Memorandum 1992 (September 2012). ISSN 1874-4850. Available from: http://www.math.utwente.nl/publications
Department of Applied Mathematics, University of Twente, Enschede, The Netherlands

A short guide to exponential Krylov subspace time integration for Maxwell's equations[☆]

Mike A. Botchev

Department of Applied Mathematics and MESA+ Institute for Nanotechnology, University of Twente, P.O. Box 217, NL-7500 AE Enschede, the Netherlands, mbotchev@na-net.ornl.gov.

Abstract

The exponential time integration, i.e., time integration which involves the matrix exponential, is an attractive tool for solving Maxwell's equations in time. However, its application in practice often requires a substantial knowledge of numerical linear algebra algorithms, in particular, of the Krylov subspace methods. This note provides a brief guide on how to apply exponential Krylov subspace time integration in practice. Although we consider Maxwell's equations, the guide can readily be used for other similar time-dependent problems. In particular, we discuss in detail the Arnoldi shift-and-invert method combined with recently introduced residual-based stopping criterion.

Two of the algorithms described here are available as MATLAB codes and can be downloaded from the website `http://eprints.eemcs.utwente.nl/` together with this note.

Keywords: matrix exponential, Maxwell's equations, Krylov subspace methods, exponential time integration, shift-and-invert, stopping criterion
2008 MSC: 65F60, 65F30, 65N22, 65L05, 35Q61

1. Introduction

Efficient numerical solution of Maxwell's equations, and their time integration in particular, remains a challenging task. This is due to both an increasing complexity of the electromagnetic models and a frequent need

[☆]This work was partially supported by Russian federal program "Scientific and scientific-pedagogical personnel of innovative Russia".

to couple them with other models relevant for the process under consideration. An important aspect in the numerical solution of Maxwell's equations is their space–time field geometric structure: to be successful a numerical solution procedure has to respect it. Examples of such mimetic space discretization schemes for Maxwell's equations are the well-known Yee cell [69] and the edge–face Whitney–Nédélec vector finite elements [48, 49, 6, 46].

Once a proper space discretization is applied, one is left with a large system of ordinary differential equations (ODEs) to integrate in time. On one hand, a good time integration scheme has to accurately resolve the wave structure of the equations. On the other hand, it has to cope with possible time step restrictions due to, for instance, a locally refined space grid or large conduction terms. The exponential time integration schemes seem to be especially well suited in this context, as they combine excellent (unconditional) stability properties with ability to produce a very accurate solution even for relatively large time step sizes.

The aim of this note is to provide a short introduction to the exponential time integration methods based on the Krylov subspace methods for Maxwell's equations. Within the electromagnetic engineering community, exponential methods seem to slowly but surely gain acceptance and popularity over the last years [17, 43, 11, 59, 9].

Nevertheless, there are some recent developments in the Krylov subspace methods which significantly increase the efficiency of the exponential time integration but seem to be up to now overlooked by physicists and engineers. Such developments are the shift-and-invert (SAI) techniques, restarting and residual-based stopping criteria [64, 24, 29, 7, 8]. For instance, we are unaware of any single electromagnetics paper where the Krylov SAI exponential methods are employed to solve Maxwell's equations. This note aims at filling this gap by providing a relatively self-contained guide for some of these methods.

We consider Maxwell's equations in the form

$$\begin{aligned} \mu \partial_t \boldsymbol{H} &= -\nabla \times \boldsymbol{E}, \\ \varepsilon \partial_t \boldsymbol{E} &= \nabla \times \boldsymbol{H} - \sigma \boldsymbol{E} + \boldsymbol{J}, \end{aligned} \tag{1}$$

where $\boldsymbol{H} = \boldsymbol{H}(x, y, z, t)$ and $\boldsymbol{E} = \boldsymbol{E}(x, y, z, t)$ are vector functions denoting respectively magnetic and electric fields, $\mu = \mu(x, y, z)$ is magnetic permeability, $\varepsilon = \varepsilon(x, y, z)$ is electric permittivity, $\sigma = \sigma(x, y, z)$ is electric conduction and $\boldsymbol{J} = \boldsymbol{J}(x, y, z, t)$ is the electric current. We assume that the space variables (x, y, z) vary within a domain $\Omega \subset \mathbb{R}^3$ and that suitable initial and boundary conditions are set for system (1).

A chosen finite difference or (vector) finite element space discretization then yields an ODE system

$$\begin{aligned} M_\mu h' &= -Ke + j_h, \\ M_\varepsilon e' &= K^T h - M_\sigma e + j_e, \end{aligned} \tag{2}$$

where $h = h(t) : \mathbb{R} \to \mathbb{R}^{n_h}$ and $e = e(t) : \mathbb{R} \to \mathbb{R}^{n_e}$ are vector functions, whose components are time-dependent degrees of freedom associated with the magnetic and electric fields, respectively. Furthermore, $M_\mu \in \mathbb{R}^{n_h \times n_h}$ and $M_\varepsilon, M_\sigma \in \mathbb{R}^{n_e \times n_e}$ are the mass matrices and $K \in \mathbb{R}^{n_h \times n_e}$ is the discretized curl operator. Finally, $j_h = j_h(t) : \mathbb{R} \to \mathbb{R}^{n_h}$, $j_e = j_e(t) : \mathbb{R} \to \mathbb{R}^{n_e}$ are the source functions containing contributions of the discretized current function $\boldsymbol{J}$ and possibly also from the discretized boundary conditions. If the standard Yee cell space discretization is used then n_h is the total number of the cell faces in the space grid, where the discrete values of $\boldsymbol{H}$ are defined, and n_e is the total number of the edges in the space grid, where the values of $\boldsymbol{E}$ are defined. In this case M_μ, M_ε and M_σ are simply diagonal matrices containing the grid values of μ, ε and σ, respectively. For details on how an edge–face vector finite element discretization leads the ODE system of type (2) see e.g. [54, 10].

Putting $h(t)$ and $e(t)$ into one vector function $y = y(t) : \mathbb{R} \to \mathbb{R}^n$, $n = n_h + n_e$, we can cast (2) into an equivalent form

$$y'(t) = -Ay(t) + g(t), \tag{3}$$

with

$$y(t) = \begin{bmatrix} h(t) \\ e(t) \end{bmatrix}, \quad A = \begin{bmatrix} 0 & M_\mu^{-1} K \\ -M_\varepsilon^{-1} K^T & M_\varepsilon^{-1} M_\sigma \end{bmatrix}, \quad g(t) = \begin{bmatrix} M_\mu^{-1} j_h(t) \\ M_\varepsilon^{-1} j_e(t) \end{bmatrix}.$$

Here, unless the mass matrices are diagonal, one does not need to compute their inverses. The algorithms considered in this paper avoid the explicit use of the inverse mass matrices. However, we need to compute the action of the inverses on given vectors. This can be done by computing a sparse Cholesky factorization for each of the mass matrices once and applying it every time the action of the inverse is needed. If the problem is too big, so that the sparse Cholesky is not possible, then the action of the inverses can be computed by solving the linear systems with the mass matrix by a preconditioned conjugate gradient (PCG) method, see e.g. [3, 66, 56].

If ε and σ are constant[1] then the eigenvalues of A can be determined

[1] More exactly, it is sufficient to assume that ε and σ are such that $M_\varepsilon^{-1} M_\sigma = \alpha I$, with $\alpha \in \mathbb{R}$ and I the identity matrix.

analytically. In this case system (2) can be uncoupled by an orthogonal transformation into a set of harmonic oscillators, see [10, Section 2.1].

