  3
RELATIVE ERROR =  0.5E-07
EXECUTION TIME =     0.8 SEC
```

Table 6.14 (cont)

Test-matrix 4	MEXP,NOTRAC=0				MEXP,NOTRAC=1				COSA			
Order n	k	M	Exec. time, s.	$\delta\cdot10^8$	k	M	Exec. time, s.	$\delta\cdot10^8$	k	M	Exec. time, s.	$\delta\cdot10^8$
2	2	9	<0.1	2	0	7	<0.1	2	2	12	0.1	0.5
3	3	10	0.2	2	1	8	0.2	4	3	14	0.2	1
4	3	10	0.4	4	2	9	0.3	7	3	14	0.5	4
5	4	11	0.6	20	2	9	0.6	3	3	14	0.8	5
6	4	11	1.0	9	3	10	1.0	6	4	16	1.3	10
7	4	11	1.5	5	3	10	1.4	4	4	16	1.9	10
8	4	11	2.1	20	3	10	2.0	20	4	16	2.7	10
9	5	12	2.9	70	3	10	2.6	20	4	16	3.7	10
10	5	12	3.9	20	4	11	3.7	10	4	16	4.8	9
11	5	12	4.9	50	4	11	4.7	20	5	18	6.8	20
12	5	12	6.2	30	4	11	5.9	40	5	18	8.5	30
13	5	12	7.7	200	4	11	7.3	70	5	18	10.6	60

Table 6.15
Number of scalings k, total number of matrix multiplications M, execution time and relative error δ when computing exp(A), where A is testmatrix 4, with MEXP(NOTRAC=0 and NOTRAC = 1) and COSA.

Finally two examples of sampling a linear system with subroutine COSA will be given.

Consider the continuous linear system

$$\frac{dx}{dt} = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} x + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u \tag{6.7}$$

If the system is sampled with sampling interval τ, the discrete system will be:

$$x(t+\tau) = \begin{bmatrix} \cos\tau & \sin\tau \\ -\sin\tau & \cos\tau \end{bmatrix} x + \begin{bmatrix} 1-\cos\tau \\ \sin\tau \end{bmatrix} u \tag{6.8}$$

We obtain a special case if τ is a multiple of 2π:

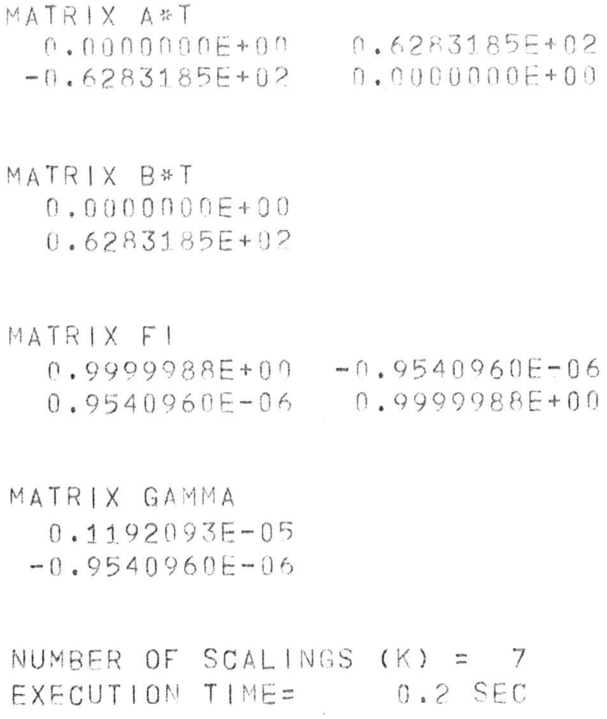
$$x(t+\tau) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} x + \begin{bmatrix} 0 \\ 0 \end{bmatrix} u \qquad (6.9)$$

where $\tau = k \cdot 2\pi$, $k = 0, \pm 1, \pm 2, \ldots$
Tables 6.16 and 6.17 show the results from COSA when $\tau = 2\pi$ and $\tau = 20\pi$.

