Krylov subspace exponential time domain solution of Maxwell's equations in photonic crystal modeling[☆]

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, `botchev@ya.ru`.

Abstract

The exponential time integration, i.e., time integration which involves the matrix exponential, is an attractive tool for time domain modeling involving Maxwell's equations. However, its application in practice often requires a substantial knowledge of numerical linear algebra algorithms, such as Krylov subspace methods.

In this note we discuss exponential Krylov subspace time integration methods and provide a simple guide on how to use these methods in practice. While specifically aiming at nanophotonics applications, we intentionally keep the presentation as general as possible and consider full vector Maxwell's equations with damping (i.e., with nonzero conductivity terms).

Efficient techniques such as the Krylov shift-and-invert method and residual-based stopping criteria are discussed in detail. Numerical experiments are presented to demonstrate the efficiency of the discussed methods and their mesh independent convergence. Some of the algorithms described here are available as Octave/Matlab codes from `www.math.utwente.nl/~botchevma/expm/`.

Keywords: finite difference time domain (FDTD) method, matrix exponential, Maxwell's equations, Krylov subspace methods, photonic crystals
2008 MSC: 65F60, 65F30, 65N22, 65L05, 35Q61

[☆]Supported in part by the Russian Science Foundation, Grant 14-11-00659 during author's stay at Institute of Numerical Mathematics, Russian Academy of Sciences, Gubkin St. 8, Moscow 119333, Russia.

1. Introduction

Photonic crystals, periodic nanostructures made of dielectric materials, have been instrumental in many areas of science and technology, where understanding and controlling optical material properties is the key issue [49, 55, 74]. Usually, physical models of processes in photonic crystals are based on Maxwell's equations formulated in time or frequency domain. 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 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 [92] and the edge–face Whitney–Nédélec vector finite elements [64, 65, 8, 62].

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 conductivity terms[1]. 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.

Within the electromagnetic engineering community, exponential methods seem to slowly but surely gain acceptance and popularity over the last years, see e.g. [23, 59, 16, 76]. 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 efficient stopping criteria [84, 31, 37, 11, 10]. Some promising results with Krylov / SAI type of methods have recently been reported for electromagnetic problems [47, 7] in the Numerical Analysis community. Yet, up to now we are unaware of any single electromagnetics paper where the Krylov SAI exponential meth-

[1]In nanophotonics applications the conductivity is typically zero. However, large nonzero conductivity values can be introduced by the so-called absorbing or nonreflecting boundary conditions such as Perfectly Matched Layers (PMLs).

ods are employed to solve Maxwell's equations. This note aims to fill this gap by demonstrating the efficiency of these new methods and providing a simple guide on how to use these methods in practice and related software.

We note that in many cases splitting methods are a very competitive class of time integration methods, especially when the underlying spatial mesh is Cartesian (see [80, Chapter 18] and [48, 53, 54, 71, 15, 89]). A popular splitting method (related to the famous Yee scheme and referred here to as CO2, Composition Order 2 scheme) is discussed below and used in the experiments presented. Multirate explicit local time stepping schemes form another important class of time integration methods for solving Maxwell's equations, see e.g. [35].

While specifically aiming at nanophotonics applications, we intentionally keep the presentation as general as possible and consider full vector Maxwell's equations

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

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 the magnetic permeability, $\varepsilon = \varepsilon(x, y, z)$ is the electric permittivity, $\sigma = \sigma(x, y, z)$ is the R1:1
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. [71, 15].

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

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

We emphasize that, unless the mass matrices are diagonal, one does not need to compute their inverses. Indeed, as generally accepted in the numerical literature, the notation M^{-1} here means a system solution with a matrix M rather than its explicit inversion. The systems with the mass matrices can be solved 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 large, 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, 87, 73]. We note that the mass lumping techniques can also be employed in this context, even though a special care should be taken for vector finite element discretizations (see, e.g., [4, 29]). To keep the presentation general, we use the non-lumped form in this article. R2:1 R2a:1

If ε and σ are constant[2] then the eigenvalues of A can be expressed in terms of the singular values of the discrete rotor operator. In this case system (2) can be uncoupled by an orthogonal transformation into a set of harmonic oscillators, see [15]. R1:2

Application of the Krylov/SAI type of methods to electromagnetic problems has recently been discussed in [47, 7]. The focus of [47] is on discontinuous Galerkin formulations for various wave problems, where the considered exponential time integration methods employ matrix exponentials *within* each time step. Our approach discussed here is aimed at using matrix exponential *across* time steps, which, in our experience, often leads to a better efficiency. The work [7] is devoted to a transient problem leading to a ODE system with symmetric mass and stiffness curl-curl matrices, so that the

[2]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.

full nonsymmetric Maxwell system is not considered. We note that Krylov subspace methods are by no means the only way to compute the actions of the matrix exponentials, but it is considered as one of the most efficient methods for large scale matrices [61, 41]. Other possible methods, applicable especially when the matrix is symmetric or skew-symmetric, include e.g. [81, 82, 23, 2, 56, 57]. However, the full Maxwell matrix considered in this note is neither symmetric nor skew-symmetric in general.

