ON THE COMPUTATION OF FUNCTIONS OF MATRICES

Herbert H. H. Homeier
Institut fuer Physikalische und Theoretische Chemie
Universitaet Regensburg
D-93040 Regensburg, Germany

June 21, 2019

Abstract

The evaluation of functions of square matrices can be based on either the Taylor series of the function, or on diagonalization techniques. In the present contribution it is shown that suitable extrapolation techniques enhance the efficiency of the Taylor series approach. As an important example, the exponential of a matrix can be obtained via this method. The exponential of matrices has to be calculated frequently in recursive methods for the solution of linear systems of ordinary differential equations, as occur in the solution of evolution equations, and also in the solution of the heat-conduction equation or the time-dependent Schrödinger equation after suitable discretization [Yung-Ya Lin and Lian-Pin Hwang, Computers Chem. **16** (1992), 285]. Several extrapolation methods will be compared. It is discussed whether these methods may also be useful for the extrapolation of vector sequences which occur for instance in iterative solutions of nonlinear equations. Examples for the latter are for instance ab initio SCF and MCSCF equations.

[1]

[1] Technical Report TC-NA-94-3, Institut fuer Physikalische und Theoretische Chemie, Universität Regensburg

Contents

List of Tables

1 Introduction

Given are a general survey of convergence problems with a special emphasis to matrix function problems, a general information on acceleration methods, and an outline of the present contribution.

1.1 Convergence Problems for Vectors and Matrices

There are a number of chemistry-related problems where slowly convergent sequences of matrices and vectors turn up:

- Matrix functions computed via Taylor series
- Extrapolation of linear polymer calculations:
  Problem: Get the Fock matrix

  $$F(k) \text{ at wave number } k \in (-\pi/a, \pi/a)$$

  of the elementary cell of length a of the infinite system

  - Cluster calculations:
    Split the Fock matrix

    $$F_n \text{ of the } (2n+1)\text{-monomer cluster into blocks } F_{n,ij}$$

    which contain Fock matrix elements between basis functions of the i-th and the j-th monomer. Extrapolate the matrix

    $$F_n(k) = \sum_{j=-n}^{n} \exp(i\, j\, k\, a)\, F_{n,0j}, \qquad a\text{: lattice constant}, \quad i^2 = -1 \tag{1}$$

    at wave number k with respect to the number of monomers

    $$F(k) = \lim_{n\to\infty} F_n(k) \tag{2}$$

  - Crystal orbital calculations:
    Denote by

    $$F^{(r)}(k)$$

    the Fock matrix at wavenumber k when Coulomb interactions between orbitals in unit cells within the distance r are taken into account. Extrapolate with respect to the range of the Coulomb interaction

    $$F(k) = \lim_{r\to\infty} F^{(r)}(k) \tag{3}$$

- Iterative solution of nonlinear equation systems:
  These can often be regarded as fixpoint problems

$$\mathbf{x} = \mathbf{F}(\mathbf{x}) \tag{4}$$

  to be solved via Picard iterations

$$\mathbf{x_0}\,, \mathbf{x_1} = \mathbf{F}(\mathbf{x_0})\,, \ldots, \mathbf{x_{n+1}} = \mathbf{F}(\mathbf{x_n})\,, \ldots\,. \tag{5}$$

  Examples are

  - SCF and MCSCF iterations
  - Geometry optimizations

The basic approach taken here is to apply acceleration methods to speed up the convergence.

1.2 Matrix Function Problems

- Solve evolution equations:
  The solution of an homogeneous, linear evolution equation

$$\frac{d}{d\,t}\mathbf{U}(t) = \mathbf{K}(t)\,\mathbf{U}(t)\,, \qquad \mathbf{U}(0) = \mathbf{U}_0 \tag{6}$$

  proceeds via a time-discretized method like

$$\mathbf{U}(t + \Delta t) \approx \exp(\mathbf{K}(t)\Delta t)\,\mathbf{U}(t) = \exp(\mathbf{A})\,\mathbf{U}(t) \tag{7}$$

  and, thus, requires computation of the matrix exponential of

$$\mathbf{A} = \mathbf{K}(t)\Delta t\,. \tag{8}$$

  Examples are the following:

