Efficient convolution based impedance boundary condition

A. Paganini and M. López-Fernández

Research Report No. 2013-07

Seminar für Angewandte Mathematik
Eidgenössische Technische Hochschule
CH-8092 Zürich
Switzerland

EFFICIENT CONVOLUTION BASED IMPEDANCE BOUNDARY CONDITION

ALBERTO PAGANINI[*] AND MARÍA LÓPEZ-FERNÁNDEZ[†]

Abstract. We consider an eddy current problem in time-domain and rely on impedance boundary conditions on the surface of the conductor(s). We assume additional translational symmetries and, with a "method of lines policy" in mind, we pursue a semi-discretization in space by a finite element Ritz-Galerkin discretization. The resulting set of Volterra integral equation in time is discretized by means of *Runge-Kutta convolution quadrature* (CQ) focusing on fast and oblivious implementations. The final algorithm is validated by several numerical experiments.

Key words. eddy current problem, impedance boundary conditions, convolution quadrature, fast and oblivious algorithms

1. Introduction.

Due to the skin effect, alternating electromagnetic fields decay exponentially when penetraiting a good conductor. In transient eddy current problems this property can be exploited by modelling the conductor with the well-known Leontovich boundary condition (Senior 1960, De Santis, Cruciani, Feliziani and Okoniewski 2012). In frequency domain, lowest order impedance boundary condition involves a multiplication of the fields with a frequency dependent term. This multiplication becomes a convolution in time domain, when harmonic oscillations of the fields can not be assumed.

Since convolution is generically non-local in time, developing a stable and memory efficient discretization becomes a challenge. In (Oh and Schutt-Aine 1995) this issue has been tackled in the context of FDTD methods. There the Laplace transform of the convolution kernel is approximated via a truncated series expansion and then an approximated impedance boundary condition in time domain is derived.

As an alternative C. Lubich developed the so-called Convolution Quadrature method (CQ) in (Lubich 1988*a*, Lubich 1988*b*, Lubich and Ostermann 1993). It requires only knowledge of the Laplace transform $K(s)$ of the convolution kernel $k(t)$ and enjoys excellent convergence and stability properties both for computing convolutions and solving Volterra convolution equations (Banjai, Lubich and Melenk 2011). Finally a fast "oblivious" algorithm for approximate CQ (FOCQ) with considerably reduced memory requirements was presented in (Schädle, López-Fernández and Lubich 2006).

In this paper we demonstrate how the FOCQ can be applied for the efficient temporal discretization of eddy current problems that involve impedance boundary conditions. Discretization in space relies on finite elements. The eventual scheme inherits the algebraic convergence in timestep size and meshwidth, respectively, of both discretizations. Thanks to FOCQ it is unconditionally stable and the computational cost scales almost linearly with the number of timesteps.

2. Eddy Current model.

We consider a linear transient eddy current problem with a conductor occupying the bounded and connected polyhedron $\Omega_C \subset \mathbb{R}^3$. In frequency domain at fixed angular frequency $\omega > 0$ the Leontovich boundary condition reads (Senior 1960,

[*]SEMINAR FOR APPLIED MATHEMATICS, SWISS FEDERAL INSTITUTE OF TECHNOLOGY, ZURICH, SWITZERLAND, EMAIL: ALBERTO.PAGANINI@SAM.MATH.ETHZ.CH . PARTLY SUPPORTED BY ETH GRANT CH1-02 11-1

[†]INSTITUTE OF MATHEMATICS, UNIVERSITY OF ZURICH, ZURICH, SWITZERLAND. PARTIALLY SUPPORTED BY GRANTS MTM 2010-19510 AND MTM2012-31298, FROM MINISTERIO DE ECONOMIA Y COMPETITIVIDAD, SPAIN.

De Santis et al. 2012)

$$(\widehat{\mathbf{H}} \times \boldsymbol{n})(\boldsymbol{x}) = \sqrt{\frac{i\omega\sigma(\boldsymbol{x})}{\mu(\boldsymbol{x})}}\widehat{\mathbf{E}}_t(\boldsymbol{x}) \ , \quad \boldsymbol{x} \in \Gamma \ , \tag{2.1}$$

where $\boldsymbol{n} : \Gamma \to \mathbb{R}^3$ is the exterior unit normal vector field on the conductor surface Γ, and $\widehat{\mathbf{H}}$ and $\widehat{\mathbf{E}}$ denote the complex amplitudes of the magnetic and electric field, respectively, and $\widehat{\mathbf{E}}_t := (\boldsymbol{n} \times \widehat{\mathbf{E}}) \times \boldsymbol{n}$ is the tangential component. The material coefficients μ (magnetic permeability) and σ (conductivity) are uniformly positive, but may vary in space.