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. We discuss the Krylov subspace Krylov/SAI method and its implementation. Numerical tests are presented in Section 4 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 [15] often used in the geometric time integration [38]. 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

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

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 [15], and it can also be used in practical computations at the first and last time steps. In the vector finite element context, paper [71] 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 [15], 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, 87, 73].

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 least for smooth initial data. 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) [90]. 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 R2:2

be efficient and a choice of a proper preconditioned iterative solver is far from trivial, see [90] 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 advantageous to use another type of scheme, namely, an exponential time integration scheme. This will be demonstrated in the numerical experiments section.

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. [48, 15, 88]. We now give a short discussion of the exponential time integration. It is an active field of research [45, 5, 6, 60, 67, 42] and we only give some basic ideas here. 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 the matrix exponential [33, 34, 61, 40, 30, 41]. 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, $j = 0, 1, \dots$, which are extensively used in exponential time integration:

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

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:

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

This formula is a backbone for exponential time integration, it often serves as a starting point for the derivation of exponential integrators. Note that, R1:3 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. [44]. 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[3].

We now give an example of second-order accurate exponential time integrator, called EK2, exponential Krylov scheme of order 2, see [90]:

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

This method, which goes back to [19, 58], can be derived by interpolating the function g under the integral in (13) linearly and evaluating the integral. This approach is sometimes referred to as exponential time differencing [20, 70] and is closely related to a class of exponential Runge–Kutta–Rosenbrock methods [46].

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 [90], even though some promising, in terms of computational efficiency, results with exponential integration are reported in [12]. An approach to reduce the number of matrix function actions and, hence, to increase efficiency of the exponential integration is presented in [10].

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 [86, 25, 52, 32, 72, 26, 43, 44, 27], the Chebyshev polynomials, scaling and squaring with Padé or Taylor approximations and other methods [82, 23, 75, 17,

[3]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.

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 [57]. 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. [23].

3.1. Computing the action of $\exp(-tA)$ R1:4

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

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

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 (see, e.g., [87, 73]), which is also implemented in the software provided with this article [9]. As will be discussed slightly later, the software algorithm R1a:1
determines the value of m adaptively, based on the accuracy requirements. R2:4
The matrix V_m satisfies the so-called Arnoldi decomposition [87, 73]

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

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

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

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 $e_1 = [1\,, 0, \dots 0]^T \in \mathbb{R}^m$ 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 a hundred approximately. The exponential of $-tH_{m,m}$ then can be computed R1:5
by any suitable scheme for matrices of a moderate size, see e.g. [40, Chapter 10] and [78, 5]. Note that software packages Matlab and Mathematica use the scaling and squaring algorithm of [39].

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 available as the function `expm_Arnoldi.m` in [9]. An essential part of the algorithm is a residual-based stopping criterion, see [18, 24, 51, 11]. 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 [18]

$$r_m(t) \equiv -Ay_m(t) - y_m'(t). \qquad (19)$$

Indeed, with (17) and (18) it is not difficult to see that [24, 11]

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

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 $(\int_0^t \|r_m(s)\|^2 \mathrm{d}s)^{1/2}$. In practice it appears to be sufficient to R1:6
compute the residual norm at several time moments $s \in (0, t]$. For more R2:5
detail and discussion on the relation to other possible stopping criteria we refer to [11].

The exponential $\exp(-tH_{m,m})$ 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 [91] and [39], 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 to the other costs of the algorithm. These are dominated by the matrix–vector products with A and Gram-Schmidt orthogonalization of the Arnoldi process. 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 [78], in particular the `dgpadm.f` subroutine.

3.2. Computing the action of $\varphi_1(-tA)$ R1:7

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 the inhomogeneous projected IVP R1:8

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

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 `expm_Arnoldi.m` in [9] should then be adjusted accordingly. The small matrix function $\varphi_1(-tH_{m,m})$ can be computed with the help of the freely available code `phipade` [5].

3.3. Actions of $\exp(-tA)$ *with the Krylov shift-and-invert (SAI) method*

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 [31, 36, 37]. A relatively simple yet powerful representative of this class of methods is the shift-and-invert (SAI) Krylov subspace method [63, 84]. 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, to be chosen later (see the numerical experiments section).

The resulting algorithm is referred to as Krylov/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. An implementation of the algorithm, available as the function `expm_ArnoldiSAI.m` in [9], closely follows the lines of `expm_Arnoldi.m` discussed above. 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. To carry out operations of the form $w := (I+\gamma A)^{-1}v_j$ in the Krylov/SAI method, we 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 Krylov/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 and convergence of the method. Since the action of $(I + \gamma A)^{-1}$ is computed approximately, we have a method which belongs to a class of the R2:6
inexact Krylov subspace methods. As analysis of these methods suggests

(see, e.g., [79, 85]), the inexactness of the matrix-vector multiplication (in our case, with $(I+\gamma A)^{-1}$) can be bounded to maintain the convergence of the original exact Krylov subspace method. The question of a proper stopping criterion of the iterative linear solver within the inexact Krylov/SAI method is analyzed in [84]: we do not have to solve the SAI system $(I+\gamma A)w = v_j$ very accurately and the tolerance to which it is solved can be relaxed as R1:9
the Krylov iterations j converge. More precisely, the iterations in the inner SAI solver should be stopped as soon as the SAI residual vector $r_{(i)}^{\text{SAI}} = v_j - (I+\gamma A)w_{(i)}$ satisfies

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

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 [84] slightly differs from the one we propose here in (27), namely, [84] uses an error bound rather than the exponential residual norm.

