Jan Ježek

An efficient algorithm for computing real powers of a matrix and a related matrix function

Aplikace matematiky, Vol. 33 (1988), No. 1, 22–32

Persistent URL: `http://dml.cz/dmlcz/104283`

AN EFFICIENT ALGORITHM FOR COMPUTING REAL POWER OF A MATRIX AND A RELATED MATRIX FUNCTION

JAN JEŽEK

(Received September 10, 1986)

Summary. The paper is devoted to an algorithm for computing matrices A^r and $(A^r - I) . (A - I)^{-1}$ for a given square matrix A and a real r. The algorithm uses the binary expansion of r and has the logarithmic computational complexity with respect to r. The problem stems from the control theory.

Keywords: matrix algebra, matrix function, matrix power, computational complexity.

INTRODUCTION

In the paper, two numerical algorithms are described. The first computes $R = A^r$ for a given real $m \times m$ matrix A and for real r. The second algorithm computes $S = (A^r - I)(A - I)^{-1}$, $r \geqq 0$.

The work is motivated by needs of the control theory. A linear system to be controlled is usually of the form

$$\frac{dx(t)}{dt} = A\,x(t) + B\,u(t) \tag{1}$$

where the vector $x(t)$ denotes the state and the scalar $u(t)$ the input signal; the matrix A and the vector B are constant. For a fixed $T > 0$, let $u(t) = u_n$ be constant in every interval $nT \leqq t < (n + 1)\,T$. Denoting $x_n = x(nT)$, we can replace equation (1) by that for discrete signals:

$$x_{n+1} = Fx_n + Gu_n \tag{2}$$

where

$$F = e^{AT}, \quad G = (e^{AT} - I)\,A^{-1}B\,. \tag{3}$$

Formulas (3) provide the conversion from A, B to F, G (depending on T). The inverse conversion is given by

$$A = \frac{1}{T}\ln F\,, \quad B = \frac{1}{T}\ln F(F - I)^{-1}\,G\,. \tag{4}$$

For two intervals defined by T_1, T_2, the mutual conversions are

(5) $$F_2 = F_1^r\,, \quad G_2 = (F_1^r - I)(F_1 - I)^{-1}\, G_1$$

with $r = T_2/T_1$. That is where the functions to be computed come from.

The notions of the matrix power and logarithm call for precise definition. It can be done in terms of matrix functions [1]: for every function $f(a)$ of complex variable, analytic on the spectrum of A, the function $f(A)$ is defined via the Jordan form:

$$A = T^{-1}JT\,, \quad J = \operatorname*{diag}_i J_i\,,$$

$$f(A) = T^{-1} f(J)\, T\,, \quad f(J) = \operatorname*{diag}_i f(J_i)\,,$$

(6) $$J_i = \begin{bmatrix} a_i & 1 & & \\ & \ddots & \ddots & \\ & & \ddots & 1 \\ & & & a_i \end{bmatrix}, \quad f(J_i) = \begin{bmatrix} f(a_i), & f'(a_i), & \dfrac{f''(a_i)}{2}, & \dots \\ & \ddots & \ddots & \ddots \\ & & \ddots & \ddots \\ & & & \ddots \end{bmatrix}.$$

In our case, $f(a)$ is the main value of a^r defined for all complex a with the exception of $a \leqq 0$ real:

(7) $$r = c + z\,, \quad c = \dots -1, 0, 1, \dots, \quad 0 \leqq z < 1\,,$$

(8) $$a = |a|\,(\cos\varphi + \mathrm{i}\sin\varphi)\,, \quad -\pi < \varphi < \pi\,,$$

(9) $$a^r = a^c a^z\,, \quad a^z = |a|^z\,(\cos z\varphi + \mathrm{i}\sin z\varphi)\,.$$

Similarly, the function S can be defined in the same region. Note that $A - I$ may well be singular because the function $g(a) = (a^r - 1)(a - 1)^{-1}$ has a removable singularity for $a = 1$. The matrix logarithm is defined via the function $h(a)$ for (8):

(10) $$h(a) = \ln(a) = \ln|a| + \mathrm{i}\varphi\,.$$

In the sequel, the notation A pwr X is used instead of A^X when X is a complicated expression.