  - The one-dimensional heat-conduction (= diffusion) equation (in suitable units)

$$\frac{\partial}{\partial t}u(x,t) = \frac{\partial^2}{\partial\,x^2}u(x,t) \tag{9}$$

    for

$$0 < x < 1\,, \quad t > 0\,, \quad u(0,t) = u(1,t) = 0\,, \quad u(x,0) = u_0(x) \tag{10}$$

    is solved via discretization according to

$$x_j = j\,\Delta x\,, \quad U_j(t) = u(x_j,t) \tag{11}$$

and using a central difference approximation for the second derivative. One obtains

$$\frac{d}{d\,t}U_j(t) \approx (U_{j+1}(t) - 2\,U_j(t) + U_{j-1}(t))/(\Delta x)^2 \tag{12}$$

This of the form of Eq. 6 with a tridiagonal, time-independent matrix **K**.

– Time-dependent Schrödinger equation

$$i\,\hbar\,\frac{\partial}{\partial t}\psi(t) = \hat{H}(t)\psi(t) \tag{13}$$

after introduction of a basis expansion for the wavefunction which entails a matrix representation of the Hamiltonian.

- Computation of thermodynamic density matrices:

$$\rho = \exp(-\beta\hat{H})/\mathrm{tr}(\exp(-\beta\hat{H})) \tag{14}$$

After introduction of a matrix representation of the Hamiltonian, this requires also the computation of the matrix exponential.

- Transformation to interaction representation:
This requires the computation of the matrix exponential

$$\exp(-i/\hbar\,t\,\hat{H}_0)\,, \qquad \hat{H}_0\text{: unperturbed Hamiltonian} \tag{15}$$

- Symmetrical orthogonalization:
This requires the computation of the inverse matrix square root of the overlap matrix **S**.

1.3 Convergence Acceleration Methods in General

- The problem is to find a *structure* in the data [2] which allows to implement some method to extract information on the limit of the slowly convergent sequence under study. This structural information is often hidden in some of the last digits of the data. The structural information is then used to compute the limit faster, normally via a

- Sequence transformation:

$$s_n \Longrightarrow t_n, \quad \{s_n\}_{n=0}^{\infty}\text{: original sequence,} \quad \{t_n\}_{n=0}^{\infty}\text{: transformed sequence} \tag{16}$$

The transformed sequence hopefully converges faster.

[2] A random sequence of data normally cannot be extrapolated.

- Model sequences:
  One way to find sequence transformations is to use model sequences for which the limit can be obtained exactly from a few elements of the sequence:

$$\sigma_n = \sigma + m_n(c_i, x_i) \quad \underset{\text{exact}}{\overset{T}{\Longrightarrow}} \quad \sigma = T_n^{(k)}(\sigma_n, \ldots, \sigma_{n+k}|x_i) \tag{17}$$

  c_i: coefficients
  x_i: further parameters

  The model sequences depend on some coefficients and some further parameters. Application to the problem sequence yields a *Sequence Transformation*

$$t_n = T_n^{(k)}(s_n, \ldots, s_{n+k}|x_i) \qquad \text{(approximate)} \tag{18}$$

  that accelerates problems close to the model:

$$s_n \approx \sigma_n \tag{19}$$

  If the dependence on the coefficients is linear, the model sequences span a linear space. This is called the *kernel* of the sequence transformation T: It is exact for all sequences in the kernel.

- Remainder Estimates:
  A useful more general model is to factor the model remainder into a remainder estimate and a correction factor according to

$$\begin{array}{rcl} \sigma_n = \sigma + \omega_n & \cdot \mu_n(c_i, \xi_i) \\ \uparrow & \uparrow \\ \text{remainder estimate, } \neq 0 & \text{correction factor} \end{array} \tag{20}$$

  This allows to use for instance Levin's remainder estimates [3]

$$\omega_n = \Delta s_{n-1} \text{ ; or } \omega_n = n + 1 \Delta s_{n-1} \tag{21}$$

  which can be shown to be good estimates of the remainder

$$s_n - \lim_{n\to\infty} s_n \tag{22}$$

  up to a constant for large classes of sequences.

- Iterative transformations:
  The idea is to use a simple sequence transformation iteratively

$$T_n^{(k)}(s_n) = T_n^{(0)}(T_n^{(k-1)}(s_n)) \tag{23}$$

[3] D. Levin, *Int. J. Comput. Math. B* **3** (1973) 371.