The resulting Krylov/SAI algorithm to compute $y(t) = \exp(-tA)v$ for a given $v \in \mathbb{R}^n$ (available as `expm_ArnoldiSAI.m` in [9]) 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 [22, 21] (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 in every Krylov step. R1:10

R1:11 The algorithm of `expm_ArnoldiSAI.m` in [9] has the same structure as the Krylov algorithm from `expm_Arnoldi.m`, 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 Krylov/SAI method is slightly more involved than in the Krylov method. The Krylov/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

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

where the expression in the square brackets is a scalar value, namely the R1:12
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 of `expm_ArnoldiSAI.m` in [9] is R1:13
again the relative residual norm `resnorm` $= \|r_m(s)\|/\beta$ computed for several
values $s \in (0, t]$.

3.4. Actions of $\varphi_1(-tA)$ with Krylov/SAI

The described Krylov/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).

4. Numerical experiments

We now present two numerical tests. The first one is a photonic crystal model where the conductivity damping is present due to the absorbing PML (perfectly matched layer) boundary conditions. The second test comes from gas-and-oil industry applications; here the piecewise constant conductivity plays a central role in the model: it allows for detection of subterranean structures.

In both tests, we solve Maxwell's equations in a dimensionless form, which is obtained by the introducing the dimensionless quantities as

$$\begin{aligned} x &= \frac{1}{L}x_s \quad \text{(similarly for } y,\ z), & t &= \frac{c_0}{L}t_s, \\ E &= \frac{1}{H_0 Z_0}E_s = \frac{1}{H_0 \cdot 120\pi}E_s, & J &= \frac{L}{H_0}J_s, \end{aligned} \tag{29}$$

where the subindex $\cdot_s$ is used to indicate the values in the SI units, L is the typical length, H_0 is the typical magnetic strength, $c_0 = 1/\sqrt{\mu_0\varepsilon_0} \approx$

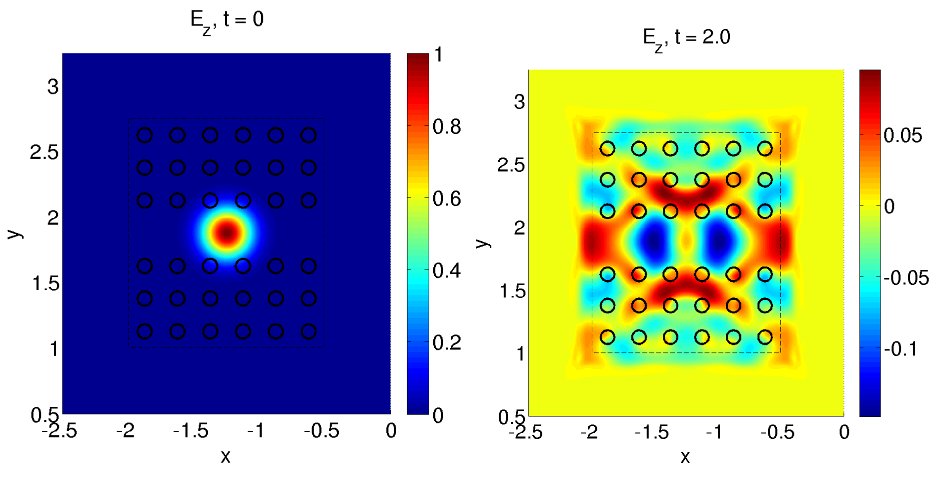


Figure 1: The photonic crystal setup and the E_z field in dimensionless units at $t = 0$ (left) and $t = 2$ (right). The region of interest lies within the dashed line, surrounded by the PML regions.

3×10^8 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 this dimensionless scaling introduces the factor $Z_0 L = 120\pi L$ in the conductivity values. In both tests R1:14
we discretize the problem in space by the standard Yee finite differences.

4.1. Test 1: a photonic crystal model

In this model, we consider a two-dimensional photonic crystal with 6×6 rods, where the top 3 rows of rods (with 6 rods per row) are separated by the bottom 3 rows of rods by a line defect, see Figure 1. Each rod has a radius of 0.055 dimensionless units and consists of a material with relative permittivity $\varepsilon_r = 8.9$ (corresponding to materials such as sapphire or diamond). The grid of rods is surrounded by air with $\varepsilon_r = 1$. In this
test the two-dimensional Maxwell's equations are solved for the x and y R1:5
components of the magnetic field and the z component of electric field (i.e., in the TM mode as it is usually called in the photonic-crystal literature [49]). A Gaussian pulse is emitted at initial time $t = 0$ (see Figure 1) and we follow its evolution until the time moment $T = 2$ dimensionless units.