ALGORITHM FOR $R = A^r$

The definition of A^r is not suitable for numerical computation because of the need to know or to compute the eigenvalues and eigenvectors, generally complex. Another possibility is to express $A^r = \mathrm{e}^{r\ln A}$ and to use algorithms for matrix exponentiation and logarithm [4] but this procedure is too complicated and numerically not satisfactory. A better way is to employ the algorithm given in the sequel which utilizes the square and the square root of a matrix.

For $r < 0$, $A^r = (A^{-1})^{-r}$ can be taken, for $r = 0$, $A^0 = I$. So it is sufficient to consider $r > 0$. Decompose r into the integer and non-integer part as in (7); then $A^r = A^c A^z$. The contribution of the integer part can be computed via successive multiplication by A; the number of operations needed grows proportionally to c.

However, the consumption of operations can be dramatically reduced by using the binary form of c:

$$c = \sum_{i=0}^{f} f_i 2^i, \quad f_i = \left\langle \begin{matrix} 0 \\ 1 \end{matrix} \right. . \tag{11}$$

For $i = 0, 1, \ldots, f$, a sequence Q_i is constructed:

$$Q_0 = A, \quad Q_{i+1} = Q_i^2, \tag{12}$$

yielding

$$A^c = \prod_{i=0}^{f} Q_i^{f_i}. \tag{13}$$

Note. For practical computation, the exponent $f_i = 1$ means: include the factor, $f_i = 0$: do not include it.

The proof of the algorithm is simple: $Q_i = A^{2^i}$ is easily proved by induction, then

$$\prod_{i=0}^{f} Q_i^{f_i} = \prod_{i=0}^{f} A^{2^i f_i} = A \operatorname{pwr}\left(\sum_{i=0}^{f} 2^i f_i\right) = A^c. \tag{14}$$

The number of operations needed is proportional to f, i.e. to $\log c$.

Similarly, the contribution of the non-integer part is computed. For a start, let the binary expansion of z be finite:

$$z = \sum_{i=1}^{g} g_i 2^{-i}, \quad g_i = \left\langle \begin{matrix} 0 \\ 1 \end{matrix} \right. . \tag{15}$$

A sequence Q_i is constructed for $i = 0, 1, \ldots, g$:

$$Q_0 = A, \quad Q_{i+1} = \sqrt{Q_i}, \tag{16}$$

yielding

$$A^z = \prod_{i=1}^{g} Q_i^{g_i}. \tag{17}$$

For the proof, $Q_i = A^{2^{-i}}$ and

$$\prod_{i=1}^{g} Q_i^{g_i} = \prod_{i=1}^{g} A^{2^{-i} g_i} = A \operatorname{pwr}\left(\sum_{i=1}^{g} 2^{-i} g_i\right) = A^z. \tag{18}$$

Now, for a general z the expansion (15) is infinite: $g \to \infty$. It is evident that $Q_g \to I$; the product in (18) is convergent because the sum in the exponent is. Practically, g is given by the computer word length. The number of operations is proportional to g, i.e. to $-\log \varepsilon$, where $\varepsilon = 2^{-g}$ is the resolution of the computer.

By the matrix square root, its main value is understood. It is computed via an iterative process based on the Newton method [4]. Before computing A^z, a good idea is to scale the matrix so that its determinant be 1 (under the above conditions, we always have $\det A > 0$):

$$d = \sqrt[m]{\det A}\,, \quad A^z = \left(\frac{A}{d}\right)^z d^z\,. \tag{19}$$

This trick facilitates a simple selection of an initial value in the iterative process for the matrix square root. During successive square-rooting of Q_i, the property $\det Q_i = 1$ is preserved.