The paper is organized as follows. In the next section some basic exponential time integration methods for Maxwell's equations are briefly discussed. Section 3 provides a short introduction to Krylov subspace methods for the matrix exponential and presents in detail a simple algorithm of this class. In Section 4 we discuss the Krylov subspace Arnoldi/SAI method and its implementation. A numerical example coming from the field of electromagnetic imaging is presented in Section 5 and conclusions are drawn in the last section. Throughout the paper, unless indicated otherwise, the norm sign $\|\cdot\|$ denotes the standard Euclidean vector 2-norm or the corresponding operator (matrix) norm.

2. Exponential time integration

A standard second order method to integrate (2) reads

$$\begin{aligned} M_\mu \frac{h_{k+1/2} - h_{k-1/2}}{\tau} &= -Ke_k + (j_h)_k, \\ M_\varepsilon \frac{e_{k+1} - e_k}{\tau} &= K^T h_{k+1/2} - \frac{1}{2} M_\sigma (e_{k+1} + e_k) + \frac{1}{2}\big((j_e)_k + (j_e)_{k+1}\big), \end{aligned} \tag{4}$$

where τ is the time step size and the subscript k refers to the time level, e.g. $(j_e)_k = j_e(k\tau)$. We refer to this method as CO2 (composition order 2) method because it can be seen as a symplectic second order composition [10] often used in the geometric time integration [30]. The method employs the explicit staggered leapfrog time stepping for the curl terms $-Ke_k$ and $K^T h_{k+1/2}$. It coincides with the classical Yee scheme for the Yee cell finite difference space discretization and $\sigma \equiv 0$. The conductivity and the source terms, i.e., $-\frac{1}{2}M_\sigma(e_{k+1}+e_k)+\frac{1}{2}((j_e)_k+(j_e)_{k+1})$, are treated by the implicit trapezoidal rule (ITR), also known as the Crank–Nicolson scheme. The scheme is thus implicit–explicit. Note that (4) can be rewritten as

$$\begin{aligned} M_\mu \frac{h_{k+1/2} - h_k}{\tau/2} &= -Ke_k + (j_h)_k, \\ M_\varepsilon \frac{e_{k+1} - e_k}{\tau} &= K^T h_{k+1/2} - \frac{1}{2} M_\sigma (e_{k+1} + e_k) + \frac{1}{2}((j_e)_k + (j_e)_{k+1}), \\ M_\mu \frac{h_{k+1} - h_{k+1/2}}{\tau/2} &= -Ke_{k+1} + (j_h)_k. \end{aligned} \tag{5}$$

The equivalence with (4) can be seen by combining the first formula above with the third formula for the previous step $k-1$. The form of CO2 given by (5) is actually the form the CO2 scheme is derived by the composition approach [10], and it can also be used in practical computations at the first and last time steps. In the vector finite element context, paper [54] discusses a closely related leapfrog scheme which differs from CO2 (4) in the way the source term is treated.

The CO2 scheme is second accurate and conditionally stable with the sufficient stability condition

$$\tau \leqslant \frac{2}{\sqrt{\max \psi}},$$

where ψ denote the eigenvalues of the matrix $M_\varepsilon^{-1} K^T M_\mu^{-1} K$. This matrix can be shown to be positive semidefinite [10], which justifies the expression $\sqrt{\max \psi}$. If $\sigma \equiv 0$ then the inequality in the stability condition above has to be strict to provide stability. Implementation of CO2 involves solution of a linear system with the symmetric positive definite matrix $M_\varepsilon + \frac{\tau}{2} M_\sigma$. The solution can be efficiently carried out by a sparse direct Cholesky solver or by PCG, see e.g. [3, 66, 56].

Another scheme which can be used for time integration of Maxwell's equations is the ITR or Crank-Nicolson scheme. When applied to (3), it reads

$$\frac{y_{k+1} - y_k}{\tau} = -\frac{1}{2} A (y_k + y_{k+1}) + \frac{1}{2} (g_k + g_{k+1}). \tag{6}$$

The scheme is second order accurate and unconditionally stable. At each time step a linear system with the matrix $I + \frac{\tau}{2} A$ has to be solved. This is a computationally expensive task, much more expensive than solving a linear system with a mass matrix (as e.g. in the CO2 scheme) [67]. The matrix $I + \frac{\tau}{2} A$ is (strongly) nonsymmetric and its sparsity structure is much less favorable for a sparse direct solver than it is in a mass matrix. Moreover, standard preconditioners may not be efficient and a choice of a proper preconditioned iterative solver is far from trivial, see [67] for a discussion and numerical results in a vector finite element setting.

On the other hand, even if an efficient direct sparse or preconditioned iterative solution of the ITR linear system is possible, it is often advisable to use another type of scheme, namely, an exponential time integration scheme (see also Section 5 in this note).

To solve the linear system with the matrix $I + \frac{\tau}{2} A$ in the context of a

finite element discretization, we can rewrite the matrix as

$$\begin{aligned} I + \frac{\tau}{2}A = I + \frac{\tau}{2}M_*^{-1}A_* = M_*^{-1}(M_* + \frac{\tau}{2}A_*), \\ \text{with} \quad A_* = \begin{bmatrix} 0 & K \\ -K^T & M_\sigma \end{bmatrix}, \quad M_* = \begin{bmatrix} M_\mu & 0 \\ 0 & M_\varepsilon \end{bmatrix}. \end{aligned} \tag{7}$$

Thus, one does not need to form the inverse mass matrices explicitly.

For a more detailed discussion of regular time integration methods for Maxwell's equations see e.g. [38]. We now give a short discussion of the exponential time integration. A complete review on this subject can hardly be given in this brief note, as it is an active field of research [36, 4, 5, 44, 51]. Instead, here we only give some basic ideas. First, note that the solution of the initial value problem (IVP) with $A \in \mathbb{R}^{n\times n}$

$$y'(t) = -Ay(t), \quad y(0) = v, \quad t \geqslant 0, \tag{8}$$

can be written as

$$y(t) = \exp(-tA)v, \quad t \geqslant 0, \tag{9}$$

where $\exp(-tA) \in \mathbb{R}^{n\times n}$ is a matrix called the matrix exponential [26, 27, 45, 32, 23, 33]. Note also that to compute $y(t)$ for several specific values of t with (9) we only need to compute the *action* of the matrix exponential on the vector v, note the matrix exponential itself. In the next section, we discuss how this action on a vector can be computed.

Consider an IVP with a constant inhomogeneous term $g_0 \in \mathbb{R}$

$$y'(t) = -Ay(t) + g_0, \quad y(0) = v, \quad t \geqslant 0. \tag{10}$$

Assuming for the moment that A is invertible, we can rewrite the ODE system $y'(t) = -Ay(t) + g_0$ as

$$(y(t) - A^{-1}g_0)' = -A((y(t) - A^{-1}g_0),$$

which is a homogeneous ODE of the form (8). Hence, its solution satisfying initial condition $y(0) = v$ reads

$$y(t) - A^{-1}g_0 = \exp(-tA)(v - A^{-1}g_0). \tag{11}$$

Let us now introduce functions φ_j, $j = 0, 1, \ldots$, which are extensively used in exponential time integration:

$$\varphi_0(x) = \exp(x), \quad \varphi_j(x) \quad = \frac{\varphi_{j-1}(x) - \varphi_{j-1}(0)}{x}, \quad j = 1, 2, \ldots .$$

Note that $\varphi_j(0) = 1/j!$ and that, in particular,

$$\varphi_1(x) = \frac{\exp(x) - 1}{x} = 1 + \frac{x}{2!} + \frac{x^2}{3!} + \frac{x^3}{4!} + \dots .$$

The expression (11) can be modified as

$$\begin{aligned} y(t) &= \exp(-tA)v + t\varphi_1(-tA)g_0 \\ &= v + t\varphi_1(-tA)(-Av + g_0), \qquad t \geqslant 0. \end{aligned} \tag{12}$$

Since the functions $\varphi_j(x)$ are smooth for all $x \in \mathbb{C}$, the last relations hold for any, not necessarily nonsingular A. The first expression for $y(t)$ in (12) is instructive as it shows the effect of the inhomogeneous constant term on the solution (cf. (9)), whereas the second one can be preferably used in computations. Indeed, the second formula requires evaluation of just a single matrix function times a vector and this can be computed similarly to the evaluation of the matrix exponential (see the next section).