- Divergent series, summation and analytic continuation:
  There are a number of acceleration methods which can sum divergent series. These are for instance powerseries outside its radius of convergence. Another example is provided by the highly divergent perturbation series of anharmonic oscillators [4] which is a (formal) powerseries in the coupling constant. In the case of powerseries, most nonlinear sequence transformations transform the sequence of partial sums (i.e., of polynomial approximations) into a sequence of rational approximations which provide an analytic continuation of the powerseries outside its radius of convergence.

1.4 Outline

In Section 2, different approaches to evaluate functions of matrices are discussed in Section 2.1. Two important examples of such functions that are studied are the matrix exponential and the matrix square-root. The concept of pseudo inverses for vectors and matrices is introduced. Application of this concept to acceleration methods leads to the epsilon algorithms and the nonscalar $\mathcal{J}$ transformation. As a further method, we discuss the DIIS method. We present numerical results which show that acceleration methods can be applied to the evaluation of matrix functions. Finally, there is a discussion which besides the application of acceleration methods to matrix functions also addresses the question whether these methods are to be expected to be applicable to iteration sequences as well.

2 Matrix Functions

2.1 Methodology

There are two main approaches to matrix functions, i.e., to functions

$$f : \mathbf{A} \in \{N \times N matrices\} \longrightarrow \mathbf{B} = f(\mathbf{A}) \in \{N \times N matrices\} \qquad (24)$$

that are sufficiently smooth (analytic, say). These approaches are the following:

- Spectral resolution:
  If the given matrix has a spectral resolution in terms of eigenvalues and projectors to corresponding eigenspaces, then B is defined to have the same eigenspaces, and the corresponding eigenvalues are simply the given

[4] E. J. Weniger, J. Čížek, and F. Vinette, *Phys. Lett. A* **156** (1991) 169. E. J. Weniger, J. Čížek, and F. Vinette, *J. Math. Phys.* **34** (1993) 571.

function of the eigenvalue of A, i.e.,

$$\mathbf{A} = \sum_{j=1}^{k} a_j \mathbf{P}_j \Longrightarrow f(\mathbf{A}) = \sum_{j=1}^{k} f(a_j)\mathbf{P}_j \tag{25}$$

- Taylor series:
  The matrix function is given as a series of matrices according to the Taylor series expansion of the function, i.e.,

$$f(z) = \sum_{j=0}^{\infty} f_j \, z^j \Longrightarrow f(\mathbf{A}) = \sum_{j=0}^{\infty} f_j \, \mathbf{A}^j \tag{26}$$

The spectral resolution technique requires the computation of the complete spectrum of the matrix. This can be an enormous computational burden. Also, this technique is limited to matrices which possess a spectral resolution.

On the other hand, in several cases, the Taylor series is only slowly convergent and limited to its circle of convergence. Both (!) of these problems can be solved by the use of suitable convergence acceleration schemes.

2.2 Matrix Exponential

The Taylor series is

$$\exp(\mathbf{A}) = \sum_{j=0}^{\infty} \mathbf{A}^j / j! \, . \tag{27}$$

Its radius of convergence is infinite since the exponential function is analytic. The partial sums

$$\mathbf{S_n} = \sum_{j=0}^{n} \mathbf{A}^j / j! \tag{28}$$

are the input for the acceleration methods as described below.

2.3 Matrix Square-root

The Taylor series is

$$(id - \mathbf{A})^{-1/2} = \sum_{j=0}^{\infty} (1/2)_j \mathbf{A}^j / j! \, . \tag{29}$$

Here, a Pochhammer symbol

$$(a)_j = a(a+1)\cdots(a+j-1) \, , (a)_0 = 1 \tag{30}$$

has been used. The radius of convergence of the Taylor series is 1 since the nearest branchpoint of the square root has distance 1. The partial sums

$$\mathbf{S_n} = \sum_{j=0}^{\infty} (1/2)_j \mathbf{A}^j / j! \,. \tag{31}$$

are the input for the acceleration methods as described below.