ALGORITHM FOR $S_r = (A^r - I)(A - I)^{-1}$

This matrix function can be computed by making use of the above algorithm for A^r. However, this method fails for $A = I$ and in the neighbourhood of that case. As it is just this domain which is the most interesting from the point of view of application, an independent algorithm for S_r was developed. It also has the logarithmic grow of number of operations.

The algorithm is presented in the form of two theorems. The former deals with an integer exponent, the latter covers the general case of non-integer exponent.

Theorem 1. *Let A be a real $m \times m$ matrix, $c \geqq 0$ an integer; denote f, f_i as in (11). For $i = 0, 1, \ldots, f - 1$ construct a sequence Q_i:*

$$Q_0 = A\,, \quad Q_{i+1} = Q_i^2\,. \tag{20}$$

For $i = 0, 1, \ldots, f$ construct a sequence T_i:

$$T_0 = I\,, \quad T_{i+1} = Q_i^{f_i}(I + Q_i)\,T_i\,. \tag{21}$$

Then

$$S_c = (A^c - I)(A - I)^{-1} = \sum_{j=0}^{c-1} A^j = \sum_{i=0}^{f} f_i T_i\,. \tag{22}$$

Note. For practical computation, the factor $f_i = 1$ in (22) means: include the term, $f_i = 0$: do not include it.

Proof. The number f will be called the order of c. Denote by c_i, $i = -1, 0, 1, \ldots, f$, the sequence of integer numbers obtained by successively including the binary digits of c:

$$c_i = \sum_{k=0}^{i} f_k 2^k\,, \quad c_{-1} = 0\,, \quad c_f = c\,. \tag{23}$$

Every c_i is of order i. A recurrent relation

$$c_i = c_{i-1} + f_i 2^i \tag{24}$$

holds. First,

$$S_c = \sum_{i=0}^{f} \sum_{j=c_{i-1}}^{c_i - 1} A^j \tag{25}$$

will be proved by induction on f. For $f = -1$, we have $c = 0$, $S_c = 0$. Let (25) hold for all numbers of order f; take c of order $f + 1$:

$$\sum_{i=0}^{f+1} \sum_{j=c_{i-1}}^{c_i - 1} A^j = \sum_{i=0}^{f} \sum_{j=c_{i-1}}^{c_i - 1} A^j + \sum_{j=c_f}^{c_{f+1}-1} A^j = \tag{26}$$

$$= \sum_{j=0}^{c_f - 1} A^j + \sum_{j=c_f}^{c_{f+1}-1} A^j = \sum_{j=0}^{c_{f+1}-1} A^j = S_c\,.$$

Formula (25) is proved. Next, rearrange it with help of (24):

$$S_c = \sum_{i=0}^{f} \sum_{j=c_{i-1}}^{c_i - 1} A^j = \sum_{i=0}^{f} A^{c_{i-1}} \sum_{j=0}^{c_i - c_{i-1} - 1} A^j = \tag{27}$$

$$= \sum_{i=0}^{f} A^{c_{i-1}} \sum_{j=0}^{f_i 2^i - 1} A^j = \sum_{i=0}^{f} f_i A^{c_{i-1}} \sum_{j=0}^{2^i - 1} A^j\,.$$

The last sum can be written in the form

$$\sum_{j=0}^{2^i - 1} A^j = \prod_{k=0}^{i-1} (I + A^{2^k})\,. \tag{28}$$

This will be proved by induction on i. For $i = 0$, both sides are equal to 1. For $i + 1$:

$$\sum_{j=0}^{2^{i+1}-1} A^j = \sum_{j=0}^{2^i - 1} A^j + \sum_{j=2^i}^{2^i + 2^i - 1} A^j = (I + A^{2^i}) \sum_{j=0}^{2^i - 1} A^j = \tag{29}$$

$$= (I + A^{2^i}) \prod_{k=0}^{i-1} (I + A^{2^k}) = \prod_{k=0}^{i} (I + A^{2^k})\,.$$