For ODE system (3) with general, non-constant source term $g(t)$, its solution satisfying initial condition $y(0) = v$ can be expressed with the help of the so-called variation-of-constants formula:

$$y(t) = \exp(-tA)v + \int_0^t \exp\big(-(t-s)A\big)g(s)\mathrm{d}s, \qquad t \geqslant 0. \tag{13}$$

This formula is a backbone for exponential time integration, it often serves as a starting point for derivation of exponential integrators. Note that, as soon as $g(t) = g_0 = \mathrm{const}(t)$, (12) can be obtained directly from (13) by evaluating the integral term. Similarly, if $g(t)$ is assumed to be constant for $t \in [t_k, t_k + \tau]$, i.e, $g(t) = g_k$, we can write

$$y_{k+1} = \exp(-\tau A)y_k + \tau\varphi_1(-\tau A)g_k = y_k + \tau\varphi_1(-\tau A)(-Ay_k + g_k), \tag{14}$$

which is known as the exponentially fitted Euler scheme, see e.g. [35]. The method is first order accurate when applied to (3) with general, time dependent $g(t)$ and exact for any $\tau \geqslant 0$ as soon as $g = \mathrm{const}(t)$. The method is unconditionally stable in the sense of A-stability[2].

[2] A-stability of a method means that the method applied to the so-called Dahlquist scalar test problem $y' = \lambda y$ yields a bounded solution for any time step size as soon as $\lambda \in \mathbb{C}$ has a negative real part. All exponential methods considered in this note are exact for this test problem and, thus, are A-stable.

We now give an example of second-order accurate exponential time integrator, called EK2, exponential Krylov scheme of order 2, see [67]:

$$y_{k+1} = y_k + \tau(-Ay_k + g_k) + \tau\varphi_2(-\tau A)\big(-\tau A(-Ay_k + g_k) + g_{k+1} - g_k\big). \quad (15)$$

This method, which goes back to [13, 42], can be derived by interpolating the function g under the integral in (13) linearly and evaluating the integral. This approach is sometimes refered to as exponential time differencing [14, 53] and is closely related to a class of exponential Runge–Kutta–Rosenbrock methods [37].

It should be emphasized that, as we see, exponential time integrators considered in this section can be applied in two essentially different settings. In the first one, just a few actions of matrix functions are required to compute solution at any time t of interest, $t \in [0, T]$. This is possible when the source term is zero or (almost) constant, cf. (9),(12). In the second setting, we have a time stepping procedure, where actions of matrix functions are required every time step, see (14),(15). In our limited experience, exponential time integrators are computationally efficient for Maxwell's equations primarily in the first setting. For instance, the CO2 scheme appears to be much more efficient than EK2 in the experiments from [67], eventhough some promising results with exponential integration are reported in [9]. A possible approach to reduce the number of matrix function actions and, hence, to increase efficiency of the exponential integration is presented in [8].

We now describe a way to compute $\exp(-tA)v$ or $\varphi_k(-tA)v$ for a given vector $v \in \mathbb{R}^n$.

3. Computing the matrix exponential action by Krylov subspaces

There are several ways to compute the action of the matrix exponential $\exp(-tA)$ (or the related matrix functions $\varphi_k(-tA)$) of a large matrix A on a given vector v. These methods include Krylov subspace methods [65, 19, 40, 25, 55, 20, 34, 35, 21], the Chebyshev polynomials, scaling and squaring with Padé or Taylor approximations and other methods [62, 17, 58, 12, 2]. Here, we describe only one group of the methods, namely, the Krylov subspace methods. We choose to restrict ourselves to these methods because they seem to combine versatility and efficiency. The Chebyshev polynomials are mostly used for computing the matrix functions of symmetric or skew-symmetric matrices, whereas the matrix A from (3) is in general neither of both. Computing matrix functions with the Chebyshev polynomials for general matrices is possible but not trivial [41]. For $\sigma \equiv 0$ the Maxwell

matrix A can be transformed into a skew-symmetric matrix and, hence, its exponential can be computed via the Chebyshev polynomials, see e.g. [17].

3.1. Computing action of $\exp(-tA)$

In Krylov subspace methods an orthogonal basis $\{v_1, \ldots, v_m\}$ of the Krylov subspace

$$\mathcal{K}_m(A, v) = \text{span}\{v, Av, A^2v, \ldots, A^{m-1}v\}$$

is built and stored as the columns of a matrix $V_m = [v_1, \ldots, v_m] \in \mathbb{R}^{n\times m}$. The matrix V_m is usually computed with the help of the Arnoldi process (outlined below in Figure 1) and satisfies the so-called Arnoldi decomposition [66, 56]

$$AV_m = V_{m+1}H_{m+1,m}, \qquad H_{m+1,m} \in \mathbb{R}^{m+1,m}. \tag{16}$$

The matrix $H_{m+1,m}$ is upper-Hessenberg, which means that its entries $h_{i,j}$ are zero as soon as $i > j + 1$. Denoting by $H_{m,m}$ the matrix composed of the first m rows of $H_{m+1,m}$, we can rewrite (16) as

$$AV_m = V_m H_{m,m} + v_{m+1}h_{m+1,m}e_m^T, \tag{17}$$

where $e_m = [0, \ldots 0,\ 1]^T \in \mathbb{R}^m$ is the last canonical basis vector in $\mathbb{R}^m$. The last relation basically says that A times any vector of the Krylov subspace is again a vector from the same subspace plus a multiple of the next Krylov basis vector v_{m+1}. Krylov subspace methods are usually successful if this last term $v_{m+1}h_{m+1,m}e_m^T$ turns out to be, for some m, small. This means that the Krylov subspace is close to an invariant subspace of A.

To compute $y(t) = \exp(-tA)v$ for a given $v \in \mathbb{R}^n$, we set the first Krylov basis vector v_1 to be the normalized vector v ($\beta := \|v\|$, $v_1 := v/\beta$), and, once V_m and $H_{m,m}$ are computed, obtain an approximation $y_m(t)$ to $y(t)$ as

$$\begin{aligned} y(t) &= \exp(-tA)v = \exp(-tA)(V_m\beta e_1) \\ &\approx y_m(t) = V_m \underbrace{\exp(-tH_{m,m})\beta e_1}_{u_m(t)}. \end{aligned} \tag{18}$$

Here $e_1 = [1\,, 0, \ldots 0]^T \in \mathbb{R}^m$ is the first canonical basis vector in $\mathbb{R}^m$. The rational behind (18) is that if A times a Krylov subspace vector is approximately again a Krylov subspace vector, then so is $\exp(-tA)$ times a Krylov subspace vector. This is true because $\exp(-tA)$, as any matrix function of A, is a polynomial in A. Computing $y_m(t)$ in (18) is much cheaper than

computing $y(t)$ because we hope to have $m \ll n$ and usually m does not exceed several hundreds. The exponential of $-tH_{m,m}$ then can be computed by any suitable scheme for matrices of a moderate size, see e.g. [32, Chapter 10] and [61, 4]. Note that software packages MATLAB and Mathematica use the scaling and squaring algorithm of [31].

An algorithm for computing $y(t) = \exp(-tA)v$ for a given $v \in \mathbb{R}^n$ by the just described Krylov subspace method is shown in Figure 1. The algorithm is based on the Arnoldi process and uses the fact that the Krylov subspace is scaling invariant, i.e., tA and A produce the same Krylov basis matrix V_m. The essential parts of the algorithm are the orthogonalization of the newly computed Krylov basis vector w with the modified Gram-Schmidt process (lines 16–19), computation of the vector $u_m(t)$ from (18) and a residual-based stopping criterion (lines 22–28) which we adopt from [18, 39, 7].