2.4 Pseudo Inverses

- Motivation:
  - Scalar methods use inversion. Analog methods for vector and non-square matrices *require* pseudo inverses.
  - Scalar methods can be translated one to one to methods for square matrices. But the computation of the inverse is too expensive.
- Pseudo Inverses for Vectors:
  - Pseudo Inverse with respect to a scalar product

$$\vec{v}^{-} = \frac{\vec{v}}{\langle \vec{v} | \vec{v} \rangle} \Longrightarrow \langle \vec{v}^{-} | \vec{v} \rangle = 1 \tag{32}$$

  - Pseudo Inverse with respect to some norm $\|(.)\|$

$$\vec{v}^{I} = \frac{\vec{v}}{\|\vec{v}\|^2} \Longrightarrow \|\vec{v}^{I}\| = 1/\|\vec{v}\| \tag{33}$$

- Pseudo Inverses for Matrices:
  - Left inverse of a $m \times n$ matrix A:

$$A^L = (A^+ A)^{-1} A^+ \Longrightarrow A^L A = id_n \tag{34}$$

  - Right inverse of a $m \times n$ matrix A:.

$$A^R = A^+ (A\, A^+)^{-1} \Longrightarrow A\, A^R = id_m \tag{35}$$

  - Pseudo inverse of a $m \times n$ matrix A with respect to the trace norm

$$A^{tr} = \frac{A^+}{tr(A^+ A)} \Longrightarrow tr(A A^{tr}) = tr(A^{tr} A) = 1 \tag{36}$$

  - Pseudo inverse of a $m \times n$ matrix A with respect to a matrix norm

$$A^{No} = \frac{A^+}{\|A^+ A\|} \Longrightarrow \|A^{No} A\| = 1 \tag{37}$$

  - Pseudo Inverse by Singular Value Decomposition

3 Acceleration Methods

Many methods for scalar, vector and matrix sequences are known. [5] Here, we do not treat methods for scalar sequences. Also, only some important methods for vector and matrix sequences are given. We note that methods which are taylored for vector problems can also be applied to matrix problems if one regards each matrix as a vector where the elements of the vector are identical to the matrix elements in some arbitrary but fixed order. However, it is probable that this approach does not fully exploit the additional algebraic properties of the matrices, such that methods taylored to matrices can well be advantageous.

3.1 The DIIS method

The Direct Inversion in the Iterative Subspace [6] method was introduced for iteration sequences of vectors.

- Ansatz: The true vector of parameters as function of m members of an (iteration) sequence

$$\vec{p} = \sum_{i=1}^{m} c_i \vec{p}_i \tag{38}$$

- Calculation of the coefficients by *Least-Squares*. The residual vector

$$\Delta\vec{p} = \sum_{i=1}^{m} c_i \Delta\vec{p}_i \tag{39}$$

has to approximate zero:

$$\| \sum_{i=1}^{m} c_i \Delta\vec{p}_i \|^2 = \min , \quad \Delta\vec{p}_i = \vec{p}_{i+1} - \vec{p}_i \tag{40}$$

yields matrix equation

$$\begin{pmatrix} B_{11} & B_{12} & \dots & B_{1m} & -1 \\ B_{21} & B_{22} & \dots & B_{2m} & -1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ B_{m1} & B_{m2} & \dots & B_{mm} & -1 \\ -1 & -1 & \dots & -1 & 0 \end{pmatrix} \begin{pmatrix} c_1 \\ c_2 \\ \vdots \\ c_m \\ \lambda \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ -1 \end{pmatrix} \tag{41}$$

with

$$B_{ij} = \langle \Delta\vec{p}_i | \Delta\vec{p}_j \rangle , \quad \lambda\text{: Lagrange multiplier for } \sum c_i = 1$$

[5] See C. Brezinski und M. Redivo Zaglia, *Extrapolation methods. Theory and Practice*, North-Holland, Amsterdam, 1991.

[6] P. Pulay, Chem. Phys. Lett. 73 (1980) 393.

- Compute new estimate for solution with calculated coefficients.
- Experience: Choose $m = 6 - 8$.

3.2 Epsilon algorithms

- VECTOR EPSILON ALGORITHM:
  Wynn's [7] direct translation of the scalar epsilon algorithm via Pseudo Inverse:

$$\begin{aligned} &\vec{\epsilon}\,{}^{(n)}_{-1} = 0, \qquad \vec{\epsilon}\,{}^{(n)}_{0} = \vec{S}_n \\ &\vec{\epsilon}\,{}^{(n)}_{k+1} = \vec{\epsilon}\,{}^{(n+1)}_{k-1} + \left[\vec{\epsilon}\,{}^{(n+1)}_{k} - \vec{\epsilon}\,{}^{(n)}_{k}\right]^{-} \end{aligned} \tag{42}$$