Assuming that all fields vanish for $t \leq 0$, from (2.1) we derive the following *transient impedance boundary condition* for the time-dependent fields

$$\mathbf{H}(\boldsymbol{x}, t) \times \boldsymbol{n}(\boldsymbol{x}) = \int_0^t \eta(\boldsymbol{x}) k(t-\tau) \mathbf{E}_t(\boldsymbol{x}, \tau) \, \mathrm{d}\tau \ , \quad t \geq 0 \ , \quad \boldsymbol{x} \in \Gamma \ , \tag{2.2}$$

with a uniformly positive function $\eta(\boldsymbol{x}) := \sqrt{\sigma(\boldsymbol{x})\mu(\boldsymbol{x})^{-1}}$, $\boldsymbol{x} \in \Gamma$, and a *convolution kernel* $k : \Gamma \times \mathbb{R}^+ \to \mathbb{R}$, whose temporal Laplace transform is given by

$$K(s) := (\mathcal{L}k(\cdot))(s) = \sqrt{s} \ , \quad s \in \mathbb{C} \setminus (-\infty, 0) \ . \tag{2.3}$$

For the sake of brevity we adopt the "operational calculus notation" for (2.2) (Lubich 1988*a*), expressing it as $\mathbf{H} \times \boldsymbol{n} = \eta K(\partial_t)\mathbf{E}_t$.

With a finite element discretization in mind we artificially truncate the fields to a simple bounded computational domain $\Omega \subset \mathbb{R}^3$ with $\overline{\Omega}_C \subset \Omega$. Then, the evolution of the (scaled) electromagnetic fields in $D := \Omega \setminus \Omega_C$ is governed by the following initial-boundary value problem that we consider up to a fixed final time $T > 0$:

$$\begin{aligned}
&\mathbf{curl}\,\mathbf{curl}\,\mathbf{E} = \mathbf{j}(\boldsymbol{x}, t) \quad , \quad \operatorname{div} \mathbf{E} = 0 && \text{in } D \times]0, T[\ , && (2.4a)\\
&\mathbf{curl}\,\mathbf{E} \times \boldsymbol{n} = \eta(\boldsymbol{x}) K(\partial_t) \mathbf{E}_t && \text{on } \Gamma \times]0, T[\ , && (2.4b)\\
&\mathbf{curl}\,\mathbf{E} \times \boldsymbol{n} = 0 && \text{on } \partial\Omega \times]0, T[\ , && (2.4c)\\
&\mathbf{E}(\cdot, 0) = 0 && \text{on } D \ . && (2.4d)
\end{aligned}$$

This is the so-called $\mathbf{E}$-based formulation of an eddy current problem (Alonso-Rodriguez and Valli 2010, Sect. 2.1). The zero divergence condition on $\mathbf{E}$ in (2.4a) should be regarded as a *gauging*, which ensures uniqueness of the electric field inside D. The right hand side $\mathbf{j}$ stands for a solenoidal source current supported inside D that engenders an exciting magnetic field.

Remark. In the case of translational symmetry we end up with the so-called TM/TE eddy current models that give rise to boundary value problems for a single scalar unknown, e.g.,

$$\begin{aligned}
&-\Delta u = f && \text{in } \widetilde{D} \times]0, T[\ , && (2.5a)\\
&\mathbf{grad}\, u \cdot \widetilde{\boldsymbol{n}} = \eta(\widetilde{\boldsymbol{x}}) K(\partial_t) u && \text{on } \widetilde{\Gamma} \times]0, T[\ , && (2.5b)\\
&u = 0 && \text{on } \partial\widetilde{\Omega} \times]0, T[\ , && (2.5c)\\
&u(\cdot, 0) = 0 && \text{on } \widetilde{D} \ , && (2.5d)
\end{aligned}$$

where the scalar unknown $u = u(\widetilde{\boldsymbol{x}}, t)$ represents a single component of the electric field and the ˜ tags two-dimensional cross-sections of the domains/boundaries. In Section 7 we report numerical results for this dimensionally reduced model.

3. Spatial discretization. Adopting the “method of lines policy” an approximation of (2.4) is obtained by discretizing in space first, followed by a suitable discretization in time. We rely on a finite element Galerkin discretization in space, starting from the variational formulation: seek $\mathbf{E} = \mathbf{E}(t) \in \mathbf{H}(\mathbf{curl}, D)$ and a “dummy potential” $V \in H^1_\Gamma(D) := \{v \in H^1(D) :\ v_{|\Gamma} = 0\}$, such that

$$\begin{array}{lclclcl} \int\limits_D \mathbf{curl}\,\mathbf{E}\cdot\mathbf{curl}\,\mathbf{E}'\,\mathrm{d}\boldsymbol{x} & + & \int\limits_D \mathbf{E}'\cdot\mathbf{grad}\,V\,\mathrm{d}\boldsymbol{x} & + & K(\partial_t)\int\limits_\Gamma \eta\,\mathbf{E}_t\cdot\mathbf{E}'_t\,\mathrm{d}S & = & \int\limits_D \mathbf{j}\cdot\mathbf{E}'\,\mathrm{d}\boldsymbol{x}\ , \\ \int\limits_D \mathbf{E}\cdot\mathbf{grad}\,V'\,\mathrm{d}\boldsymbol{x} & - & \int\limits_D V\,V'\,\mathrm{d}\boldsymbol{x} & & & = & 0\ , \end{array} \tag{3.1}$$

for all $\mathbf{E}' \in \mathbf{H}(\mathbf{curl}, D)$, $V' \in H^1_\Gamma(D)$. Note that a priori $V = 0$ is known as $\operatorname{div}\mathbf{j} = 0$.