The stopping criterion is based on controlling the residual of the approximation $y_m(t)$ from (18) with respect to the ODE (8), i.e., the exponential residual is defined as

$$r_m(t) \equiv -Ay_m(t) - y'_m(t). \tag{19}$$

Indeed, with (17) and (18) it is not difficult to see that [18, 7]

$$r_m(t) = -(h_{m+1,m}e_m^T u_m(t))v_{m+1}, \quad \|r_m(t)\| = |h_{m+1,m}e_m^T u_m(t)|. \tag{20}$$

The value `resnorm` computed by the algorithm is the residual norm relative to the norm β of the initial vector v,

$$\texttt{resnorm} = \frac{\|r_m(t)\|}{\beta}.$$

Since $r_m(t)$ is a time dependent function it is possible that $\|r_m(t)\| \approx 0$ occasionally at some specific points t only. Ideally, one might want to compute the L_2 norm $\int_0^t \|r_m(s)\|^2 \mathrm{d}s$. In practice it seems sufficient to compute the residual norm at several time moments $s \in (0, t]$, this is done in the lines 23–27. For more detail and discussion on the relation to other possible stopping criteria we refer to [7].

At line 25 of the algorithm at Figure 1, the exponential of $\exp(-tH_{m,m})$ is computed with a function `expm`, a built in function available in both Octave and MATLAB. Both the Octave and MATLAB implementations of `expm`, based respectively on papers [68] and [31], are the scaling and squaring algorithms combined with Padé approximations. We should emphasize that the costs for computing $\exp(-tH_{m,m})$ are usually negligible with respect

```
function y = expm_Arnoldi(A,v,t,toler,m)
% y = expm_Arnoldi(A,v,t,toler,m)
% computes $y = \exp(-t A) v$
% input: A (n x n)-matrix, v n-vector, t>0 time interval,
%        toler>0 tolerance, m maximal Krylov dimension

n = size (v,1);
V = zeros(n  ,m+1);
H = zeros(m+1,m);

beta   = norm(v);
V(:,1) = v/beta;

for j=1:m
    w = A*V(:,j);
    for i=1:j
        H(i,j) = w'*V(:,i);
        w      = w - H(i,j)*V(:,i);
    end
    H(j+1,j) = norm(w);
    e1 = zeros(j,1); e1(1) = 1;
    ej = zeros(j,1); ej(j) = 1;
    s  = [0.01, 1/3, 2/3, 1]*t;
    for q=1:length(s)
        u         = expm(-s(q)*H(1:j,1:j))*e1;
        beta_j(q) = -H(j+1,j)* (ej'*u);
    end
    resnorm = norm(beta_j,'inf');
    fprintf('j = %d, resnorm = %.2e\n',j,resnorm);
    if resnorm<=toler
       break
    elseif j==m
       disp('warning: no convergence within m steps');
    end
    V(:,j+1) = w/H(j+1,j);
end
y = V(:,1:j)*(beta*u);
```

Figure 1: An Octave/MATLAB implementation of the Krylov subspace method to compute $y(t) = \exp(-tA)v$ for a given $v \in \mathbb{R}^n$ based on the Arnoldi process.

to the other costs of the algorithm. These are dominated by the matrix–vector products with A and Gram-Schmidt orthogonalization (lines 15–19 of the algorithm). Therefore, if m is not too large, the choice of method to compute $\exp(-tH_{m,m})$ hardly influences the total performance. When implementing the algorithm in languages other than MATLAB/Octave, where `expm` is not available, to compute $\exp(-tH_{m,m})$ one could use C/C++ embeddable FORTRAN codes from the EXPOKIT package [61], in particular the `dgpadm.f` subroutine.

3.2. Computing action of $\varphi_1(-tA)$

The solution of IVP (10) can be written as $y(t) = v + t\varphi_1(-tA)(-Av + g_0)$, cf. (12). The Krylov subspace method described above can be easily adapted to compute the action of $\varphi_1(-tA)$. First of all, the initial vector v_1 of the Krylov subspace is defined as

$$\beta = \| - Av + g_0\|, \quad v_1 = \frac{1}{\beta}(-Av + g_0) \tag{21}$$

and the approximate Krylov subspace solution now reads

$$y_m = v + V_m \underbrace{t\varphi_1(-tH_{m,m})\beta e_1}_{u_m(t)}, \tag{22}$$

with the familiar Arnoldi matrices V_m and $H_{m,m}$ defined as above. Note is that $u_m(t)$ is not the same as one from (18) and satisfies inhomogeneous projected IVP

$$u_m' = -H_{m,m}u_m + \beta e_1, \quad u_m(0) = 0, \quad t \geqslant 0. \tag{23}$$

If we now introduce the residual of y_m as

$$r_m(t) \equiv -Ay_m(t) - y_m'(t) + g_0, \tag{24}$$

it is straightforward to show that $\|r_m(t)\|$ can be computed as given by (20) with the new u_m from (23).

The code given in Figure 1 should then be adjusted accordingly. The matrix function $\varphi_1(-tH_{m,m})$ can be computed with the help of the freely available code `phipade` [4].

```
function y = expm_ArnoldiSAI(A,v,t,toler,m)
% y = expm_ArnoldiSAI(A,v,t,toler,m)
% computes $y = \exp(-t A) v$
% input: A (n x n)-matrix, v n-vector, t>0 time interval,
%        toler>0 tolerance, m maximal Krylov dimension
n = size (v,1);
V = zeros(n  ,m+1);
H = zeros(m+1,m);
gamma = t/10;   I_gammaA = speye(n,n)+gamma*A;
[L,U,P,Q] = lu(I_gammaA);

beta = norm(v);  V(:,1) = v/beta;
for j=1:m
    w = Q*( U\( L\(P*V(:,j)) ) );
    for i=1:j
        H(i,j) = w'*V(:,i);
        w      = w - H(i,j)*V(:,i);
    end
    H(j+1,j) = norm(w);
    e1 = zeros(j,1); e1(1) = 1;
    ej = zeros(j,1); ej(j) = 1;
    invH = inv(H(1:j,1:j));
    Hjj = ( invH-eye(j,j) )/gamma;
    C   = norm(I_gammaA*w);
    s   = [1/3, 2/3, 1]*t;
    for q=1:length(s)
        u         = expm(-s(q)*Hjj)*e1;
        beta_j(q) = C/gamma * (ej'*(invH*u));
    end
    resnorm = norm(beta_j,'inf');
    fprintf('j = %d, resnorm = %.2e\n',j,resnorm);
    if resnorm<=toler
       break
    elseif j==m
       disp('warning: no convergence within m steps');
    end
    V(:,j+1) = w/H(j+1,j);
end
y = V(:,1:j)*(beta*u);
```

Figure 2: An Octave/MATLAB implementation of the Krylov subspace method to compute $y(t) = \exp(-tA)v$ for a given $v \in \mathbb{R}^n$ based on the Arnoldi/SAI process.

4. Krylov subspace Arnoldi shift-and-invert (SAI) method

4.1. Computing action of $\exp(-tA)$ *with Arnoldi/SAI*

A well-known problem with the Krylov subspace methods for evaluating the matrix exponential is their slow convergence for matrices with stiff spectrum, i.e., with the eigenvalues of both relatively small and large magnitude. The eigenvalues of the matrix $H_{m,m}$ tend to better approximate the large eigenvalues of A, whereas the components corresponding to these eigenvalues are not important for the matrix exponential (due to the exponential decay). To emphasize the important small eigenvalues, the so-called rational Krylov subspace approximations can be used [24, 28, 29]. A simple representative of this class of methods is the shift-and-invert (SAI) Krylov subspace method [47, 64]. The idea is to build up the Krylov subspace for the transformed, shifted and inverted matrix A, namely, for $(I + \gamma A)^{-1}$. Here γ is a parameter which in this note is set to $0.1t$, with t being the time interval from $\exp(-tA)v$. For more detail on the choice of γ see [64, Table 3.1].