- TOPOLOGICAL EPSILON ALGORITHM [8]:
  Choose a suitable vector $\vec{y}$ and define

$$\begin{aligned} &\vec{\epsilon}\,{}^{(n)}_{-1} = 0, \qquad \vec{\epsilon}\,{}^{(n)}_{0} = \vec{S}_n \\ &\vec{\epsilon}\,{}^{(n)}_{2k+1} = \vec{\epsilon}\,{}^{(n+1)}_{2k-1} + \frac{\vec{y}}{(\vec{y}, \vec{\epsilon}\,{}^{(n+1)}_{2k} - \vec{\epsilon}\,{}^{(n)}_{2k})} \\ &\vec{\epsilon}\,{}^{(n)}_{2k+2} = \vec{\epsilon}\,{}^{(n+1)}_{2k} + \frac{\vec{\epsilon}\,{}^{(n+1)}_{2k} - \vec{\epsilon}\,{}^{(n)}_{2k}}{(\vec{\epsilon}\,{}^{(n+1)}_{2k+1} - \vec{\epsilon}\,{}^{(n)}_{2k+1}, \vec{\epsilon}\,{}^{(n+1)}_{2k} - \vec{\epsilon}\,{}^{(n)}_{2k})} \end{aligned} \tag{43}$$

  with

$$(\vec{a}, \vec{b}) = \sum a_i b_i .$$

- For both algorithms are the

$$\vec{\epsilon}\,{}^{(n)}_{2k+1}$$

  only auxiliary quantities.

- Both algorithms are *exact* for sequences of the form

$$\vec{S}_{n+1} = A\vec{S}_n + \vec{b} \text{ with matrix } A \tag{44}$$

  or of the form

$$\sum_{i=0}^{k} a_i(\vec{S}_{n+i} - \vec{S}) = 0 \tag{45}$$

  with

$$a_k \neq 0\,, \qquad a_0 + \cdots + a_k \neq 0\,.$$

  Then one has

$$\vec{\epsilon}\,{}^{(0)}_{2k} = \vec{S}\,. \tag{46}$$

[7] P. Wynn, *Math. Comput.* **16** (1962) 301.
[8] C. Brezinski, *Calcolo* **12** (1975) 205.

- Both algorithms can be used to accelerate Conjugate-Gradient methods. [9]

3.3 Nonscalar $\mathcal{J}$ Transformation

The $\mathcal{J}$ transformation [10] :
is based on the idea to use simple transformations with kernel

$$S_n = S + \Omega_n(c_0 + c_1 R_n) \tag{47}$$

iteratively and to choose the remainder estimates for each stage of the iteration differently. This has been studied in great detail in the technical report TC-NA-94-1. [11]

- A variant with matrix inverse for $N \times N$ matrices

  $$S.., \Omega.., R..$$

  is given by

  $$\begin{aligned} &S_n^{(0)} = S_n; \qquad \Omega_n^{(0)} = \Omega_n; \\ &S_n^{(k+1)} = S_n^{(k)} - \Omega_n^{(k)}[\Delta\Omega_n^{(k)}]^{-1}\Delta S_n^{(k)}; \\ &\Omega_n^{(k+1)} = -\Omega_n^{(k)}[\Delta\Omega_n^{(k)}]^{-1}\Omega_{n+1}^{(k)}\Delta R_n^{(k)}. \end{aligned} \tag{48}$$

- A variant with left inverses for

  $$M \times N \text{ matrices } S.., M \times K \text{ matrices } \Omega_n \text{ and numbers } R..$$

  is given by

  $$\begin{aligned} &S_n^{(0)} = S_n; \qquad \Omega_n^{(0)} = \Omega_n; \\ &S_n^{(k+1)} = S_n^{(k)} - \Omega_n^{(k)}([\Delta\Omega_n^{(k)}]^L\Delta S_n^{(k)}); \\ &\Omega_n^{(k+1)} = -\Omega_n^{(k)}([\Delta\Omega_n^{(k)}]^L\Omega_{n+1}^{(k)})\Delta R_n^{(k)}. \end{aligned} \tag{49}$$

- A variant with pseudoinverse with respect to the trace norm for

  $$M \times N \text{ matrices } S.., \Omega.. \text{ and numbers } R..$$

  is given by

  $$\begin{aligned} &S_n^{(0)} = S_n; \qquad \Omega_n^{(0)} = \Omega_n; \\ &S_n^{(k+1)} = S_{n+1}^{(k)} - \Omega_{n+1}^{(k)} tr([\Delta\Omega_n^{(k)}]^{tr}\Delta S_n^{(k)}); \\ &\Omega_n^{(k+1)} = -\Omega_{n+1}^{(k)} tr([\Delta\Omega_n^{(k)}]^{tr}\Omega_n^{(k)})\Delta R_n^{(k)}. \end{aligned} \tag{50}$$

  This variant with

  $$R_n^{(k)} = 1/(n+1) \text{ and } \Omega_n = \Delta S_n$$

  is used exclusively in the following.