Formula (28) is proved. With its help, (27) can be further modified to

$$S_c = \sum_{i=0}^{f} f_i A \operatorname{pwr}\Big(\sum_{k=0}^{i-1} f_k 2^k\Big) \sum_{j=0}^{2^i - 1} A^j = \sum_{i=0}^{f} f_i \Big[\prod_{k=0}^{i-1} A^{f_k 2^k}\Big] \Big[\prod_{k=0}^{i-1} (I + A^{2^k})\Big]\,, \tag{30}$$

$$S_c = \sum_{i=0}^{f} f_i \prod_{k=0}^{i-1} A^{f_k 2^k} (I + A^{2^k})\,. \tag{31}$$

This form is exactly what formulas (20), (21) for Q_i, T_i yield. It can be seen with formulas

$$Q_i = A^{2^i}\,, \quad T_i = \prod_{k=0}^{i-1} Q_k^{f_k} (I + Q_k) \tag{32}$$

which can be easily proved by induction. □

Example.

$c = 21 = 10101_2\,, \qquad f = 4\,,$

$$f_0 = 1\,, \quad f_1 = 0\,, \quad f_2 = 1\,, \quad f_3 = 0\,, \quad f_4 = 1\,,$$

$$c_0 = 1\,, \quad c_1 = 1\,, \quad c_2 = 5\,, \quad c_3 = 5\,, \quad c_4 = 21\,,$$

$$\begin{aligned} S_{21} &= I + \ldots + A^{20} = I + (A + \ldots + A^4) + (A^5 + \ldots + A^{20}) = \\ &= I + A^1(I + \ldots + A^3) + A^{1+4}(I + \ldots + A^{15}) = \\ &= I + A^1(I + A)(I + A^2) + A^{1+4}(I + A)(I + A^2)(I + A^4)(I + A^8)\,, \end{aligned}$$

$$\begin{aligned} Q_0 &= A\,, & T_0 &= I\,, \\ Q_1 &= A^2\,, & T_1 &= A^1(I + A)\,, \\ Q_2 &= A^4\,, & T_2 &= A^1(I + A)(I + A^2)\,, \\ Q_3 &= A^8\,, & T_3 &= A^1(I + A)(I + A^2)(I + A^4)\,, \\ & & T_4 &= A^{1+4}(I + A)(I + A^2)(I + A^4)(I + A^8)\,. \end{aligned}$$

Theorem 2. *Let A be a real $m \times m$ matrix with no real nonpositive eigenvalue. Let $r \geqq 0$ be a real whose binary expansion is finite. Denote c, z, g, g_i as in (7), (15). Construct sequences Q_i, B_i, D_i for $i = 0, 1, \ldots, g$ and G_i for $i = 0, 1, \ldots, g - 1$:*

(33) $$Q_0 = A\,, \qquad Q_i = Q_{i-1}^{2^{-1}}\,,$$

(34) $$B_0 = I\,, \qquad B_i = 2^{-1}(I + Q_i)\, B_{i-1}\,,$$

(35) $$G_0 = A^c\,, \qquad G_i = 2^{-1} Q_i^{g_i} G_{i-1}\,,$$

(36) $$D_0 = (A^c - I)(A - I)^{-1}\,, \quad D_i = 2^{-1}[(I + Q_i)\, D_{i-1} + g_i G_{i-1}]\,.$$

Then all B_i are nonsingular and

(37) $$S_r = (A^r - I)(A - I)^{-1} = D_g B_g^{-1}\,.$$

Moreover, for general r whose binary expression is infinite, all the sequences are convergent for $g \to \infty$, B_∞ remains nonsingular, (37) remains valid.