The resulting algorithm is refered to as Arnoldi/SAI and proceeds as follows. The Krylov subspace built up for $(I + \gamma A)^{-1}$ gives, cf. (17),

$$\begin{aligned}(I + \gamma A)^{-1} V_m &= V_{m+1} \widetilde{H}_{m+1,m} \\ &= V_m \widetilde{H}_{m,m} + v_{m+1} \widetilde{h}_{m+1,m} e_m^T.\end{aligned} \tag{25}$$

The approximation $y_m(t) \approx \exp(-tA)v$ is then obtained by (18), with

$$H_{m,m} = \frac{1}{\gamma}(\widetilde{H}_{m,m}^{-1} - I). \tag{26}$$

Here we, in fact, apply the inverse SAI transformation on the projected matrix. To implement the Arnoldi/SAI algorithm, we can follow the lines of the regular Arnoldi algorithm, Figure 1. An important point is that the matrix $(I+\gamma A)^{-1}$ does not have to be available, only its *action* on vectors is needed. Of course, in many real life problems, including three-dimensional Maxwell's equations, obtaining $(I + \gamma A)^{-1}$ would not be possible. More precisely, the step $w := Av_j$ of the standard Arnoldi method (line 15 at Figure 1) transforms into $w := (I + \gamma A)^{-1} v_j$ in the Arnoldi/SAI method. To compute the vector w we then solve a linear system $(I + \gamma A)w = v_j$. This can be done either by a direct or a preconditioned iterative solver, similarly to implicit time integration schemes. In the former case, a (sparse) LU factorization of $I + \gamma A$ can be computed once at the beginning of the algorithm and reused every time a new vector w has to be computed. In

practice, the Arnoldi/SAI method often converges fast so that additional work to solve the SAI systems is paid off.

If a preconditioned iterative solver is used to solve $(I+\gamma A)w = v_j$, then we have to know when to stop the iterations. Too many iterations would be a waste of computational work, too few iterations could be harmful for the accuracy of the method. The question of a proper stopping criterion was analyzed in [64]: we do not have to solve the SAI system $(I+\gamma A)w = v_j$ very accurate and the tolerance to which it is solved can be relaxed as the Arnoldi iterations j converge. More precisely, the iterations in the inner SAI solver should be stopped as soon as the SAI residual vector $r_{(i)}^{\mathrm{SAI}} = v_j - (I+\gamma A)w_{(i)}$ satisfies

$$\frac{\|r_{(i)}^{\mathrm{SAI}}\|}{\|v_j\|} = \|r_{(i)}^{\mathrm{SAI}}\| \leqslant \frac{\texttt{toler}}{\texttt{resnorm} + \texttt{toler}}, \tag{27}$$

where $\|v_j\| = 1$ due the Gram-Schmidt orthonormalization, i is the iteration number in the inner SAI solver, $w_{(i)}$ is the approximate solution at iteration i, `toler` is the required tolerance for the vector $y_m(t) \approx \exp(-tA)v$ and `resnorm` is the exponential residual norm, cf. (19), evaluated at step j. Note that the SAI stopping criterion proposed in [64] slightly differs from the one we propose here in (27), namely, [64] uses an error bound rather than the exponential residual norm.

The resulting Arnoldi/SAI algorithm to compute $y(t) = \exp(-tA)v$ for a given $v \in \mathbb{R}^n$ is presented in Figure 2. The algorithm solves the SAI linear system $(I+\gamma A)w = v_j$ by the sparse LU factorization $PAQ = LU$, where P and Q are permutation matrices (PAQ is A with permuted rows and columns). This sparse factorization is provided by the UMFPACK package [16, 15] (and adopted in both Octave and MATLAB as the `lu` function). It is crucial for the computational performance that the factorization is computed once and reused every Krylov step.

The algorithm at Figure 2 has the same structure as the Arnoldi algorithm from Figure 1, i.e., the Gram-Schmidt orthogonalization of the new Krylov basis vector w is followed by computing $u_m(t)$ (cf. (18)) and the residual norm check. The main costs of the algorithm are the solution of the linear system $(I+\gamma A)w = v_j$ and the Gram-Schmidt orthogonalization.

Computing the residual (19) in the Arnoldi/SAI method is slightly more involved than in the Arnoldi method. The Arnoldi/SAI decomposition (25) can be transformed into

$$AV_m = V_m H_{m,m} - \frac{\widetilde{h}_{m+1,m}}{\gamma}(I+\gamma A)v_{m+1}e_m^T \widetilde{H}_{m,m}^{-1}.$$

Then

$$r_m(t) = -y_m' - Ay_m = (V_m H_m - AV_m)\exp(-tH_{m,m})(\beta e_1) \\ = \frac{\widetilde{h}_{m+1,m}}{\gamma}(I+\gamma A)v_{m+1}\big[e_m^T \widetilde{H}_{m,m}^{-1} u_m(t)\big], \tag{28}$$

where the expression in the square brackets is a scalar value, namely the last component of the vector $\widetilde{H}_{m,m}^{-1}u_m(t)$, with $u_m(t)$ defined in (18). The value `resnorm` computed by the algorithm at Figure 2 is again the relative residual norm $\texttt{resnorm} = \|r_m(s)\|/\beta$ computed for several values $s \in (0,t]$.

4.2. Computing action of $\varphi_1(-tA)$ with Arnoldi/SAI

The described Arnoldi/SAI method can be easily applied to compute the action of $\varphi_1(-tA)$ as well. The starting Krylov vector v_1 should be defined according to (21) and the approximate solution $y_m(t)$ is given by (22),(23). It is easy to show that the residual of $y_m(t)$, defined for $\varphi_1(-tA)$ as $r_m(t) = -Ay_m(t) - y_m'(t) + g_0$, satisfies (28), namely,

$$r_m(t) = \big[e_m^T \widetilde{H}_{m,m}^{-1} u_m(t)\big],$$

where $u_m(t)$ is defined in (23).

5. A numerical example

This numerical test comes from the field of electromagnetic imaging and fault detection in gas-and-oil industry [60]. Maxwell's equations (1) are posed in a cubical physical domain $[-20,20]^3$ (the size is given in meters), which is divided by the plane $x = 10$ into two regions, where the conductivity is defined as

$$\sigma = \begin{cases} 0.1\,\mathrm{S/m}, & x \leqslant 10, \\ 0.001\,\mathrm{S/m}, & x > 10, \end{cases} \tag{29}$$

and $\mu = \mu_0$, $\varepsilon = \varepsilon_0$ in the whole domain. In the larger region $x \leqslant 10$ a coil of a square shape is placed connecting four points, whose coordinates (x,y,z) are $(-2,-2,0)$, $(-2,2,0)$, $(2,2,0)$ and $(2,-2,0)$. The boundary conditions are the far field conditions (homogeneous Dirichlet) and the initial conditions are zero for the both fields. At the initial time $t = 0\,\mathrm{s}$ a current in the coil is switched on and increases linearly to reach 1 A at the time moment $t = 10^{-6}\,\mathrm{s}$. The current remains constant for $10^{-4}\,\mathrm{s}$, is switched off at $t = 1.01\times10^{-4}\,\mathrm{s}$ and decays linearly to reach its zero value at $t = 1.02\times10^{-4}\,\mathrm{s}$.