We equip D with a (tetrahedral and hexahedral) finite element mesh, approximate $\mathbf{E}$ by means of lowest order edge elements (Hiptmair 2002, Section 3.2), and V by means of piecewise (bi-)linear continuous functions. Using the standard locally supported basis functions for these finite element spaces along with mass lumping for the L^2-inner product occurring in (3.1), we end up with the linear evolution problem

$$\begin{array}{lclclcl} \mathbf{C}\,\boldsymbol{\mu}(t) & + & \mathbf{G}^T\,\boldsymbol{\psi}(t) & + & K(\partial_t)\mathbf{B}\,\boldsymbol{\mu}(t) & = & \boldsymbol{\varphi}(t)\ , \\ \mathbf{G}\,\boldsymbol{\mu}(t) & - & \mathbf{D}\,\boldsymbol{\psi}(t) & & & = & 0\ . \end{array} \tag{3.2}$$

Here $\boldsymbol{\mu}(t)$, $\boldsymbol{\psi}(t)$ are the time-dependent basis coefficient vectors for the approximations of $\mathbf{E}$ and V, respectively, and

$$\boldsymbol{\varphi}(t) := \left(\int_D \mathbf{j}(t)\cdot\boldsymbol{\phi}_h^1\,\mathrm{d}\boldsymbol{x}, \dots, \int_D \mathbf{j}(t)\cdot\boldsymbol{\phi}_h^M\,\mathrm{d}\boldsymbol{x}\right)^T\ ,$$

with $\boldsymbol{\phi}_h^i$, $i = 1, \dots, M$, denoting the basis of the edge element space. The matrices $\mathbf{C}$, $\mathbf{G}$, $\mathbf{B}$, and $\mathbf{D}$ are the sparse Galerkin matrices arising from the various bilinear forms in (3.1), where $\mathbf{D}$ is diagonal thanks to mass lumping (Hiptmair 2002, Section 6.1).

Thus an elimination of $\psi(t)$ becomes feasible and we end up with the Volterra integral equation in $\mathbb{R}^M$

$$\underbrace{(\mathbf{C} + \mathbf{G}^T\mathbf{D}^{-1}\mathbf{G})}_{=:\mathbf{A}}\,\boldsymbol{\mu}(t) + K(\partial_t)\mathbf{B}\,\boldsymbol{\mu}(t) = \boldsymbol{\varphi}(t)\ , \tag{3.3}$$

with $\mathbf{A} \in \mathbb{R}^{M,M}$ symmetric positive definite, and symmetric positive semi-definite $\mathbf{B} \in \mathbb{R}^{M,M}$. In particular both matrices are sparse and $\mathbf{B}$ acts only on the boundary degrees of freedom.

Remark. Spatial discretization of (2.5) is easier: testing with $H^1_{\partial\widetilde{\Omega}}(\widetilde{D})$ functions, the variational formulation of (2.5) reads

$$\int_{\widetilde{D}} \mathbf{grad}\,u\cdot\mathbf{grad}\,v\,\mathrm{d}\boldsymbol{x} + K(\partial_t)\int_{\widetilde{\Gamma}} \eta\,u\,v\,\mathrm{d}S = \int_{\widetilde{D}} f\,v\,\mathrm{d}\boldsymbol{x} \quad \text{for all } v \in H^1_{\partial\widetilde{\Omega}}(\widetilde{D})\ . \tag{3.4}$$

A Ritz-Galerkin discretization of (3.4) by piecewise linear Lagrangian finite elements leads to a system of integral equations

$$\mathbf{A}\,\boldsymbol{\mu}(t) + K(\partial_t)\mathbf{B}\,\boldsymbol{\mu}(t) = \boldsymbol{\varphi}(t) \qquad \text{for } t \in]0, T[\ , \tag{3.5}$$

where the vector $\boldsymbol{\mu}(t) \in \mathbb{R}^M$ contains the time-dependent coefficients of an approximation in space of u with respect to the finite element basis.

4. Convolution quadrature (CQ). The Convolution Quadrature (CQ) is based on Runge-Kutta methods and approximates the continuous convolution

$$K(\partial_t)g := \int_0^T k(t-\tau)g(\tau)\, d\tau \tag{4.1}$$

at the time $T = (N+1)\Delta t$ with the last component of the discrete convolution vector

$$\left(K(\underline{\partial_{\Delta t}})\mathbf{g}\right)^{(N)} := \sum_{j=0}^{N} \mathbf{W}_{N-j}\mathbf{g}_j \ , \tag{4.2}$$

where the entries of the vector

$$\mathbf{g}_j := (g(t_j + c_1\Delta t), \ldots, g(t_j + c_m\Delta t))^T \in \mathbb{R}^m \tag{4.3}$$

are the values of the function g at the Runge–Kutta internal times at $t_j := j\Delta t$.

In this article we consider the family of m-stage RadauIIA methods (Hairer and Wanner 2010, Chapter IV.5) as underlying Runge-Kutta method because it has become the standard choice for CQ.