```
MATRIX  A*T
   0.0000000E+00    0.6283185E+01
  -0.6283185E+01    0.0000000E+00


MATRIX  B*T
   0.0000000E+00
   0.6283185E+01


MATRIX  FI
   0.1000000E+01   -0.3064813E-06
   0.3064813E-06    0.1000000E+01


MATRIX  GAMMA
  -0.1192093E-06
  -0.3064813E-06


NUMBER  OF  SCALINGS  (K)  =   3
EXECUTION  TIME=       0.2  SEC
```

Table 6.16.

Result from subroutine COSA. The sampling interval τ is 2π.



```
MATRIX A*T
  0.0000000E+00    0.6283185E+02
 -0.6283185E+02    0.0000000E+00


MATRIX B*T
  0.0000000E+00
  0.6283185E+02


MATRIX FI
  0.9999988E+00   -0.9540960E-06
  0.9540960E-06    0.9999988E+00


MATRIX GAMMA
  0.1192093E-05
 -0.9540960E-06


NUMBER OF SCALINGS (K) =  7
EXECUTION TIME=      0.2 SEC
```

Table 6.17

Result from subroutine COSA. The sampling interval τ is 20π.

Conclusions

The numerical examples show, that the results obtained from subroutines MEXP and COSA often are very accurate, with regard to the accuracy of the computer. The execution times are short, in spite of the fact that the computer uses software real arithmetic.

When computing exp(A), subroutine MEXP with origin shift (NOTRAC = 1) often gives the shortest computing time, while subroutine COSA gives the longest computing time. The longer execution time is due to the fact that the scaling procedure is more complicated in COSA. Nine terms are used when evaluating $\Psi(A)$ in COSA, but to obtain exp(A) from $\Psi(A)$ requires another matrix multiplication. This is comparable to use ten terms in the series expansion of exp(A), and this implies that the achieved accuracy of evaluating exp(A) by COSA sometimes is higher than evaluating exp(A) with MEXP.

If we compare the accuracy of evaluating exp(A) by MEXP with and without origin shift, we can conclude that the shift often implies a higher accuracy. There are, however, applications when the matrix A is known to have the eigenvalues centered around the origin, and the option NOTRAC therefore is a way to get shorter computing times in these cases, by avoiding an unnecessary trace computation.