Table 1: Results for the test runs on the $20 \times 20 \times 20$ mesh, $n = 55\,566$. The CPU timings are made in MATLAB and thus give only an indication of the actual performance. The results for Arnoldi/SAI, $T = 750$ are given for two different tolerance values.

scheme	T	# time steps	CPU time, s	rel. error	Krylov dimension
CO2	100	4000	18	4.6e−07	—
Arnoldi	100	1805	$> 1\,000$	1.0e−03[a]	restart 300
Arnoldi/SAI	100	1	6.5	1.5e−10	25
ITR	100	400	31	2.7e−05	—
EXPOKIT	100	—	143	1.1e−10[b]	100
CO2	750	30 000	142	2.2e−07	—
Arnoldi/SAI	750	4	7.3	2.7e−05	17,12,5,8
Arnoldi/SAI	750	4	9.8	2.1e−08	31,16,8,6

[a] a higher accuracy can be reached if necessary
[b] provided by the EXPOKIT error estimator

After that the current remains zero until the final time 2.02×10^{-4} s is reached.

We solve Maxwell's equations in a dimensionless form, which is obtained by the introducing the dimensionless quantities as

$$\begin{aligned} x &= \frac{1}{L} x_s \quad \text{(similarly for } y,\, z\text{)}, \qquad t = \frac{c_0}{L} t_s, \\ E &= \frac{1}{H_0 Z_0} E_s = \frac{1}{H_0 \cdot 120\pi} E_s, \qquad J = \frac{L}{H_0} J_s, \end{aligned} \tag{30}$$

where the subindex $\cdot_s$ is used to indicate the values in the SI units, $L = 40\,\mathrm{m}$ is the typical length, $H_0 = 1\,\mathrm{A/m}$ is the typical magnetic strength, $c_0 = 1/\sqrt{\mu_0 \varepsilon_0} \approx 3 \times 10^8\,\mathrm{m/s}$ is the speed of light in vacuum, $Z_0 = \sqrt{\mu_0/\varepsilon_0} = 120\pi\,\Omega$ is the free space intrinsic impedance. Note that the dimensionless scaling introduces the factor $Z_0 L = 4800\pi$ in the conductivity values (29), which makes the problem mildly stiff. We discretize the problem in space by the standard Yee finite differences.

The tests are carried out in MATLAB on a powerful Linux computer with two "quad core" 2.40GHz CPU's, each with 48 Gb memory. The results of the test runs are presented in Tables 1 and 2. Several methods are compared there: the CO2 scheme (4), the Crank–Nicolson ITR scheme (6), an exponential scheme `phiv.m` of the EXPOKIT package [61], the Arnoldi

Table 2: Results for the test runs on the $40 \times 40 \times 40$ mesh, $n = 413\,526$. The CPU timings are made in MATLAB and thus give only an indication of the actual performance. The results for Arnoldi/SAI, $T = 750$ are given for two different tolerance values.

scheme	T	# time steps	CPU time, s	rel. error	Krylov dimension
CO2	100	8000	130	1.2e−07	—
Arnoldi	100	2318	$> 3\,000$	3.0e−01[a]	restart 300
Arnoldi/SAI	100	1	133	2.1e−08	20
ITR	100	400	903	3.9e−05	—
EXPOKIT	100	—	2 673	1.7e−10[b]	100
CO2	750	60 000	1142	5.6e−08	—
Arnoldi/SAI	750	4	203	4.7e−05	17,10,5,8
Arnoldi/SAI	750	4	225	1.2e−07	28,14,10,6

[a] a higher accuracy can be reached if necessary
[b] provided by the EXPOKIT error estimator

and Arnoldi/SAI methods. The runs are done for two time intervals $[t_0; t_0 + T]$, with the initial (dimensionless) time $t_0 = 765$ (the moment when the coil current has become zero) and either $T = 100$ or $T = 750$. In the latter case the dimensionless time 1515 corresponds to the physical time 2.02×10^{-4} s, the final time of interest. The initial values for $t_0 = 765$ and the reference solution for $t_0 + T$ are obtained by running the CO2 scheme with a tiny time step and extrapolating the results. The relative error with respect to the reference solution y_{ref} is computed as $\|y - y_{\text{ref}}\|/\|y_{\text{ref}}\|$, with y being the numerical solution vector containing all the degrees of freedom for both fields.

The CO2 scheme is run with roughly a maximal allowable time step size (increasing the time step size by a factor of two leads to an instability). As we see from Tables 1 and 2, the regular Arnoldi method is not efficient. The method has been used in combination with restarting after every 300 Krylov steps. The restarting is similar to the restarting for linear system Krylov solvers, e.g. for GMRES(m) [57], and means that at most $m = 300$ Krylov vectors have to be stored. There are different restarting strategies for Krylov subspace matrix exponential methods [63, 1, 22, 28, 50], the one we used is the residual based restarting from [7] which seems to be well suited for the matrix exponential. Taking a restart value different than 300 does not help.

The Arnoldi/SAI method is used with the UMFPACK sparse LU solver

Table 3: CPU time needed to compute the sparse LU factorization of the matrix $I + \gamma A$ versus the $\tau \equiv 10\gamma$ values. The $40 \times 40 \times 40$ mesh is used ($n = 413\,526$). The fill-in factors are approximately 250 for $\tau \leqslant 200$ and 1000 for $\tau = 400$.

$\tau \equiv 10\gamma$	50	100	200	400
CPU time, s	152.7	148.9	153.8	3989

(the `lu` function in MATLAB), which provides the LU factorization as discussed in Section 4. This sparse solver uses strategies with compromise between the sparsity in the triangular factors and numerical stability of the LU factorization. Increasing the time interval τ leads at some point to a very off-diagonal-dominant matrix $I + \gamma A$, $\gamma = \tau/10$, and, hence, to a dramatic increase in the CPU time to compute its sparse LU factorization, see Table 3. For this reason we use the Arnoldi/SAI method with the time step $\tau = 200$ at most. For the time interval $T = 750$, the method carries out four time steps, $3 \times 200 + 1 \times 150$, and uses the same LU factorization for all four steps. For this reason, the CPU timings for Arnoldi/SAI are not proportional to the time interval T. Note that using the LU factorization computed for $\tau = 200$ for the last time step $\tau = 150$ means that we effectively change the value of γ. This is not a problem because, as observed in [64] and confirmed in our experiments, the Arnoldi/SAI is known to be not very sensitive to the choice of γ.

It is important to realize that a similar, efficient performance could be achieved with Arnoldi/SAI method combined with an efficient preconditioned iterative solver, if one was available. However, standard preconditioners appear not to be efficient for this problem, see [67] for possible iterative solution strategies in the context of Maxwell's equations.

Of course, once a sparse LU factorization is affordable, one can use a fully implicit scheme such as ITR. The scheme computes the same (as used for the SAI system) sparse LU factorization once and keeps on using it all the time steps. However, the CPU timings of the scheme are much higher than for Arnoldi/SAI due many more time steps needed.

This numerical example with ITR seems to be rather instructive. Indeed, assume a more complex problem is solved, when the source function $g(t)$ in (3) is not constant and, hence, the simple evaluation $y_m(t) \approx \exp(-tA)v$ or $y_m(t) \approx \varphi_1(-tA)v$ does not suffice to get a solution. The test shows that an exponential time integration scheme, such as EK2 (15), combined with the Arnoldi/SAI method will likely not be more efficient than CO2. Indeed,

one would need to make many more time steps (and many more matrix function evaluations). These arguments are confirmed by the tests reported for EK2 in [67]. Thus, for general time dependent $g(t)$ we need exponential time integration methods which would allow (a) very large steps without an accuracy loss and (b) reusing numerical linear algebra work as much as possible. An example of such a scheme is presented in [8].

The EXPOKIT package [61] is able to compute the action of a large exponential or the φ_1 matrix function on a given vector, this can be done with the help of the `phiv` function of EXPOKIT. This function employs an Arnoldi method, where the time interval is divided into subintervals to facilitate convergence (this approach is recently extended in [51]). In the tests reported in Tables 1 and 2 EXPOKIT's `phiv` was run with maximal Krylov dimension increased to 100 (the default EXPOKIT's value is 30).