Proof. Denote by z_i, $i = 0, 1, \ldots, g$ the sequence of real numbers obtained by successively including the binary digits of z:

(38) $$z_i = \sum_{k=1}^{i} g_k\, 2^{-k}\,, \quad z_0 = 0\,, \quad z_g = z\,.$$

The recurrent relation

(39) $$z_i = z_{i-1} + g_i\, 2^{-i}$$

is evident. Furthermore, introduce integers z_i':

(40) $$z_i = 2^{-i} z_i'\,, \quad z_i' = \sum_{k=1}^{i} g_k\, 2^{i-k}\,.$$

Formulas (39) and (7) have the form

(41) $$z_i' = 2z_{i-1}' + g_i ,$$

(42) $$r = c + 2^{-g}z' = 2^{-g}(2^g c + z') .$$

Rearrange (37):

(43) $$S_r = (A^r - I)(A - I)^{-1} =$$

$$= 2^{-g}[(A^{2^{-g}})^{2^g c + z'} - I]\,[A^{2^{-g}} - I]^{-1}\,[A^{2^{-g}} - I]\,[(A^{2^{-g}})^{2^g} - I]^{-1}\,2^g =$$

$$= [2^{-g}\sum_{j=0}^{2^g c + z' - 1}(A^{2^{-g}})^j]\,[2^{-g}\sum_{j=0}^{2^g - 1}(A^{2^{-g}})^j]^{-1} = \hat{D}_g \hat{B}_g^{-1} .$$

$\hat{D}_g$ can be rearranged, too:

(44) $$\hat{D}_g = 2^{-g}\sum_{j=0}^{2^g c - 1}(A^{2^{-g}})^j + 2^{-g}\sum_{j=2^g c}^{2^g c + z' - 1}(A^{2^{-g}})^j .$$

In the first sum, introduce two summation indices k, j' as the quotient and the remainder $j = 2^g k + j'$. In the second sum, introduce j' by $j = 2^g c + j'$:

(45) $$\hat{D}_g = 2^{-g}\sum_{k=0}^{c-1}\sum_{j'=0}^{2^g - 1}(A^{2^{-g}})^{2^g k + j'} + 2^{-g}\sum_{j'=0}^{z'-1}(A^{2^{-g}})^{2^g c + j'} =$$

$$= \sum_{k=0}^{c-1} A^k\,2^{-g}\sum_{j=0}^{2^g - 1}(A^{2^{-g}})^j + 2^{-g}A^c\sum_{j=0}^{z'-1}(A^{2^{-g}})^j .$$

For $\hat{B}_g$, the following recurrent formula is used:

(46) $$\sum_{j=0}^{2^g - 1}(A^{2^{-g}})^j = \prod_{i=1}^{g}(I + A^{2^{-i}}) .$$

Using (28) we prove

(47) $$\prod_{i=1}^{g}(I + A^{2^{-i}}) = \prod_{i=1}^{g}[I + (A^{2^{-g}})^{2^{g-i}}] =$$

$$= \prod_{i'=0}^{g-1}[I + (A^{2^{-g}})^{2^{i'}}] = \sum_{j=0}^{2^g - 1}(A^{2^{-g}})^j .$$

Now we shall prove that $\hat{B}_g$, $\hat{D}_g$ is what the formulas (33)–(36) for Q_i, B_i, G_i, D_i yield, i.e. that $\hat{B}_g = B_g$, $\hat{D}_g = D_g$. The formulas

(48) $$Q_i = A^{2^{-i}}, \quad B_i = 2^{-i}\prod_{k=1}^{i}(I + A^{2^{-k}}) = 2^{-i}\sum_{j=0}^{2^i - 1}(A^{2^{-g}})^j$$

can be easily proved by induction. For G_i, we evidently have

(49) $$G_i = 2^{-i}A^c\prod_{k=1}^{i} Q_k^{g_k} = 2^{-i}A^c\prod_{k=1}^{i} A^{g_k 2^{-k}} = 2^{-i}A\,\mathrm{pwr}(c + \sum_{k=1}^{i} g_k\,2^{-k}) = 2^{-i}A^{c+z_i} .$$

The formula for D_i remains to be proved:

(50) $$D_i = S_c B_i + 2^{-i} A^c \sum_{j=0}^{z_i'-1} (A^{2^{-i}})^j .$$

Induction by i: for $i = 0$ we have $D_0 = S_c$ as required. Let (50) hold for D_{i-1} and compute D_i by (36):