Around the region of interest $[a_x, b_x] \times [a_y, b_y]$, with $a_x = -2$, $b_x = -0.5$, $a_y = 1$, $b_y = 2.75$, the PMLs are situated. We implemented the PMLs following [50]. The PML conductivity values σ_x and σ_y are zero inside the domain of interest and grow quadratically with the distance to the region of interest to reach the maximum value of 1000. In Figure 2 we show the

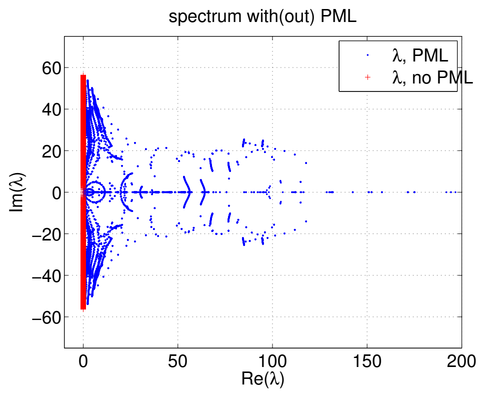


Figure 2: The influence of the PMLs on the spectrum of the Maxwell matrix. The eigenvalues λ are plot on the complex plane.

effect of the PMLs on the spectrum of the Maxwell matrix: the PMLs introduce damping by adding a symmetric part to the matrix so that the matrix spectrum is not purely imaginary anymore. R1:17

Choice of the parameters in the Krylov/SAI algorithm

To have a good performance, it is important that the parameters in the Krylov/SAI algorithm are chosen properly and we now explain how this can be done. Two parameters have to be chosen: the shift value γ, introduced in (25), and the restart value m_{restart}. Restart means that, to avoid storing all the Krylov basis vectors (the columns of V_m), we store only the m_{restart} basis vectors, which are typically the last m_{restart} computed vectors. In this note we use the restarting strategy introduced in [66], for other possible ways to restart see [83, 1, 28, 36, 66, 11].

The standard ways to choose the shift value γ, see e.g. [84], is to assume that the matrix is a discretized elliptic operator, so that the spectrum is more or less evenly spread along the real axis. In our case, the damping is only present in the PML regions, which means that the eigenvalues are mostly situated near the imaginary axis, with a few clusters of outliers along the real axis. Since the bulk of the spectrum lies closely to the imaginary axis, we might try to decrease the shift value γ. Another important point is that the Krylov/SAI, as suggested by the analysis in [63, 84], usually exhibits a mesh independent convergence. In [84], a mesh independent convergence is proved for discretized elliptic operators. This means that we can choose R2:7a
both parameters (namely, the shift value γ and the restart value m_{restart})
cheaply by test runs on a coarse mesh and then use the chosen values for all R2:7b

Table 1: Dependence of the residual and error on the shift parameter γ. A coarse mesh 200×220 is used.

γ/T	0.2	0.1	0.05	0.025	0.012	0.005
aver.residual	0.4788	0.1887	0.0254	0.0190	0.0077	0.9854
rel.error	3.3e-02	7.1e-03	1.4e-03	7.9e-04	4.4e-04	6.0e-01

the meshes.

We first choose the shift value γ. It is usually chosen relatively with respect to the length of the time interval T, owing to the fact that the action of the matrix exponential is computed for the matrix $-TA$. Table 1 contains results of test runs where we vary γ around a suitable value of $0.1T$ ($T = 2$ in this test), as suggested by [84, Table 3.1]. We carry out 100 Krylov steps without restart and judge the convergence by looking at the average residual

$$\sum_{m=91}^{100} \|r_m(T)\|$$

for the last 10 Krylov steps. Averaging is necessary because the residual converges to zero very wiggly; this is typical for wave problems. We see that our expectations are correct: decreasing γ to some extent helps to improve R2:7c convergence. From now on we set γ to $0.012T$ for all runs in this sections. Note that we report the error values in Table 1 only for reference purposes. In practice, the error values are not available, of course, and we choose γ based on the residual.

To choose a proper restart value m_{restart}, we take a coarse mesh 200×220, and carry out several test runs for different, decreasing values of m_{restart}. This means that the Krylov/SAI iterative process, running iterations $m =$ R2:7d $0, 1, 2, \ldots,$ stores and uses only the last m_{restart} Krylov basis vectors (see [66] for details). The smaller m_{restart}, the less computational work and memory resources are required by the method, as fewer Krylov basis vectors have to handled. However, the convergence of the Krylov/SAI method may deteriorate with decreasing m_{restart} and we look for a compromise value for m_{restart}. As plots in Figure 3 demonstrate, the convergence of the Krylov/SAI scheme is quite robust with respect to the restart value and therefore we set $m_{\mathsf{restart}} = 2$ in all remaining tests in this section.

We compare the Krylov/SAI method against the implicit trapezoidal rule (ITR) scheme and the `phiv` function from the EXPOKIT package [78]. The `phiv` function solves, following the first formula in (12) and using the R2a:2 conventional (non SAI) Krylov subspace method, IVP (10). It is instruc-