6. Conclusions

We have presented some recent numerical techniques for computing the action of the matrix exponential of a big matrix in the context of time dependent Maxwell's equations. In particular, a Krylov subspace Arnoldi method, combined with the shift-and-invert (SAI) technique and a residual-based stopping criterion, is shown to be a very competitive tool for the test problem.

Due to the more work per time step than in conventional schemes, the exponential methods seem to be efficient only for sufficiently large time steps or when the additional numerical linear algebra work can be reused through the time stepping (see numerical results and discussion in Section 5 and in [67]). An attractive feature of the schemes is that their accuracy can often be retained for very large time steps.

A closely related issue is the efficiency of the SAI system solvers. Although formally the time step size may not be bounded, as the exponential schemes discussed here are unconditionally stable *and* exact for any step, in practice the time step size can often be bounded by the solver. Indeed, solving the SAI system for very large time steps may become very expensive.

Furthermore, the presented experiments show superiority of the exponential schemes with respect to the standard implicit schemes in cases when both types of schemes can be implemented efficiently.

In overall, the efficient solution of the SAI systems seems to be very important. Therefore it is advisable to concentrate a further research on development of (parallel) preconditioned iterative solvers, possibly including the so-called Krylov subspace recycling [52].

The MATLAB functions `expm_Arnoldi` and `expm_ArnoldiSAI` described in Sections 3 and 4 can be downloaded together with this note from `http://eprints.eemcs.utwente.nl/`.

Acknowledgments

The author would like to thank Oleg Nechaev for his kind advise concerning the implementation of the test problem.

References

[1] M. Afanasjew, M. Eiermann, O. G. Ernst, and S. Güttel. Implementation of a restarted Krylov subspace method for the evaluation of matrix functions. *Linear Algebra Appl.*, 429:2293–2314, 2008.

[2] A. H. Al-Mohy and N. J. Higham. Computing the action of the matrix exponential, with an application to exponential integrators. *SIAM J. Sci. Comput.*, 33(2):488–511, 2011. `http://dx.doi.org/10.1137/100788860`.

[3] R. Barrett, M. Berry, T. F. Chan, J. Demmel, J. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine, and H. A. van der Vorst. *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods*. SIAM, Philadelphia, PA, 1994. Available at `www.netlib.org/templates/`.

[4] H. Berland, B. Skaflestad, and W. M. Wright. EXPINT—a MATLAB package for exponential integrators. *ACM Trans. Math. Softw.*, 33(1), Mar. 2007. `http://www.math.ntnu.no/num/expint/`.

[5] S. Blanes, F. Casas, J. A. Oteo, and J. Ros. The magnus expansion and some of its applications. *Physics Reports*, 470(5–6):151–238, 2009. `http://dx.doi.org/10.1016/j.physrep.2008.11.001`.

[6] A. Bossavit. *Computational electromagnetism. Variational formulations, complementarity, edge elements*. Electromagnetism. Academic Press Inc., San Diego, CA, 1998.

[7] M. A. Botchev. Residual, restarting and Richardson iteration for the matrix exponential, revised. *ArXiv e-prints*, Dec. 2011. `http://arxiv.org/abs/1112.5670`.

[8] M. A. Botchev. A block Krylov subspace time-exact solution method for linear ODE systems. Memorandum 1973, Department of Applied Mathematics, University of Twente, Enschede, January 2012. http://eprints.eemcs.utwente.nl/21277/.

[9] M. A. Botchev, D. Harutyunyan, and J. J. W. van der Vegt. The Gautschi time stepping scheme for edge finite element discretizations of the Maxwell equations. *J. Comput. Phys.*, 216:654–686, 2006. http://dx.doi.org/10.1016/j.jcp.2006.01.014.

[10] M. A. Botchev and J. G. Verwer. Numerical integration of damped Maxwell equations. *SIAM J. Sci. Comput.*, 31(2):1322–1346, 2009. http://dx.doi.org/10.1137/08072108X.

[11] K. Busch, J. Niegemann, M. Pototschnig, and L. Tkeshelashvili. A Krylov-subspace based solver for the linear and nonlinear Maxwell equations. *Phys. Stat. Sol. (b)*, 244(10):3479–3496, 2007.

[12] M. Caliari and A. Ostermann. Implementation of exponential Rosenbrock-type integrators. *Appl. Numer. Math.*, 59(3-4):568–581, 2009.

[13] J. Certaine. The solution of ordinary differential equations with large time constants. In K. E. A. Ralston, H.S. Wilf, editor, *Mathematical Methods for Digital Computers*, pages 128–132. Wiley, New York, 1960.

[14] S. M. Cox and P. C. Matthews. Exponential time differencing for stiff systems. *J. Comput. Phys.*, 176(2):430–455, 2002.

[15] T. A. Davis. Algorithm 832: UMFPACK V4.3—an unsymmetric-pattern multifrontal method. *ACM Trans. Math. Software*, 30(2):196–199, 2004.

[16] T. A. Davis. A column pre-ordering strategy for the unsymmetric-pattern multifrontal method. *ACM Trans. Math. Software*, 30(2):167–195, 2004.

[17] H. De Raedt, K. Michielsen, J. S. Kole, and M. T. Figge. One-step finite-difference time-domain algorithm to solve the Maxwell equations. *Phys. Rev. E*, 67:056706, 2003.

[18] V. Druskin, A. Greenbaum, and L. Knizhnerman. Using nonorthogonal Lanczos vectors in the computation of matrix functions. *SIAM J. Sci. Comput.*, 19(1):38–54, 1998.

[19] V. L. Druskin and L. A. Knizhnerman. Two polynomial methods of calculating functions of symmetric matrices. *U.S.S.R. Comput. Maths. Math. Phys.*, 29(6):112–121, 1989.

[20] V. L. Druskin and L. A. Knizhnerman. Krylov subspace approximations of eigenpairs and matrix functions in exact and computer arithmetic. *Numer. Lin. Alg. Appl.*, 2:205–217, 1995.

[21] V. L. Druskin and L. A. Knizhnerman. Extended Krylov subspaces: approximation of the matrix square root and related functions. *SIAM J. Matrix Anal. Appl.*, 19(3):755–771 (electronic), 1998.

[22] M. Eiermann, O. G. Ernst, and S. Güttel. Deflated restarting for matrix functions. *SIAM J. Matrix Anal. Appl.*, 32(2):621–641, 2011. `http://dx.doi.org/10.1137/090774665`.

[23] A. Frommer and V. Simoncini. Matrix functions. In W. H. A. Schilders, H. A. van der Vorst, and J. Rommes, editors, *Model Order Reduction: Theory, Research Aspects and Applications*, pages 275–304. Springer, 2008.

[24] A. Frommer and V. Simoncini. Stopping criteria for rational matrix functions of Hermitian and symmetric matrices. *SIAM J. Sci. Comput.*, 30(3):1387–1412, 2008.

[25] E. Gallopoulos and Y. Saad. Efficient solution of parabolic equations by Krylov approximation methods. *SIAM J. Sci. Statist. Comput.*, 13(5):1236–1264, 1992.

[26] F. R. Gantmacher. *The Theory of Matrices. Vol. 1.* AMS Chelsea Publishing, Providence, RI, 1998. Translated from the Russian by K. A. Hirsch, Reprint of the 1959 translation.

[27] G. H. Golub and C. F. Van Loan. *Matrix Computations*. The Johns Hopkins University Press, Baltimore and London, third edition, 1996.

[28] S. Güttel. *Rational Krylov Methods for Operator Functions*. PhD thesis, Technischen Universität Bergakademie Freiberg, March 2010. `www.guettel.com`.