(51) $$D_i = 2^{-1}(I + A^{2^{-i}}) [S_c B_{i-1} + 2^{-i+1} A^c \sum_{j=0}^{z_{i-1}'-1} (A^{2^{-i+1}})^j] +$$
$$+ g_i 2^{-1} 2^{-i+1} A^{c+z_{i-1}} = S_c B_i +$$
$$+ 2^{-i} A^c [\sum_{j=0}^{z_{i-1}'-1} A^{2^{-i+1}j} + \sum_{j=0}^{z_{i-1}'-1} A^{2^{-i}+2^{-i+1}j} + g_i A^{z_{i-1}}] .$$

Denote the bracket by E_i and work on it:

(52) $$E_i = \sum_{j=0}^{z_{i-1}'-1} A^{2^{-i}2j} + \sum_{j=0}^{z_{i-1}'-1} A^{2^{-i}(2j+1)} + g_i A^{2^{-i+1} z_{i-1}'} .$$

Summation indices $j' = 2j$ and $j' = 2j + 1$ are introduced:

(53) $$E_i = \sum_{\substack{j'=0 \\ j'\text{even}}}^{2z_{i-1}'-2} A^{2^{-i}j'} + \sum_{\substack{j'=1 \\ j'\text{odd}}}^{2z_{i-1}'-1} A^{2^{-i}j'} + g_i A^{2^{-i}2z_{i-1}'} =$$
$$= \sum_{j=0}^{2z_{i-1}'-1} A^{2^{-i}j} + g_i A^{2^{-i}2z_{i-1}'} .$$

The last term fits into the sum with $j = 2z_{i-1}'$ but only when $g_i = 1$. In that case, the upper bound is $2z_{i-1}'$; in the case $g_i = 0$ the bound $2z_{i-1}' - 1$ remains. We can write

(54) $$E_i = \sum_{j=0}^{2z_{i-1}'-1+g_i} A^{2^{-i}j} = \sum_{j=0}^{z_{i-1}'} A^{2^{-i}j} .$$

This completes the induction for D_i. Now, comparing (48) with (43), $\hat{B}_g = B_g$ is evident. From (50) and (45), $\hat{D}_g = D_g$ follows.

To prove nonsingularity of B_i, it suffices to prove nonsingularity of all $I + A^{2^{-k}}$, see (48). Suppose some $I + A^{2^{-k}}$ is singular, i.e. $1 + a^{2^{-k}} = 0$ for some eigenvalue a of A, i.e. $a^{2^{-k}} = -1$, $\arg a^{2^{-k}} = \pi$. However, $-\pi < \arg a < \pi$ was supposed, hence $-2^{-k}\pi < \arg a^{2^{-k}} < 2^{-k}\pi$.

As for the convergence $g \to \infty$, it is clear that $Q_g \to I$, $G_g \to 0$; B_g, D_g are to be investigated:

(55) $$\lim_{g\to\infty} B_g = \lim_{g\to\infty} 2^{-g}(A - I)(A^{2^{-g}} - I)^{-1} = (A - I)[\lim_{g\to\infty} 2^g(A^{2^{-g}} - I)]^{-1} =$$
$$= (A - I)[\lim_{h\to\infty} h(A^{1/h} - I)]^{-1} = (A - I)\left[\lim_{x\to 0} \frac{A^x - I}{x}\right]^{-1} = (A - I)(\ln A)^{-1} .$$

Similarly,

(56) $$\lim_{g\to\infty} D_g = \lim_{g\to\infty} 2^{-g}(A^{c+2^{-g}z_g'} - I)(A^{2^{-g}} - I)^{-1} = (A^r - I)(\ln A)^{-1} .$$

The convergence is proved. By virtue of (55), (56), B_∞, D_∞ evidently exist even if some eigenvalue $a = 1$ and $\ln A$ is singular. At that point, the function $l(a) = (a - 1)/\ln a$ has a removable singularity. The matrix B_∞ is nonsingular because $l(a)$ never vanishes (a is never real nonpositive). It follows from (55), (56) that (37) remains valid. □