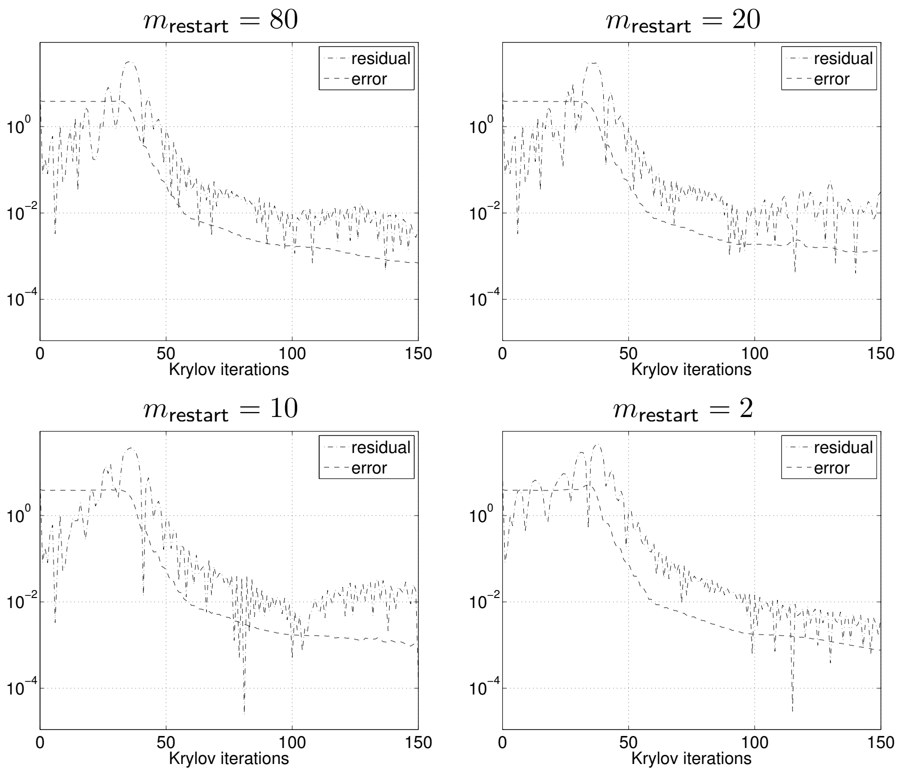


Figure 3: Convergence of the Krylov/SAI method for different restart values m_{restart}. The residual (cf. (28)) and the absolute error norms (y axis) are shown versus the number of Krylov subspace iterations m (x axis). The iterative process is restarted every m_{restart} iterations. A coarse 200×220 mesh is used.

tive to compare our method with these two methods because the ITR is an implicit scheme well suited for Maxwell's equations [90], and it requires efforts to solve the linear systems comparable to those in Krylov/SAI method. Furthermore, the `phiv` scheme is an efficient implementation of the regular Krylov method where, to achieve a better performance, the time interval can be adaptively divided into subintervals (this approach is recently extended in [67]). In the tests reported here EXPOKIT's `phiv` was run with maximal Krylov dimension increased to 100 (the default EXPOKIT's value is 30). Increasing the maximal Krylov subspace dimension in `phiv` is, as far as we know, profitable for the performance of the function and does not influence its stopping criterion [78]. In fact, if the stopping criterion is not satisfied within the maximal allowed Krylov subspace dimension, the code divides the time interval into subintervals and evaluates the matrix function actions on each of the subintervals. R2a:3

In both Krylov/SAI and ITR schemes the linear systems are solved by the UMFPACK sparse LU solver (the `lu` function in Matlab and Octave), which provides the LU factorization as discussed in Section 3.3.

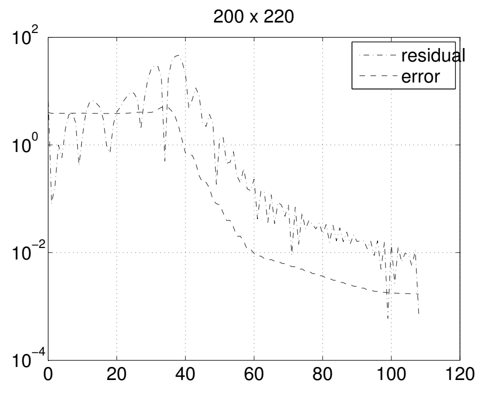
The tests of this section are carried out on a Linux PC with 8 Gb memory and eight CPU cores of 3.40 GHz. We run the Krylov/SAI method with the tolerance value `1e-03` and EXPOKIT with the tolerance `1e-01`. A less stringent tolerance for EXPOKIT is taken because in this test it tends to produce a more accurate solution than required by the tolerance. The results of the test runs are presented in Table 2. The error values reported in the table are relative errors with respect to the reference solution (which was produced by EXPOKIT run with a stringent tolerance `1e-06`). We see that Krylov/SAI outperforms the ITR scheme by approximately a factor of 4 in CPU time and that EXPOKIT is much slower than the other two schemes. For larger problems and in three spatial dimensions the relative cost of linear system solution will be higher in the total costs. This favors the Krylov/SAI method which requires a factor of 8 less linear system solutions (800 ITR time steps with one solution per time step versus 2×54 solutions in Krylov/SAI). Last but not least, the results confirm the mesh independent convergence of the Krylov/SAI method, see also Figure 4. R2:7e R2a:4

4.2. Test 2: electromagnetic imaging

This numerical test comes from the field of electromagnetic imaging and fault detection in gas-and-oil industry [77]. 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

Table 2: Numerical results for the photonic crystal model. The accuracy values are relative error norms with respect to the reference solution. Matvec stands for a matrix–vector multiplication.