---

[9] N. Aboun, Thèse 3ème cycle, Université de Paris VI, 1985. N. Rahmani-Gasmi, Thèse 3ème cycle, Université de Paris VI, 1985.

[10] H. H. H. Homeier, Int. J. Quantum Chem. 45 (1993) 545.

[11] H. H. H. Homeier, Numer. Algo., in press.

4 Test Results

Treated is the matrix exponential for the matrices

$$A = \begin{pmatrix} -0.001 & -0.2 & -0.3 \\ -0.4 & -0.5 & -0.6 \\ -0.7 & -0.8 & -0.9 \end{pmatrix} \tag{51}$$

$$B = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 0.1 & 0.1 & 0.1 & 0.1 & 0.1 \\ -5 & -5 & -5 & -5 & -5 \\ 0.3 & 0.3 & 0.3 & 0.3 & 0.3 \\ 0.01 & 0.01 & 0.01 & 0.01 & 0.01 \end{pmatrix} \tag{52}$$

$$C = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} \tag{53}$$

Plotted as table entries are the number of exact digits (i.e., the negative decadic logarithm of the error norm) for each method. The accuracy of term-by-term summation is compared to acceleration results.

Table 1: Matrix Exponential: Definitions

N	:	Total number of sequence elements used
n	:	Number of applications of the acceleration method
A_N	:	$\sum_{j=0}^{N-1} \frac{A^j}{j!}$
B_N	:	$\sum_{j=0}^{N-1} \frac{B^j}{j!}$
C_N	:	$\sum_{j=0}^{N-1} \frac{C^j}{j!}$

In Table 1, the definitions of various quantities occurring in the other tables are collected. In particular, it should be noted that the application of the acceleration methods was not always started with the first partial sum that was available, but instead a later partial sum was regarded as the 0th element of the sequence to be transformed. In such a way one may study the influence of the first few terms of the sequence which may be not well described by the remainder estimates. The latter are asymptotic in character.

In Table 2, the DIIS method is used as acceleration method. The data show that it is a moderately efficient method, but only if two criteria are met: It has

Table 2: Matrix Exponential: DIIS Method

N=n+3	A_N	DIIS
4	0.9	0.9
5	1.4	0.9
6	1.9	2.8
7	2.5	3.4
8	3.2	3.6
9	4.0	4.1
10	4.8	3.6
11	5.6	4.0
12	6.5	4.2

N=n+10	A_N	DIIS
11	5.6	5.6
12	6.5	5.6
13	7.4	8.6
14	8.3	8.5
15	9.3	10.2
16	10.3	9.3
17	11.3	10.0
18	12.4	11.3
19	13.4	12.4

N=n+8	B_N	DIIS
9	0.7	0.7
10	1.2	0.7
11	1.7	2.9
12	2.2	2.4
13	2.7	1.7
14	3.3	2.7
15	3.9	3.5
16	4.6	2.4
17	5.2	3.1

N=n+16	B_N	DIIS
16	4.6	2.9
17	5.2	5.2
18	5.9	5.2
19	6.7	8.1
20	7.4	8.8
21	8.2	7.6
22	9.0	8.4
23	9.8	8.1
24	10.6	9.0

N=n+30	C_N	DIIS
31	3.2	3.2
32	3.5	3.2
33	3.8	4.8
34	4.2	5.2
35	4.5	5.4
36	4.9	5.5
37	5.2	6.1
38	5.6	6.0
39	6.0	5.9

N=n+40	C_N	DIIS
41	6.8	6.8
42	7.2	6.8
43	7.7	8.9
44	8.1	8.9
45	8.6	9.6
46	9.0	8.7
47	9.5	9.4
48	10.0	9.6
49	10.5	9.1

Table 3: Matrix Exponential: Epsilon Algorithms

N=n+0	A_N	Top. ϵ
1	0.3	0.3
3	0.5	1.2
5	1.4	2.7
7	2.5	4.3
9	4.0	6.8
11	5.6	8.0
13	7.4	10.9
15	9.3	12.7
17	11.3	15.2