[29] S. Güttel. Rational Krylov approximation of matrix functions: Numerical methods and optimal pole selection. Preprint submitted for publication, March 2012. `www.guettel.com`.

[30] E. Hairer, C. Lubich, and G. Wanner. *Geometric numerical integration. Structure-preserving algorithms for ordinary differential equations*. Springer-Verlag, Berlin, second edition, 2006.

[31] N. J. Higham. The scaling and squaring method for the matrix exponential revisited. *SIAM J. Matrix Anal. Appl.*, 26(4):1179–1193, 2005.

[32] N. J. Higham. *Functions of Matrices: Theory and Computation*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2008.

[33] N. J. Higham and A. H. Al-Mohy. Computing matrix functions. *Acta Numer.*, 19:159–208, 2010.

[34] M. Hochbruck and C. Lubich. On Krylov subspace approximations to the matrix exponential operator. *SIAM J. Numer. Anal.*, 34(5):1911–1925, Oct. 1997.

[35] M. Hochbruck, C. Lubich, and H. Selhofer. Exponential integrators for large systems of differential equations. *SIAM J. Sci. Comput.*, 19(5):1552–1574, 1998.

[36] M. Hochbruck and A. Ostermann. Exponential integrators. *Acta Numer.*, 19:209–286, 2010.

[37] M. Hochbruck, A. Ostermann, and J. Schweitzer. Exponential Rosenbrock-type methods. *SIAM J. Numer. Anal.*, 47(1):786–803, 2008/09. `http://dx.doi.org/10.1137/080717717`.

[38] R. Horváth, I. Faragó, and W. Schilders. Investigation of numerical time-integrations of Maxwell's equations using the staggered grid spatial discretization. *Int. J. Numer. Model.*, 18:149–169, 2005.

[39] L. Knizhnerman and V. Simoncini. A new investigation of the extended Krylov subspace method for matrix function evaluations. *Numer. Linear Algebra Appl.*, 2009. To appear.

[40] L. A. Knizhnerman. Calculation of functions of unsymmetric matrices using Arnoldi's method. *U.S.S.R. Comput. Maths. Math. Phys.*, 31(1):1–9, 1991.

[41] V. I. Lebedev. Explicit difference schemes for solving stiff systems of ODEs and PDEs with complex spectrum. *Russian J. Numer. Anal. Math. Modelling*, 13(2):107–116, 1998.

[42] J. Legras. Résolution numérique des grands systèmes différentiels linéaires. *Numer. Math.*, 8:14–28, 1966.

[43] X. Ma, X. Zhao, and Y. Zhao. A 3-d precise integration time-domain method without the restraints of the Courant-Friedrich-Levy stability condition for the numerical solution of Maxwell's equations. *Microwave Theory and Techniques, IEEE Transactions on*, 54(7):3026–3037, july 2006.

[44] B. V. Minchev and W. M. Wright. A review of exponential integrators for first order semi-linear problems. Technical Report 2/05, Department of Mathematics, NTNU, Norway, April 2005. `http://www.ii.uib.no/~borko/pub/N2-2005.pdf`.

[45] C. B. Moler and C. F. Van Loan. Nineteen dubious ways to compute the exponential of a matrix, twenty-five years later. *SIAM Rev.*, 45(1):3–49, 2003.

[46] P. Monk. *Finite Element Methods for Maxwell's Equations*. Oxford University Press, 2003.

[47] I. Moret and P. Novati. RD rational approximations of the matrix exponential. *BIT*, 44:595–615, 2004.

[48] J.-C. Nédélec. Mixed finite elements in $\mathbf{R}^3$. *Numer. Math.*, 35(3):315–341, 1980.

[49] J.-C. Nédélec. A new family of mixed finite elements in $\mathbf{R}^3$. *Numer. Math.*, 50(1):57–81, 1986.

[50] J. Niehoff. *Projektionsverfahren zur Approximation von Matrixfunktionen mit Anwendungen auf die Implementierung exponentieller Integratoren*. PhD thesis, Mathematisch-Naturwissenschaftlichen Fakultät der Heinrich-Heine-Universität Düsseldorf, December 2006.

[51] J. Niesen and W. M. Wright. Algorithm 919: A Krylov subspace algorithm for evaluating the φ-functions appearing in exponential integrators. *ACM Trans. Math. Softw.*, 38(3):22:1–22:19, Apr. 2012.

[52] M. L. Parks, E. de Sturler, G. Mackey, D. D. Johnson, and S. Maiti. Recycling Krylov subspaces for sequences of linear systems. *SIAM J. Sci. Comput.*, 28(5):1651–1674, 2006.

[53] P. G. Petropoulos. Analysis of exponential time-differencing for FDTD in lossy dielectrics. *IEEE transactions on antennas and propagation*, 45(6):1054–1057, 1997.

[54] G. Rodrigue and D. White. A vector finite element time-domain method for solving Maxwell's equations on unstructured hexahedral grids. *SIAM J. Sci. Comput.*, 23(3):683–706, 2001.

[55] Y. Saad. Analysis of some Krylov subspace approximations to the matrix exponential operator. *SIAM J. Numer. Anal.*, 29(1):209–228, 1992.

[56] Y. Saad. *Iterative Methods for Sparse Linear Systems*. Book out of print, 2000. `www-users.cs.umn.edu/~saad/books.html`.

[57] Y. Saad and M. H. Schultz. GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM J. Sci. Stat. Comput.*, 7(3):856–869, 1986.

[58] T. Schmelzer and L. N. Trefethen. Evaluating matrix functions for exponential integrators via Carathéodory-Fejér approximation and contour integrals. *Electron. Trans. Numer. Anal.*, 29:1–18, 2007/08.

[59] W. Schoenmaker. Speeding-up transient EM-TCAD using matrix exponential forms. Presentation at the European Conference for Mathematics in Industry, ECMI2012, July 2012. Lund, Sweden.

[60] E. P. Shurina, A. V. Gelber, M. A. Gelber, and M. I. Epov. Mathematical modelling of non-stationary electromagnetic fields of defectoscope of casings. *Computational technologies*, 7(6):114–129, 2002. In Russian. `www.ict.nsc.ru/jct/annotation/346?l=eng`.

[61] R. B. Sidje. EXPOKIT. A software package for computing matrix exponentials. *ACM Trans. Math. Softw.*, 24(1):130–156, 1998. `www.maths.uq.edu.au/expokit/`.

[62] H. Tal-Ezer. Spectral methods in time for parabolic problems. *SIAM J. Numer. Anal.*, 26(1):1–11, 1989.

[63] H. Tal-Ezer. On restart and error estimation for Krylov approximation of $w = f(A)v$. *SIAM J. Sci. Comput.*, 29(6):2426–2441 (electronic), 2007.

[64] J. van den Eshof and M. Hochbruck. Preconditioning Lanczos approximations to the matrix exponential. *SIAM J. Sci. Comput.*, 27(4):1438–1457, 2006.

[65] H. A. van der Vorst. An iterative solution method for solving $f(A)x = b$, using Krylov subspace information obtained for the symmetric positive definite matrix A. *J. Comput. Appl. Math.*, 18:249–263, 1987.

[66] H. A. van der Vorst. *Iterative Krylov methods for large linear systems.* Cambridge University Press, 2003.

[67] J. G. Verwer and M. A. Botchev. Unconditionally stable integration of Maxwell's equations. *Linear Algebra and its Applications*, 431(3–4):300–317, 2009.

[68] R. C. Ward. Numerical computation of the matrix exponential with accuracy estimate. *SIAM J. Numer. Anal.*, 14(4):600–610, 1977.

[69] K. S. Yee. Numerical solution of initial boundary value problems involving Maxwells equations in isotropic media. *IEEE Trans. Antennas Propagat.*, 14(3):302–307, March 1966.