The CQ requires only the knowledge of the Laplace Transform $K(s)$ of the convolution kernel $k(t)$, which is assumed to be analytic in the sector

$$\Sigma(\varphi,\sigma) := \{s \in \mathbb{C} :\ |\arg(s-\sigma)| < \pi - \varphi, \quad \text{with } \varphi < \frac{1}{2}\pi \text{ and } \sigma \geq 0\} \ . \tag{4.4}$$

The convolution weights $\mathbf{W}_n \in \mathbb{R}^{m,m}$ are defined by the power series expansion (Lubich and Ostermann 1993, Section 2)

$$\sum_{n=0}^{\infty} \mathbf{W}_n \zeta^n := K\left(\frac{\boldsymbol{\Delta}(\zeta)}{\Delta t}\right) \ , \quad \boldsymbol{\Delta}(\zeta) := \left(\mathcal{O}\!\!\!\!\mathcal{Q} + \frac{\zeta}{1-\zeta}\mathbb{1}\mathbf{b}^T\right)^{-1} \ , \tag{4.5}$$

where $\mathcal{O}\!\!\!\!\mathcal{Q}$ is the Runge–Kutta coefficient matrix, $\mathbf{b}^T$ is the Runge-Kutta weight vector and $\mathbb{1} = (1,\ldots,1)^T$.

The convolution weights can be approximated by discretizing the Cauchy integral

$$\begin{aligned} \mathbf{W}_n &= \frac{1}{2\pi i}\int_{|\zeta|=\rho} \zeta^{-1-n} K\left(\frac{\boldsymbol{\Delta}(\zeta)}{\Delta t}\right) d\zeta \ , \\ &\approx \frac{\rho^{-n}}{L}\sum_{\ell=0}^{L-1} K\left(\frac{\boldsymbol{\Delta}(\rho e^{2\pi i\ell/L})}{\Delta t}\right) e^{-2\pi i n\ell/L} \ , \end{aligned} \tag{4.6}$$

with $0 < \rho < 1$ (Lubich and Ostermann 1993, Section 2). An error of magnitude $\mathcal{O}(\sqrt{\varepsilon})$, where ε stands for the machine precision, can be easily achieved by choosing $\rho \approx \sqrt[2N]{\varepsilon}$ and the number of quadrature poins $L = N$ (Lubich 1988*b*, Section 7).

The CQ is also efficient in solving Volterra convolution equations (Banjai et al. 2011). A discretization of (3.3) is achieved by "evaluating" the coefficient vector $\boldsymbol{\mu}(t)$ at the Runge–Kutta internal times and by introducing the vectors

$$\tilde{\boldsymbol{\mu}}_i \approx (\boldsymbol{\mu}(t_i + c_1\Delta t), \cdots, \boldsymbol{\mu}(t_i + c_m\Delta t))^T \in \mathbb{R}^{mM} \ .$$

Replacing the continuous convolution with Runge–Kutta CQ turns (3.3) into the linear implicit scheme

$$(\mathbf{I}_m \otimes \mathbf{A})\ \tilde{\boldsymbol{\mu}}_i + \sum_{j=0}^{i} (\mathbf{W}_{i-j} \otimes \mathbf{B})\, \tilde{\boldsymbol{\mu}}_j = \tilde{\boldsymbol{\varphi}}_i \quad \text{for } i = 0,..,N \ , \tag{4.7}$$

where $\otimes$ is the Kronecker product, $\mathbf{I}_m \in \mathbb{R}^{m,m}$ is the identity matrix and

$$\tilde{\boldsymbol{\varphi}}_i := (\boldsymbol{\varphi}(t_i + c_1\Delta t), \dots, \boldsymbol{\varphi}(t_i + c_m\Delta t))^T \ .$$

The solution vectors $\tilde{\boldsymbol{\mu}}_i$ are then recursively given by

$$(\mathbf{I}_m \otimes \mathbf{A} + \mathbf{W}_0 \otimes \mathbf{B})\, \tilde{\boldsymbol{\mu}}_i = \tilde{\boldsymbol{\varphi}}_i - \sum_{j=0}^{i-1} (\mathbf{W}_{i-j} \otimes \mathbf{B})\, \tilde{\boldsymbol{\mu}}_j \ , \tag{4.8}$$

with $i = 0, \dots, N$. The last M entries of $\tilde{\boldsymbol{\mu}}_N$ are an approximation in time of algebraic order $\min(2m-1, m+1)$ of the exact solution $\boldsymbol{\mu}(T)$, see (Hiptmair, Paganini and López-Férnandez 2013, Lemma 4.1).

Note that at the iteration time i all terms in the right handside of (4.8) are explicitely known. Note also that the indices of the summands shift as the iteration time increases. Hence to recover an approximation of (3.3) at the final time $T = (N+1)\Delta t$ with a naive implementation all the vectors $\tilde{\boldsymbol{\mu}}_i$ must be stored and $\mathcal{O}(N^2)$ multiplications must be computed.