problem size (mesh)	method	CPU time, s / accuracy	number of steps / Krylov dimension m
$n = 167\,023$	EXPOKIT	27.1 / `9.8e-05`	1428 matvecs
(200×220)	ITR	9.9 / `7.1e-04`	800 steps
	Krylov/SAI	2.6 / `4.4e-04`	54 restarts with $m_{\mathsf{restart}} = 2$
$n = 665\,243$	EXPOKIT	174 / `6.7e-04`	1938 matvecs
(400×440)	ITR	46 / `7.0e-04`	800 steps
	Krylov/SAI	13.4 / `3.0e-04`	65 restarts with $m_{\mathsf{restart}} = 2$
$n = 1\,494\,663$	EXPOKIT	621 / `1.4e-03`	2652 matvecs
(600×660)	ITR	117 / `6.9e-04`	800 steps
	Krylov/SAI	27 / `3.4e-04`	54 restarts with $m_{\mathsf{restart}} = 2$
$n = 2\,655\,283$	EXPOKIT	1271 / `1.7e-05`	3060 matvecs
(800×880)	ITR	220 / `6.9e-04`	800 steps
	Krylov/SAI	54 / `3.2e-04`	54 restarts with $m_{\mathsf{restart}} = 2$



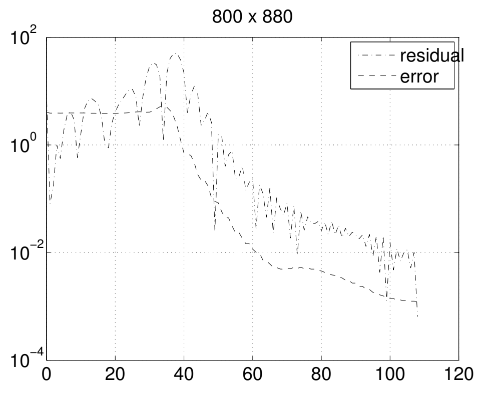


Figure 4: Convergence of the Krylov/SAI method for 200×220 (left) and 800×880 (right) meshes. The residual (cf. (28)) and absolute error norms are shown versus the number of Krylov subspace iterations m. The iterative process is restarted every $m_{\mathsf{restart}} = 2$ iterations.

Table 3: 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 Krylov/SAI, $T = 750$ are given for two different tolerance values `1e-10` and `1e-14`.

scheme	T	# time steps	CPU time, s	rel. error	Krylov dimension
CO2	100	4000	18	4.6e−07	—
Krylov	100	1	$>1\,000$	1.0e−03[a]	restart 300
Krylov/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	—
Krylov/SAI	750	4	7.3	2.7e−05	17,12,5,8
Krylov/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

is defined as

$$\sigma = \begin{cases} 0.1\,\mathrm{S/m}, & x \leqslant 10, \\ 0.001\,\mathrm{S/m}, & x > 10, \end{cases} \tag{30}$$

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 $t = 1.01 \times 10^{-4}\,\mathrm{s}$ and decays linearly to reach its zero value at $t = 1.02 \times 10^{-4}\,\mathrm{s}$. After that the current remains zero until the final time $2.02 \times 10^{-4}\,\mathrm{s}$ is reached.

The tests of this section are carried out in Matlab on a 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 3 and 4. Several methods are compared there: the CO2 scheme (4), the Crank–Nicolson ITR scheme (6), the exponential scheme `phiv.m` of the EXPOKIT package [78], the Krylov and Krylov/SAI methods. The Krylov/SAI method uses a default value of the shift value γ, namely $\gamma = 0.1\tau$, which is suggested by [84, Table 3.1] for

Table 4: 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 Krylov/SAI, $T = 750$ are given for two different tolerance values `1e-10` and `1e-14`.

scheme	T	# time steps	CPU time, s	rel. error	Krylov dimension
CO2	100	8000	130	1.2e−07	—
Krylov	100	1	$> 3\,000$	3.0e−01[a]	restart 300
Krylov/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	—
Krylov/SAI	750	4	203	4.7e−05	17,10,5,8
Krylov/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

moderate accuracy requirements. 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 3 and 4, the regular Krylov method is not efficient. The method has been used in combination with restarting after every 300 Krylov steps. This value is chosen as the largest number of Krylov basis R2:7f
vectors to store and handle which is still practical for this problem size. Taking a restart value different than 300 does not help at all (smaller values are harmful for convergence, bigger values increase computational work and CPU time).

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

Table 5: 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). 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 5. For this reason we use the Krylov/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 Krylov/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 [84] and confirmed in our experiments, the Krylov/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 Krylov/SAI method combined with an efficient preconditioned iterative solver, if one was available. However, standard preconditioners, such as incomplete LU with dropping tolerance, appear not to be efficient for this problem. We note that the block structure of the discretized Maxwell operator should be used when constructing a preconditioner, see [90] or [13, Section 5.3] for possible inexact Schur complement preconditioners in the context of time dependent Maxwell's equations.[4]

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

[4]Replacing the direct sparse UMFPACK solver in the Krylov/SAI method by GMRES with the inexact Schur complement preconditioner of [13], we are able to get the same results in terms of accuracy and number (outer) Krylov iterations. However, at least when measured in Matlab, the CPU needed by the preconditioned iterative solver is much higher than required by the direct solver. This may be caused by the Matlab environment and we plan to investigate this further in future.