7. REFERENCES

[1] Bellman, R.: Introduction to Matrix Analysis, McGraw-Hill, 1960.

[2] Bickart, T.A.: Matrix Exponential: Approximation by Truncated Power Series, Proc. IEEE (Lett.), Vol. 56, May 1968, pp. 872--873.

[3] Brockett, R.W.: Finite Dimensional Linear Systems, Wiley, 1970.

[4] Choudary, A.K. et al.: On the Evaluation of $e^{A\tau}$, Proc. IEEE, Vol. 56, June 1968, pp. 1110-1111.

[5] Everling, W.L.: On the Evaluation of e^{At} by Power Series, Proc. IEEE, Vol. 55, March 1967, p. 413.

[6] Faddeev, D.K. and Faddeeva, V.N.: Computational Methods of Linear Algebra, Freeman, 1963.

[7] Fath, A.F.: Evaluation of a Matrix Polynomial, IEEE Trans. on Automatic Control, April 1968, Vol. AC-13, pp. 220-221.

[8] Gregory, R.T. and Karney, D.L.: A Collection of Matrices for Testing Computational Algorithms, Wiley-Interscience, 1969.

[9] Johnson, I.C. and Phillips, C.L.: Algorithm for the Computation of the Integral of the Transition Matrix, IEEE Trans. on Automatic Control, April 1921, pp. 204-205.

[10] Kalman, R.E. and Englar, T.S.: A User's Manual for the Automatic Synthesis Program, NASA CR-475, June 1966.

[11] Larsson, L.O.: Algoritmer för lösning av system av ordinära differentialekvationer, Report RE-77, Division of Automatic Control, Lund Institute of Technology, 1970.

[12] Levis, A.H.: Some Computational Aspects of the Matrix Exponential, IEEE Trans. on Automatic Control, August 1969, pp. 410--411.

[13] Liou, M.L.: A Novel Method of Evaluating Transient Response, Proc. IEEE, Vol. 54, January 1966, pp. 20-23.

[14] Liou, M.L.: Evaluation of the Transition Matrix, Proc. IEEE, Vol. 55, February 1967, pp. 228-229.

[15] Liou, M.L. and Alderson, G.E.: Evaluation of State Transition Matrix and Related Topics, 2nd Automatic Conf. IEEE, 1968.