N=n+1	A_N	Vector ϵ
3	0.5	0.5
5	1.4	2.2
7	2.5	4.1
9	4.0	6.2
11	5.6	8.4
13	7.4	10.7

N=n+0	B_N	Top. ϵ
3	-0.5	0.6
5	-0.4	1.4
7	0.0	2.5
9	0.7	3.9
11	1.7	5.4
13	2.7	7.0
15	3.9	8.8
17	5.2	10.7
19	6.7	12.8
21	8.2	14.8

N=n+0	B_N	Vector ϵ
1	0.1	0.1
3	-0.5	0.6
5	-0.4	1.4
7	0.0	2.5
9	0.7	3.9
11	1.7	5.4
13	2.7	7.0
15	3.9	8.8
17	5.2	10.7

N=n+0	C_N	Top. ϵ	Vector ϵ
31	3.2	3.2	4.1
33	3.8	6.4	5.4
35	4.5	6.1	6.3
37	5.2	6.9	7.3
39	6.0	8.6	8.7
41	6.8	9.4	—
49	10.5	13.2	—
51	11.5	14.2	—
53	12.5	13.8	—

N=n+30	C_N	Top. ϵ	Vector ϵ
33	3.8	4.8	4.8
35	4.5	6.2	6.2
37	5.2	7.7	7.7
39	6.0	9.1	9.1
41	6.8	10.6	10.6
43	7.7	12.1	—
45	8.6	13.8	—
47	9.5	13.9	—
49	10.5	13.8	—
51	11.5	13.8	—

Table 4: Matrix Exponential: Vector J Transformation

N=n+0	A_N	Vector $\mathcal{J}$
1	0.3	0.3
3	0.5	2.1
5	1.4	4.3
7	2.5	6.9
9	4.0	9.7
11	5.6	12.6
13	7.4	15.5
15	9.3	15.1
17	11.3	15.2
19	13.4	15.2

N=n+0	B_N	Vector $\mathcal{J}$
1	0.1	0.1
3	-0.5	1.0
5	-0.4	2.4
7	0.0	4.1
9	0.7	6.2
11	1.7	8.5
13	2.7	11.0
15	3.9	13.6
17	5.2	15.3
19	6.7	15.4

N=n+0	C_N	Vector $\mathcal{J}$
11	0.0	-0.6
13	0.1	1.2
15	0.2	3.4
17	0.4	4.8
19	0.6	6.7
21	0.9	8.7
23	1.2	10.6
25	1.6	13.0
27	2.1	13.1
29	2.6	12.6

N=n+30	C_N	Vector $\mathcal{J}$
31	3.2	3.2
33	3.8	5.3
35	4.5	6.5
37	5.2	7.9
39	6.0	9.5
41	6.8	11.3
43	7.7	13.1
45	8.6	15.0
47	9.5	15.3
49	10.5	15.2

to be applied close to the domain where the series itself is already starting to converge more rapidly, and no more than a few terms of the sequence should be used. Otherwise, it can well be that DIIS-transformed sequence converges slower than the problem sequence.

From the data presented in Table 3 it is clear that the nonscalar epsilon algorithms are more powerful than the DIIS method for the cases studied. The topological and the vector epsilon algorithm perform rather similarly. However, with the vector epsilon algorithm it is some times not possible to go to higher values of n due to program restrictions. This is indicated by a hyphen in the tables.

The data presented in Table 4 indicate in comparison with those of Tables 2 and 3 that for the cases studied the vector version of the $\mathcal{J}$ transformation is the most successful acceleration method.

5 Discussion

The results indicate that nonlinear convergence acceleration enhances the flexibility and efficiency of the Taylor series approach to the evaluation of matrix functions.

As regards the acceleration of fixed point iterations, some work still has to be done. However, it should be noted that there is a theorem due to ???? that the vector epsilon algorithm gives quadratic convergence [12] near a fixed point if 1 is not an eigenvalue of the Jacobian of the iteration function. This result indicates that nonlinear convergence acceleration methods might provide a new way for solving convergence problems in fixed point iterations which are not untypical for chemical problems as discussed in the introduction.

[12] Without any derivative evaluations!