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 Krylov/SAI due many more time steps needed.

This numerical test is 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 Krylov/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 [90]. 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 [10].

In the tests reported in Tables 3 and 4 EXPOKIT's `phiv` is run with maximal Krylov dimension increased to 100 (the default EXPOKIT's value is 30).

Remark on the computational complexity

It is in general difficult to provide a detailed picture for the computational complexity of the Krylov/SAI method, as it depends on many problem dependent factors. However, one conclusion can be drawn from the both numerical tests (Sections 4.1 and 4.2) presented here. If a direct solver for the SAI systems with $I + \gamma A$ is affordable, it is sensible to compare the Krylov/SAI method with an implicit scheme such as ITR.

For the ITR or other implicit schemes, one can estimate a number of R2a:6
time steps required to reach the desirable accuracy and, as we can observe from the results of the experiments, we typically need hundreds of time steps with an implicit scheme to get a reasonable accuracy. At each time step (at least) one linear system with the matrix $I + \tau A$ or alike should be solved. With the Krylov/SAI method, we need a couple of dozen actions of $(I + \gamma A)^{-1}$ but, compared to an implicit scheme, there is an additional overhead such as the Arnoldi process. This comparison holds as soon as R2a:6
the costs for solving the systems with the matrices $I + \tau A$ and $I + \gamma A$ are comparable, which may not be the case if $\gamma \not\approx \tau$ (cf. Table 5).

When direct solutions of the SAI systems are not affordable, one may use a preconditioned iterative solver in both the implicit scheme and the Krylov/SAI method. In this case, a lot depends on whether a good preconditioner is available. However, it is not trivial to properly combine an implicit scheme with an iterative solver such that the accuracy and stability properties of the scheme are maintained. Indeed, too many linear solver iterations per time steps would be a waste of computational efforts, whereas too few iterations might mean that the system is not solved accurately enough to preserve the stability properties of the implicit scheme [14]. On the other hand, for the Krylov/SAI method there is a developed theory on how to solve the SAI systems approximately [79, 85, 84]. R2a:7

5. Conclusions

We have presented recent numerical techniques for computing the action of the matrix exponential of a large matrix in the context of time dependent Maxwell's equations. The Krylov subspace 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 problems. In particular, the observed (i) mesh independent convergence and (ii) robustness with respect to the restart values make the method very promising. We have shown how the parameters in the method can be chosen in a simple way. R1:18

Due to the more work per time step than in conventional schemes, the exponential methods appear to be efficient for sufficiently large time steps and when the numerical linear algebra overhead is minimized. This can for instance be done by restarting the Krylov basis (as in the first numerical test) or by reusing the computed sparse LU factorization (as in the second numerical test).

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 our plan to concentrate a further research on development of (parallel) preconditioned iterative solvers, possibly including the recent developments in algebraic multigrid methods (as, e.g., [68]) and the so-called Krylov subspace recycling techniques [69]. R2a:8

The Matlab implementations `expm_Arnoldi` and `expm_ArnoldiSAI` of the algorithms described in Sections 3.1 and 3.3 can be downloaded from [9].

Acknowledgments

The author would like to thank anonymous referees for helpful suggestions to improve the paper and Oleg Nechaev for his kind advise concerning the implementation of the second 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] S. Benhassine, W. P. Carpes Jr., and L. Pichon. Comparison of mass lumping techniques for solving the 3D Maxwells equations in the time domain. *IEEE Trans. Magnet.*, 36:1548–1552, 2000.

[5] 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/`.

[6] 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`.

[7] R.-U. Börner, O. G. Ernst, and S. Güttel. Three-dimensional transient electromagnetic modeling using rational Krylov methods. The University of Manchester, MIMS EPrint 2014.36, 2014. `http://eprints.ma.man.ac.uk/2161/`.

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

[9] M. A. Botchev. expmARPACK: matrix exponential actions with Arnoldi methods. Octave / Matlab software package, version 1.0, 2012. `www.math.utwente.nl/~botchevma/expm/`.

[10] M. A. Botchev. A block Krylov subspace time-exact solution method for linear ordinary differential equation systems. *Numer. Linear Algebra Appl.*, 20(4):557–574, 2013.

[11] M. A. Botchev, V. Grimm, and M. Hochbruck. Residual, restarting and Richardson iteration for the matrix exponential. *SIAM J. Sci. Comput.*, 35(3):A1376–A1397, 2013.

[12] 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`.

[13] M. A. Botchev, I. V. Oseledets, and E. E. Tyrtyshnikov. Iterative across-time solution of linear differential equations: Krylov subspace versus waveform relaxation. *Computers & Mathematics with Applications*, 67(12):2088–2098, 2014. `http://dx.doi.org/10.1016/j.camwa.2014.03.002`.

[14] M. A. Botchev, G. L. G. Sleijpen, and H. A. van der Vorst. Stability control for approximate implicit time stepping schemes with minimum residual iterations. *Appl. Numer. Math.*, 31(3):239–253, 1999.

[15] 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`.

[16] 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.

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

[18] E. Celledoni and I. Moret. A Krylov projection method for systems of ODEs. *Appl. Numer. Math.*, 24(2-3):365–378, 1997.

[19] 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.

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

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

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

[23] 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.