[16] Mandal, A.K. et al.: Numerical Computation Method for the Evaluation of the Transition Matrix, Proc. IEE, Vol. 116, April 1969, pp. 500-502.

[17] Mankin, J.B. and Hung, J.C.: On Round-off Errors in Computation of Transition Matrices, J.A.C.C. 1968, pp. 60-64.

[18] Mårtensson, K.: Linear Quadratic Control Package, Part I - The Continuous Problem, Report 6802, Division of Automatic Control, Lund Institute of Technology, 1968.

[19] Program Library, Division of Automatic Control, Lund Institute of Technology.

[20] Shah, M.M.: On the Evaluation of e^{At}, Cambridge Report CUED/B-- Control/TR8, 1971.

[21] Shah, M.M.: Analysis of Round-off and Truncation Errors in the Computation of Transition Matrices, Cambridge Report CUED/B-- Control/TR12, 1971.

[22] Åström, K.J.: On the Choice of Sampling Rates in Optimal Linear Systems, IBM Research Report RJ-243, April 1963.

[23] Åström, K.J.: Reglerteori, Almqvist o. Wiksell, Stockholm, 1968.

APPENDIX A

SUMMARY OF CHARACTERISTICS FOR THE COMPUTER PDP 15/35

Word length = 18 bits
Real single precision variabler= 2 words
Real double precision variable = 3 words
Mantissa of real single precision variable = 27 bits (incl. 1 sign bit) = 7 - 8 dicimal digits.
Note that the real precision Fortran Library Functions sometimes only give 6 decimal digits.
Mantissa of real double precision variable = 36 bits (incl. 1 sign bit) ≈10 decimal digits

Matrix multiplications are partly performed in double precision.
Fortran IV compiler: F4S V12D.
Memory cycle time = 0.8 μs

Execution times óf some subroutines in the real arithmetic package:
Floating load = 40 μs
Floating store = 44 μs
Floating add = 192 μs
Floating subtract = 240 μs
Floating multiply = 190 μs
Floating divide = 240 μs

Execution times to
calculate the address of an array element:
2 dimensional array = 94 μs
3 dimensional array = 116 μs

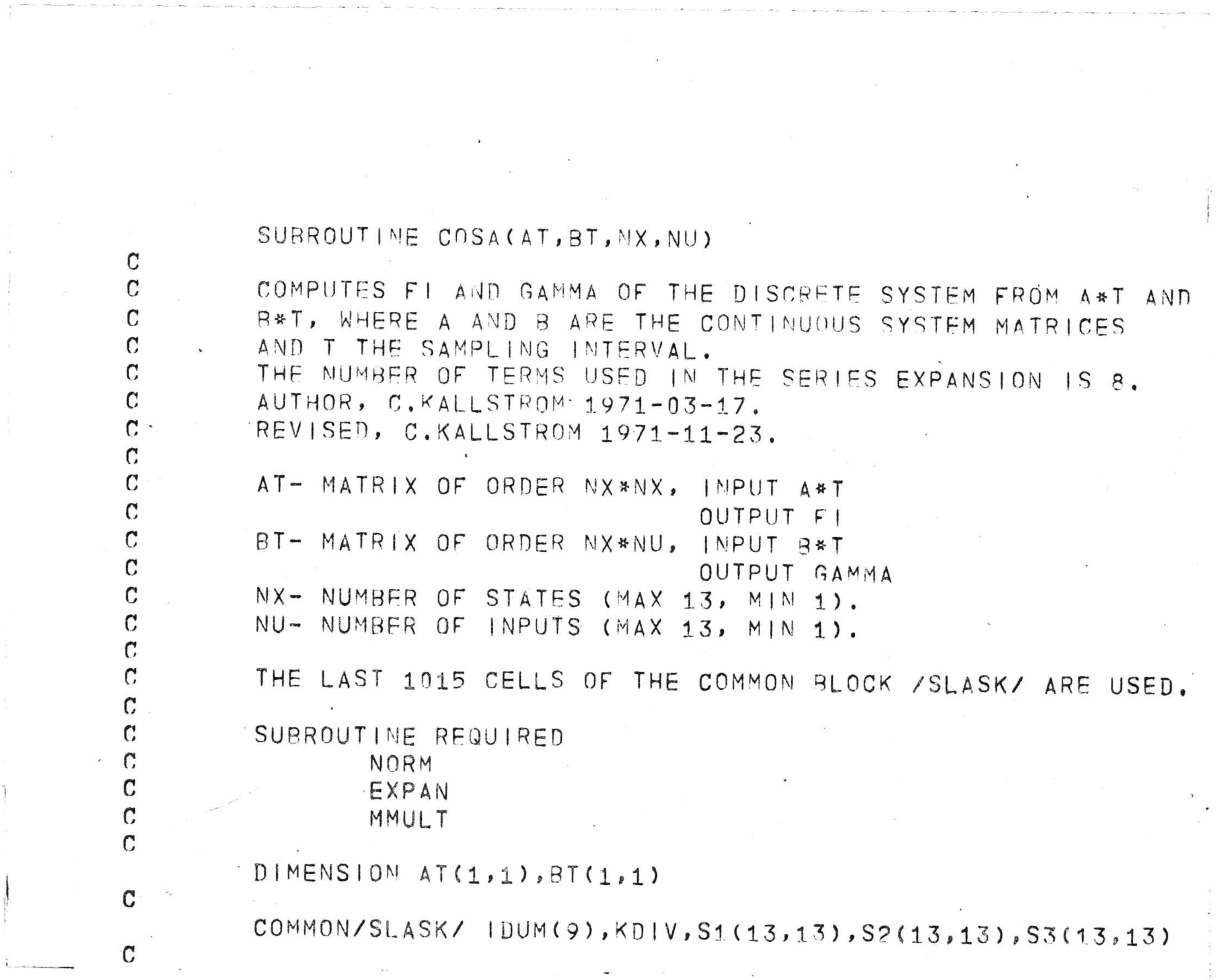
APPENDIX B