In this algorithm, it is also useful to scale the matrix whose square root is to be taken so that its determinant be 1. The algorithm gets modified:

(57) $$d_0 = \sqrt[m]{\det A}\,, \qquad d_i = d_{i-1}^{2^{-1}}\,,$$

(58) $$Q_0 = \frac{A}{d_0}\,, \qquad Q_i = Q_{i-1}^{2^{-1}}\,,$$

(59) $$B_0 = I\,, \qquad B_i = 2^{-1}(I + d_iQ_i)\,B_{i-1}\,,$$

(60) $$G_0 = A^c\,, \qquad G_i = 2^{-1}(d_iQ_i)^{g_i}\,G_{i-1}\,,$$

(61) $$D_0 = (A^c - I)(A - I)^{-1}\,, \quad D_i = 2^{-1}[(I + d_iQ_i)\,D_{i-1} + g_iG_{i-1}]\,.$$

It is easy to see that it is equivalent to (33)–(36).

Example.

$$r = \tfrac{61}{16}\,, \quad c = 3\,, \quad z = \tfrac{13}{16} = 0.1101_2\,, \quad g = 4\,,$$

$$g_1 = 1\,, \quad g_2 = 1\,, \quad g_3 = 0\,, \quad g_4 = 1\,,$$

$$z_1 = \tfrac{1}{2}\,, \quad z_2 = z_3 = \tfrac{3}{4} = \tfrac{6}{8}\,, \quad z_4 = \tfrac{13}{16}\,,$$

$$z_1' = 1\,, \quad z_2' = 3\,, \quad z_3' = 6\,, \quad z_4' = 13\,,$$

$$(A^{61/16} - I)(A - I)^{-1} = \tfrac{1}{16}(A^{61/16} - I)(A^{1/16} - I)^{-1}(A^{1/16} - I)\,.$$
$$.\,(A^{16/16} - I)^{-1}\,.\,16 =$$
$$= [\tfrac{1}{16}(I + A^{1/16} + \ldots + A^{60/16})]\,[\tfrac{1}{16}(I + A^{1/16} + \ldots + A^{15/16})]^{-1}\,,$$

$$Q_0 = A\,, \quad Q_1 = A^{1/2}\,, \quad Q_2 = A^{1/4}\,, \quad Q_3 = A^{1/8}\,, \quad Q_4 = A^{1/16}\,,$$

$$B_0 = I\,, \quad B_1 = \tfrac{1}{2}(I + A^{1/2})\,, \quad B_2 = \tfrac{1}{2}(I + A^{1/2})\,\tfrac{1}{2}(I + A^{1/4}) =$$
$$= \tfrac{1}{4}(I + A^{1/4} + \ldots + A^{3/4})\,, \quad B_3 = \tfrac{1}{4}(I + A^{1/4} + \ldots + A^{3/4})\,\tfrac{1}{2}(I + A^{1/8}) =$$
$$= \tfrac{1}{8}(I + A^{1/8} + \ldots + A^{7/8})\,,$$
$$B_4 = \tfrac{1}{8}(I + A^{1/8} + \ldots + A^{7/8})\,\tfrac{1}{2}(I + A^{1/16}) = \tfrac{1}{16}(I + A^{1/16} + \ldots + A^{15/16})\,,$$

$$G_0 = A^3\,, \quad G_1 = \tfrac{1}{2}A^{3+(1/2)}\,, \quad G_2 = \tfrac{1}{4}A^{3+(1/2)}\,, \quad G_3 = \tfrac{1}{8}A^{3+(1/2)+(1/4)}\,,$$