5. Fast and oblivious convolution quadrature (FOCQ).

A fast and oblivious Runge–Kutta based CQ (FOCQ) has been developed in (Schädle et al. 2006). Its key ingredient is the integral representation

$$\mathbf{W}_n = \frac{h}{2\pi i} \int_\gamma K(\lambda)\mathbf{E}_n(h\lambda)\, d\lambda \tag{5.1}$$

of the convolution weights. The contour γ has increasing imaginary part and lies in the analyticy region of K (see Figure 5, left) and to the left of the poles of the matrix function

$$\mathbf{E}_n(z) := R(z)^{n-1}(\mathbf{I}_m - z\mathcal{O})^{-1}\mathbb{1}\mathbf{b}^T(\mathbf{I}_m - z\mathcal{O})^{-1} \ .$$

Here

$$R(z) := 1 + z\mathbf{b}^T(\mathbf{I}_m - z\mathcal{O})^{-1}\mathbb{1}$$

denotes the stability function of the underlying Runge–Kutta method and $\mathbf{I}_m \in \mathbb{R}^{m,m}$ is the identity matrix.

Choosing a suitable parametrization of γ, the contour representation (5.1) can be approximated by means of the composite trapezoidal rule with an exponentially small error when $n > n_0$[1], see (López-Fernández, Lubich, Palencia and Schädle 2005, Theorem 3). Although the optimal choice of the contour depends on the index n, the same contour can be used for a range of convolution weights $\mathbf{W}_n$ whose indices n belong to a geometrically growing interval of the form $[B^{\ell-1}, B^\ell]$, for some prescribed ratio $B > 1$ and . Thus, with a clever choice of the parameters, few contours Γ_ℓ are enough for computing all the convolution weights with a target accuracy ε using a the same number of quadrature nodes on all contours (see Figure 5.2).

In practice the contours are parametrized as the left branch of a hyperbola

$$\mathbb{R} \to \Gamma \ : x \mapsto \gamma(x) := \mu\,(1 - \sin(\alpha + ix)) + \sigma \ , \tag{5.2}$$

whose parameters $\mu > 0$ and $0 < \alpha < \frac{\pi}{2} - \varphi$ are chosen accordingly to (López-Fernández, Palencia and Schädle 2006, Section 4) and φ and σ from (4.4) (see Figure 5).

[1]The approximation is poor for the the first few $\mathbf{W}_n$ which will be thus computed with (4.6).

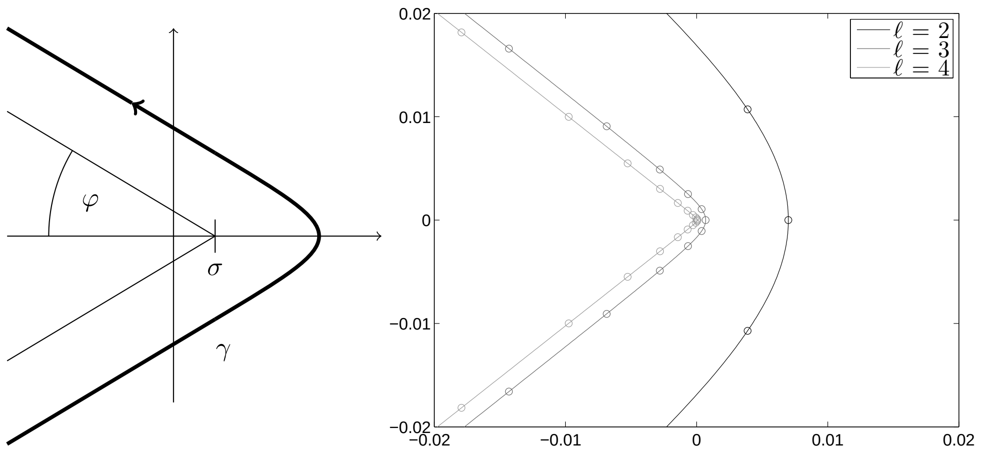

FIG. 5.1. Left: *Contour γ for the convolution weight representation formula* (5.1). Right: *Particular of truncated left branch of hyperbolae for different contours with $B = 10$, $N_Q = 10$ and $\Delta t = 0.25$. The circles indicates the position of the quadrature nodes. Note that as ℓ increases the nodes become closer.*

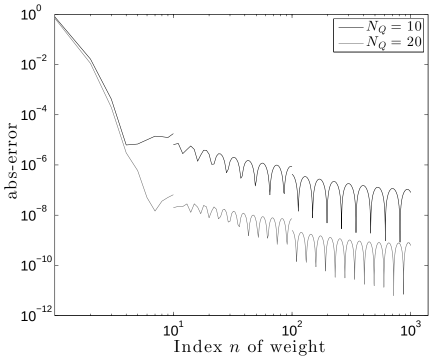

FIG. 5.2. *Convolution weight approximation error in the Euclidean norm of* (5.1) *for the* 2*-stage RadauIIA method with γ as in* (5.2) *by composite trapezoidal rule discretization with N_Q quadrature nodes. With different contours and a clever choice of the contour parameters the error can be made uniformly small for $n > n_0$.*