```
      SUBROUTINE EXPAN(A,EA,N,NLOOP,R)
C
C     COMPUTES EXP(A) = I + A + A**2/(1*2) + ...
C     OR GAMMA(A) = I + A/(1*2) + A**2/(1*2*3) + ...
C     AUTHOR, C.KALLSTROM 1971-02-25.
C     REVISED, C.KALLSTROM 1971-10-15.
C
C     A- MATRIX OF ORDER N*N, NOT DESTROYED.
C     EA- RETURNS EXP(A) OR GAMMA(A) OF ORDER N*N.
C     N- ORDER OF A (MAX 13, MIN 1).
C     NLOOP- THE DESIRED NUMBER OF TERMS (EXCEPT THE IDENTITY MATRIX)
C        OF THE SERIES EXPANSION (NO MAX, MIN 2).
C     R- PUT R=0. IF EXP(A).
C        PUT R=1. IF GAMMA(A).
C
C     THE LAST 338 CELLS OF THE COMMON BLOCK /SLASK/ ARE USED.
C
C     SUBROUTINE REQUIRED
C            NONE
C
      DIMENSION A(1,1),EA(1,1)
C
      COMMON/SLASK/ IDUM(686),S1(13,13)
C
```

APPENDIX C

```
      SUBROUTINE MEXP(A,EA,N,IE,NOTRAC)
C
C     COMPUTES EA=EXP(A) BY ORIGIN SHIFT AND SERIES EXPANSION
C     USING 8 TERMS.
C     AUTHOR, C.KALLSTROM 1971-03-15.
C     REVISED, C.KALLSTROM 1971-11-23.
C
C     A- MATRIX OF ORDER N*N, NOT DESTROYED.
C     EA- MATRIX OF ORDER N*N.
C     N- ORDER OF THE MATRICES A AND EA (MAX 13, MIN 1).
C     IE- DIMENSION PARAMETER OF EA (THE DIMENSION OF A NEED NOT
C        BE THE SAME).
C     NOTRAC- INPUT PARAMETER:
C        NOTRAC=0 : NO SHIFT WILL BE PERFORMED.
C        NOTRAC=1 : ORIGIN SHIFT WILL BE PERFORMED.
C
C     THE LAST 677 CELLS OF THE COMMON BLOCK /SLASK/ ARE USED.
C
C     NOTE: IF ABS(TRACE(A)/N) .GT. 170 NO ORIGIN SHIFT
C     WILL BE PERFORMED EVEN IF NOTRAC=1.
C
C     THE ACTUAL ARGUMENTS A AND EA CAN BE EQUIVALENCED
C     IN THE CALLING PROGRAM.
C
C     SUBROUTINE REQUIRED
C            NORM
C            EXPAN
C            MMULT
C
      DIMENSION A(1,1),EA(1,1)
C
      COMMON/SLASK/ IDUM(347),KDIV,S1(13,13),S2(13,13)
C
```

APPENDIX D



```
      SUBROUTINE COSA(AT,BT,NX,NU)
C
C     COMPUTES FI AND GAMMA OF THE DISCRETE SYSTEM FROM A*T AND
C     B*T, WHERE A AND B ARE THE CONTINUOUS SYSTEM MATRICES
C     AND T THE SAMPLING INTERVAL.
C     THE NUMBER OF TERMS USED IN THE SERIES EXPANSION IS 8.
C     AUTHOR, C.KALLSTROM 1971-03-17.
C     REVISED, C.KALLSTROM 1971-11-23.
C
C     AT- MATRIX OF ORDER NX*NX, INPUT A*T
C                                OUTPUT FI
C     BT- MATRIX OF ORDER NX*NU, INPUT B*T
C                                OUTPUT GAMMA
C     NX- NUMBER OF STATES (MAX 13, MIN 1).
C     NU- NUMBER OF INPUTS (MAX 13, MIN 1).
C
C     THE LAST 1015 CELLS OF THE COMMON BLOCK /SLASK/ ARE USED.
C
C     SUBROUTINE REQUIRED
C            NORM
C            EXPAN
C            MMULT
C
      DIMENSION AT(1,1),BT(1,1)
C
      COMMON/SLASK/ IDUM(9),KDIV,S1(13,13),S2(13,13),S3(13,13)
C
```

APPENDIX E

During the development of this report, the PDP 15/35 computer has been supplied with hardware floating point arithmetic. The new execution times when evaluating e^A for test matrices 3 and 4 are given in Table 8.1 to illustrate how much faster the computations are performed.

Order n	MEXP, NOTRAC=0	MEXP, NOTRAC=1	COSA
Test matrix 3			
2	<0.1	<0.1	<0.1
3	0.1	0.1	0.1
4	0.2	0.2	0.2
5	0.3	0.3	0.4
6	0.5	0.5	0.6
7	0.7	0.7	0.9
8	1.0	1.0	1.3
9	1.4	1.3	1.7
10	1.8	1.8	2.2
11	2.2	2.1	2.7
12	2.8	2.6	3.7
13	3.4	3.2	4.6
Test matrix 4			
2	<0.1	<0.1	<0.1
3	0.1	0.1	0.1
4	0.2	0.2	0.2
5	0.3	0.3	0.4
6	0.5	0.5	0.6
7	0.7	0.7	0.9
8	1.0	0.9	1.3
9	1.4	1.2	1.7
10	1.8	1.7	2.2
11	2.2	2.1	2.9
12	2.7	2.6	3.7
13	3.3	3.2	4.5

Table 8.1. Execution times in seconds of evaluating e^A for test matrices 3 and 4, when the PDP 15/35 computer uses hardware floating point arithmetic. Cf. Tables 6.12 and 6.15.