$$
\begin{aligned}
D_0 &= I + A + A^2\,, \\
D_1 &= (I + A + A^2)\,\tfrac{1}{2}(I + A^{1/2}) + \tfrac{1}{2}A^3 = \tfrac{1}{2}(I + A^{1/2} + \ldots + A^{6/2})\,, \\
D_2 &= \tfrac{1}{2}(I + A^{1/2} + \ldots + A^{6/2})\,\tfrac{1}{2}(I + A^{1/4}) + \tfrac{1}{4}A^{3+(1/2)} = \\
&= \tfrac{1}{4}(I + A^{1/4} + \ldots + A^{14/4})\,, \\
D_3 &= \tfrac{1}{4}(I + A^{1/4} + \ldots + A^{14/4})\,\tfrac{1}{2}(I + A^{1/8}) = \tfrac{1}{8}(I + A^{1/8} + \ldots + A^{29/8})\,, \\
D_4 &= \tfrac{1}{8}(I + A^{1/8} + \ldots + A^{29/8})\,\tfrac{1}{2}(I + A^{1/16}) + \tfrac{1}{16}A^{3+(1/2)+1(/4)} = \\
&= \tfrac{1}{16}(I + A^{1/16} + \ldots + A^{60/16})\,.
\end{aligned}
$$

CONCLUSION

The main idea of the algorithm is to cumulate computations in order to reach low (logarithmic) computational complexity. For r an integer, this trick is known, and is described e.g. in [2], [3] (not just for matrices). The generalization for r a non-integer is new.

Both algorithms were programmed in Fortran and tested on the IBM 370/135 computer with 4 byte floating point format (mantissa 24 bits). They work effectively and reliably for reasonable data, i.e. when $|\arg a_i| \ll \pi$ for all eigenvalues a_i of A. In the applications of the control theory, this condition means that the sampling interval is not too long. For $|\arg a_i| \to \pi$ the convergence of the matrix square root is lost. Numerical examples as well as full source programs are published in [5].

The idea of the algorithm is general and is not limited to the matrix algebra. It can be used in any algebra where inversion, square root and convergence are defined. It was e.g. implemented and tested for the algebra of real polynomials $R[x]$ modulo a fixed polynomial $p(x)$.

References

[1] *F. R. Gantmacher:* Theory of matrices (in Russian). Moscow 1966. English translation: Chelsea, New York 1966.

[2] *B. Randell, L. J. Russel:* Algol 60 Implementation. Academic Press 1964. Russian translation: Mir 1967.

[3] *D. E. Knuth:* The art of computer programming, vol. 2. Addison-Wesley 1969. Russian translation: Mir 1977.

[4] *J. Ježek:* Computation of matrix exponential, square root and logarithm (in Czech). Knižnica algoritmov, diel III, symposium Algoritmy, SVTS Bratislava 1975.

[5] *J. Ježek:* General matrix power and sum of matrix powers (in Czech). Knižnica algoritmov, diel IX, symposium Algoritmy, SVTS Bratislava 1987.

Souhrn

EFEKTIVNÍ ALGORITMUS PRO VÝPOČET REÁLNÉ MOCNINY MATICE A PŘÍBUZNÉ MATICOVÉ FUNKCE

JAN JEŽEK

Článek je věnován algoritmu pro výpočet matic A^r a $(A^r - I)(A - I)^{-1}$ pro danou čtvercovou matici A a pro reálné r. Algoritmus používá binárního rozvoje čísla r a vyznačuje se logaritmickou výpočtovou složitostí vzhledem k r. Úloha vyrůstá z potřeb teorie řízení.

Резюме

ЭФФЕКТИВНЫЙ АЛГОРИТМ ДЛЯ ВЫЧИСЛЕНИЯ ДЕЙСТВИТЕЛЬНОЙ СТЕПЕНИ МАТРИЦЫ И РОДСТВЕННОЙ МАТРИЧНОЙ ФУНКЦИИ

JAN JEŽEK

Статья посвящена алгоритму для вычисления матриц A^r и $(A^r - I)(A - I)^{-1}$ для данной квадратной матрицы A и действительного r. Алгоритм пользуется бинарным разложением числа r и отличается логарифмической вычислительной сложностью. Проблема вырастает из потребностей теории управления.

Author's address: Ing. *Jan Ježek,* CSc., ÚTIA ČSAV, Pod vodárenskou věží 4, 182 08 Praha 8.