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

[25] 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.

[26] 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.

[27] 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, 1998.

[28] M. Eiermann, O. G. Ernst, and S. Güttel. Deflated restarting for matrix functions. *SIAM J. Matrix Anal. Appl.*, 32(2):621–641, 2011.

[29] A. Fisher, R. N. Rieban, G. H. Rodrigue, and D. A. White. A generalized mass lumping technique for vector finite-element solutions of the time-dependent Maxwell equations. *IEEE Transections on Antennas and Propagation*, 53(9):2900–2910, 2005.

[30] 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.

[31] 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.

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

[33] 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.

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

[35] M. J. Grote and T. Mitkova. Explicit local time-stepping for Maxwell's equations. *Journal of Computational and Applied Mathematics*, 234(12):3283–3302, 2010.

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

[37] S. Güttel. Rational Krylov approximation of matrix functions: Numerical methods and optimal pole selection. *GAMM Mitteilungen*, 36(1):8–31, 2013. `www.guettel.com`.

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

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

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

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

[42] D. Hipp, M. Hochbruck, and A. Ostermann. An exponential integrator for non-autonomous parabolic problems. *ETNA*, 41:497–511, 2014. `http://etna.mcs.kent.edu/volumes/2011-2020/vol41/`.

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

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

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

[46] 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`.

[47] M. Hochbruck, T. Pažur, A. Schulz, E. Thawinan, and C. Wieners. Efficient time integration for discontinuous Galerkin approximations of linear wave equations. *ZAMM*, 2014. online first, `http://dx.doi.org/10.1002/zamm.201300306`.

[48] 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.

[49] J. D. Joannopoulos, S. G. Johnson, J. N. Winn, and R. D. Meade. *Photonic Crystals. Molding the Flow of Light.* Princeton University Press, 2 edition, 2008. Available at `http://ab-initio.mit.edu/book/`.

[50] S. G. Johnson. Notes on perfectly matched layers (PMLs). `math.mit.edu/~stevenj/18.369/pml.pdf`, March 2010.

[51] L. Knizhnerman and V. Simoncini. A new investigation of the extended Krylov subspace method for matrix function evaluations. *Numer. Linear Algebra Appl.*, 17(4):615–638, 2010.

[52] 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.

[53] J. S. Kole, M. T. Figge, and H. De Raedt. Unconditionally stable algorithms to solve the time-dependent Maxwell equations. *Phys. Rev. E*, 64:066705, 2001.

[54] J. S. Kole, M. T. Figge, and H. De Raedt. Higher-order unconditionally stable algorithms to solve the time-dependent Maxwell equations. *Phys. Rev. E*, 65:066705, 2002.

[55] M. Kolle. *Photonic Structures Inspired by Nature.* Springer, 2011.

[56] V. I. Lebedev. *How to solve stiff systems of differential equations by explicit methods*, pages 45–80. CRC Press, Boca Raton, 1994.

[57] 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.

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

[59] 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.

[60] 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`.

[61] 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.

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

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

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

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

[66] 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.

[67] 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.

[68] Y. Notay and AGMG team. AGMG: Iterative solution with AGgregation-based algebraic MultiGrid. Software package, `http://homepages.ulb.ac.be/~ynotay/AGMG/`.

[69] 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.

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

[71] 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.

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

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

[74] F. Schenk, B. D. Wilts, and D. G. Stavenga. The japanese jewel beetle: a painter's challenge. *Bioinspir. Biomim.*, 8:045002 (10pp), 2013. `http://dx.doi.org/10.1088/1748-3182/8/4/045002`.

[75] 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.

[76] 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.

[77] 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. English summary: `www.ict.nsc.ru/jct/annotation/346?l=eng`.

[78] 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/`.

[79] V. Simoncini and D. B. Szyld. Theory of inexact Krylov subspace methods and applications to scientific computing. *SIAM J. on Sci. Comput.*, 25(2):454–477, 2003. `http://dx.doi.org/10.1137/S1064827502406415`.

[80] A. Taflove and S. C. Hagness. *Computational electrodynamics: the finite-difference time-domain method*. Artech House Inc., Boston, MA, third edition, 2005.

[81] H. Tal-Ezer. Spectral methods in time for hyperbolic equations. *SIAM J. Numer. Anal.*, 23(1):11–26, 1986. `http://dx.doi.org/10.1137/0723002`.

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

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

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

[85] J. van den Eshof and G. L. G. Sleijpen. Inexact Krylov subspace methods for linear systems. *SIAM J. Matrix Anal. Appl.*, 26(1):125–153, 2004.

[86] 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.

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

[88] S. Vandekerckhove, B. Vandewoestyne, H. De Gersem, K. Van Den Abeele, and S. Vandewalle. Mimetic discretization and higher order time integration for acoustic, electromagnetic and elastodynamic wave propagation. *Journal of Computational and Applied Mathematics*, 259, Part A(0):65–76, 2014. `http://dx.doi.org/10.1016/j.cam.2013.02.027`.

[89] J. G. Verwer. Composition methods, Maxwell's equations, and source terms. *SIAM Journal on Numerical Analysis*, 50(2):439–457, 2012. `http://dx.doi.org/10.1137/100816122`.

[90] 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.

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

[92] 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.