By introducing a strongly monotone decreasing sequence $(b_\ell)_{\ell=0}^{\lceil \log_B n \rceil}$ [2] of real values so that $b_0 = n$, $b_{\lceil \log_B n \rceil} = 0$ and $n - j \in [B^{\ell-1}, B^\ell]$ for $j \in [b_\ell, b_{\ell-1} - 1]$, the convolution quadrature (4.2) can be rearranged in

$$\left(K(\underline{\partial_{\Delta t}})\mathbf{g}\right)^{(n)} = \mathbf{W}_0 \mathbf{g}_n + \sum_{\ell=1}^{\lceil \log_B n \rceil} \mathbf{U}_n^{(\ell)} \ , \tag{5.3}$$

where

$$\mathbf{U}_n^{(\ell)} := \sum_{j=b_\ell}^{b_{\ell-1}-1} \mathbf{W}_{n-j}\mathbf{g}_j = \sum_{j=b_\ell}^{b_{\ell-1}-1} \frac{\Delta t}{2\pi i} \int_{\Gamma_\ell} K(\lambda)\mathbf{E}_{n-j}(\Delta t \lambda)\mathbf{g}_j \, d\lambda \ .$$

[2]By $\lceil \log_B n \rceil$ we denote the smallest integer not less than $\log_B n$. An explicit pseudo–code for computing this sequence is given in (Schädle et al. 2006, Section 4.1)

Note that for each $\mathbf{U}_n^{(\ell)}$ we choose a suitable contour Γ_ℓ. By sorting the sum and defining

$$y(b_{\ell-1}, b_\ell, \lambda) := \sum_{j=b_\ell}^{b_{\ell-1}-1} \Delta t R(\Delta t\lambda)^{(b_{\ell-1}-1)-j} \mathbf{b}^T (\mathbf{I}_m - \Delta t\lambda \mathcal{O})^{-1} \mathbf{g}_j \ , \tag{5.4}$$

we have

$$\begin{aligned} \mathbf{U}_n^{(\ell)} &= \frac{1}{2\pi i}\int_{\Gamma_\ell} K(\lambda) R(\Delta t\lambda)^{n-b_{\ell-1}} (\mathbf{I}_m - \Delta t\lambda\mathcal{O})^{-1} \mathbb{1} y(b_{\ell-1}, b_\ell, \lambda)\, d\lambda \ , \\ &\approx \sum_{j=-N_Q}^{N_Q} \omega_j^{(\ell)} R(\Delta t\lambda_j^{(\ell)})^{n-b_{\ell-1}} (\mathbf{I}_m - \Delta t\lambda_j^{(\ell)}\mathcal{O})^{-1} \mathbb{1} y(b_{\ell-1}, b_\ell, \lambda_j^{(\ell)}) \ . \end{aligned} \tag{5.5}$$

The last step stands for a trapezoidal rule approximation of the contour representation of $\mathbf{U}_n^{(\ell)}$ with $2N_Q - 1$ quadrature nodes. The trapezoidal rule weights $\omega_j^{(\ell)}$ and the trapezoidal rule nodes $\lambda_j^{(\ell)}$ are given by

$$\omega_j^{(\ell)} = \frac{i\tau}{2\pi}\gamma_\ell'(\tau j), \quad \lambda_j^{(\ell)} = \gamma_\ell(\tau j),$$

where the parameter τ is adapted to μ and α in (López-Fernández et al. 2006, Section 4).

Note that (5.4) can be computed in parallel for the different contours with the recursive formula

$$\begin{cases} y(b,b,\lambda) = 0 & \text{for } b \in \mathbb{N}, \\ y(k+1,b,\lambda) = R(\Delta t\lambda) y(k,b,\lambda) + \Delta t \mathbf{b}^T (\mathbf{I}_m - \Delta t\lambda\mathcal{O})^{-1}\mathbf{g}_k & \text{for } k \geq b. \end{cases} \tag{5.6}$$

Thus, by exploiting the good approximation of the convolution weights along hyperbolae, each $\mathbf{U}_n^{(\ell)}$ can be approximated as in (5.5) in $\mathcal{O}(n \log(1/\varepsilon))$ effort, where ε is the target accuracy and the complexity n is only due to the computation of (5.6).

The FOCQ is very effective in the context of Volterra equations and its application to (4.8) is the central topic of the next section. Basically, rearranging the sum in (4.8) as in (5.3) and updating the values of (5.4) with (5.6) as the iteration index of (4.8) increases from 0 to N reduce the computational effort and the memory requirements to $\mathcal{O}(N \log N \log(\frac{1}{\varepsilon}))$ and to $\mathcal{O}(\log N \log(\frac{1}{\varepsilon}))$ respectively (Schädle et al. 2006). The additional perturbation error due to the convolution weight approximation along hyperbolae can be controlled by ε (Hiptmair et al. 2013, Section 5.2) and in practice the same convergence of the CQ is observed.

6. Algorithm. From (4.8) it is clear that the algorithm can be implemented in a direct way. The computational domain D is equipped with a (tetrahedral and hexahedral) finite element mesh. Precomputing the matrices $\mathbf{A}$ and $\mathbf{B}$ from (3.3) and $\mathbf{W}_0$ as in (4.6), we can define and store the sparse matrix

$$\boldsymbol{\mathcal{M}} := \mathbf{I}_m \otimes \mathbf{A} + \mathbf{W}_0 \otimes \mathbf{B} \in \mathbb{R}^{mM,mM} \ , \tag{6.1}$$

where M is the size of the finite element space and m is the number of stages of the underlying Runge–Kutta method. We can then rewrite (4.8) as the linear system

$$\boldsymbol{\mathcal{M}}\, \tilde{\boldsymbol{\mu}}_i = \tilde{\boldsymbol{\varphi}}_i - \sum_{j=0}^{i-1} (\mathbf{W}_{i-j} \otimes \mathbf{B})\, \tilde{\boldsymbol{\mu}}_j \quad \text{for } i = 0, .., N \ . \tag{6.2}$$

Algorithm 1 naive implementation

```
1: create a spatial mesh and assemble $\boldsymbol{A}$, $\boldsymbol{B}$ from (3.3)
2: set $\Delta t := T/(N+1)$
3: compute $\boldsymbol{\mathcal{M}}$ from (6.1)
4: for $i = 0 : N$ do
5:     compute $\mathbf{W}_i$ with (4.6)
6:     compute $\mathbf{oldconv} := \sum_{j=0}^{i-1} (\mathbf{W}_{i-j} \otimes \mathbf{B})\, \tilde{\boldsymbol{\mu}}_j$
7:     solve the linear system $\boldsymbol{\mathcal{M}}\, \tilde{\boldsymbol{\mu}}_i = \tilde{\boldsymbol{\varphi}}_i - \mathbf{oldconv}$
8: end for
```

Algorithm 1 shows how (6.2) can be solved with a naive implementation. At each iteration we have to compute the partial convolution in step 6. Since the matrix $\mathbf{B}$ represents an integration on the conductor boundary Γ, this convolution can be restricted to the entries of $\tilde{\boldsymbol{\mu}}_i$ related to the conductor boundary nodes Γ_h. By assuming that the time necessary for solving the $mM \times mM$ sparse linear system is $C_1(mM)$ we conclude that the computational time of Algorithm 1 is asymptotically

$$C_1(mM) \cdot N + C_2 \cdot \#\Gamma_h \cdot N^2 \ , \tag{6.3}$$

where $\#\Gamma_h$ denotes the number of nodes on the conductor boundary.

The naive implementation can be accelerated by exploiting FOCQ when the iteration time $i > n_0$. As already anticipated, the idea is to split the computation of $\mathbf{oldconv}$ in step 6 of Algorithm 1 into logarithmically few terms $\mathbf{O}_n^{(\ell)}$ as in (5.3). Each of these requires only the values of (5.4), which are computed through (5.6) and whose inputs are given by the sequence $(b_\ell^i)_{\ell=0}^{\lceil \log_B i \rceil}$. Since the values of the sequence change in a non-linear way as the iteration time i increases, it is more convenient to pre-compute all the different sequences at the beginning.

Algorithm 2 fast implementation

```
1: create a spatial mesh and assemble $\boldsymbol{A}$, $\boldsymbol{B}$ from (3.3)
2: set $\Delta t := T/(N+1)$
3: compute $\boldsymbol{\mathcal{M}}$ from (6.1)
4: for $i = 0 : N$ do
5:     compute $(b_\ell^i)_{\ell=0}^{\lceil \log_B i \rceil}$
6: end for
7: for $i = 0 : n_0$ do
8:     compute $\mathbf{W}_i$ with (4.6)
9:     compute $\mathbf{oldconv} := \sum_{j=0}^{i-1} (\mathbf{W}_{i-j} \otimes \mathbf{B})\, \tilde{\boldsymbol{\mu}}_j$
10:    solve the linear system $\boldsymbol{\mathcal{M}}\, \tilde{\boldsymbol{\mu}}_i = \tilde{\boldsymbol{\varphi}}_i - \mathbf{oldconv}$
11:    update $\mathbf{y}\{i\}$ with (5.6)
12: end for
13: for $i = n_0 + 1 : N$ do
14:     for $\ell = 1 : \lceil \log_B i \rceil$ do
15:         compute $\mathbf{O}_i^{(\ell)} := \sum_{j=b_\ell}^{b_{\ell-1}-1} (\mathbf{W}_{i-j} \otimes \mathbf{I}_M)\, \tilde{\boldsymbol{\mu}}_j$ with (5.5)
16:     end for
17:     compute $\mathbf{oldconv} := (\mathbf{I}_m \otimes \mathbf{B}) \sum_{\ell=1}^{\lceil \log_B i \rceil} \mathbf{O}_i^{(\ell)}$
18:     solve the linear system $\boldsymbol{\mathcal{M}}\, \tilde{\boldsymbol{\mu}}_i = \tilde{\boldsymbol{\varphi}}_i - \mathbf{oldconv}$
19:     update $\mathbf{y}\{i\}$ with (5.6)
20: end for
```

Combining this knowledge with (5.6), we can start computing the values of $y(b_{\ell-1}, b_\ell, \lambda)$ from the beginning for all contours ℓ and store them in a struct $\mathbf{y}\{i\}$. From (5.6) is clear that updating the values of $\mathbf{y}\{i\}$ as the algorithm runs requires only the knowledge of the values stored in $\mathbf{y}\{i-1\}$. Since only logarithmic few contours come into play, this strategy reduces the active memory requirements to $\mathcal{O}(\log_B N)$.

The fast implementation is summarized in Algorithm 2[3]. Since the approximation (5.5) is poor for the first few weights, for the first n_0 iterations we rely on a naive implementation. Note that in order to reduce the memory requirements we already start updating $\mathbf{y}\{i\}$. The computational complexity of these first steps is negligible because in practice $n_0 \ll N$. From lines 13-20 it is then clear that the computational complexity of Algorithm 2 is proportional to

$$C_1(mM) \cdot N + C_2 \cdot \#\Gamma_h \cdot N \log_B(N) \, . \tag{6.4}$$

7. Numerical Experiments. In our numerical tests we consider (2.5). We choose $\widetilde{D}$ to be an annulus around the origin with radii 0.5 and 2 and we fix $T = 4$. The source function is included by imposing the Dirichlet boundary condition $g(x, y, t) := \frac{32}{105\sqrt{\pi}} t^{7/2} + \frac{t^3}{6} \log(4)$ on $\partial\widetilde{\Omega}$. The analytical solution is then

$$u(x, y, t) := \frac{32}{105\sqrt{\pi}} t^{7/2} + \frac{t^3}{6} \left(\frac{1}{2} \log(x^2 + y^2) + \log(2) \right) . \tag{7.1}$$

In the implementation we opt for linear Lagrangian finite elements on triangular meshes with nodal basis functions[4]. For the FOCQ range parameter we choose $B = 10$ while the hyperbola parameters are chosen accordingly to (López-Fernández et al. 2006, Section 4).

A first numerical test is performed by choosing the FOCQ based on the implicit Euler method, which is the 1-step RadauIIA method (FOCQ of order 1). For 6 different spatial grids and 12 different time steps we measured the error in the time-discrete norm

$$\|u(t, \mathbf{x})\|^2_{\ell^2_{\Delta t}([0,4], H^1(\widetilde{D}))} := \Delta t \sum_{n=1}^{N} \|u(t_n, \mathbf{x})\|^2_{H^1(\widetilde{D})}$$

as well as the $L^2(\widetilde{D})$-error in space[5] at a fixed time $t = 4$. The spatial triangular meshes have been created through uniform refinement while the timesteps by repetitively halving an initial timestep.

The expected linear algebraic convergence both in time and space in the $\ell^2_{\Delta t}([0, 4], H^1(\widetilde{D}))$-norm is observed in Figure 7.1 (left). The rates of algebraic convergence become more conspicuous when we examine the $L^2(\widetilde{D})$-norm in space at a fixed time, where we have quadratic convergence in space; see figure 7.1 (right).

The impact of the FOCQ on the algorithm is investigated in Figure 7.2 (left). We consider the fourth finest spatial grid and the timestep $\Delta t = 2^{-8}$ (see the dots in Figure 7.1) and we compute the error in the $L^2(\widetilde{D})$-norm in space at a fixed time for different numbers of quadrature nodes N_Q on the contours. We see that few quadrature nodes on the contours are enough to make negligible the perturbation error due to the FOCQ approximation of the convolution weights $\mathbf{W}_i$.

[3]The pre-computation of the sequences in steps 4-6 is not strictly necessary. In (Schädle et al. 2006, Section 4.1) is given a pseudo–code for updating $\mathbf{y}\{i\}$ that does not require a priori knowledge of $(b_\ell)_{\ell=0}^{\lceil \log_B i \rceil}$.

[4]The experiments are perfomed in MATLAB and are based on the library LehrFEM developed at the ETHZ.

[5]Both the $H^1(\widetilde{D})$- and the $L^2(\widetilde{D})$-norm are computed approximately with 7 point quadrature rules on triangles.

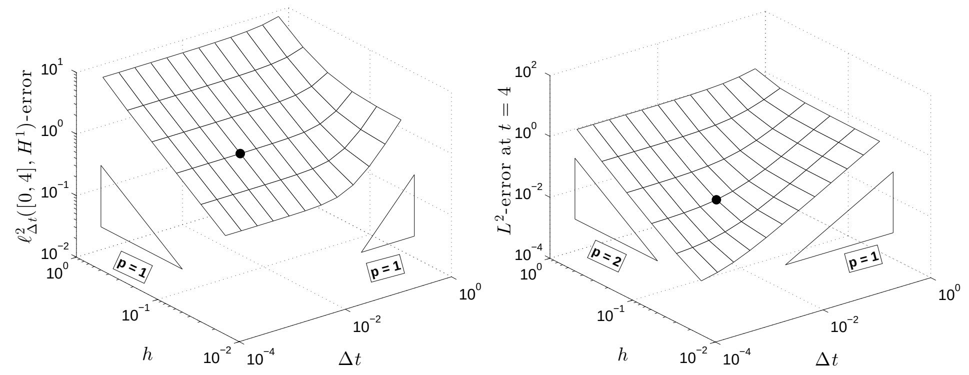

FIG. 7.1. *Error in the* $\ell^2_{\Delta t}([0,4], H^1(\widetilde{D}))$*-norm (*left*) and in the* $L^2(\widetilde{D})$ *at a fixed time* $t = 4$ *(*right*) for the coupling of FEM and FOCQ base on the implicit Euler method. The two dots denote the spatial mesh and timestep used in Figure 7.2 (left).*

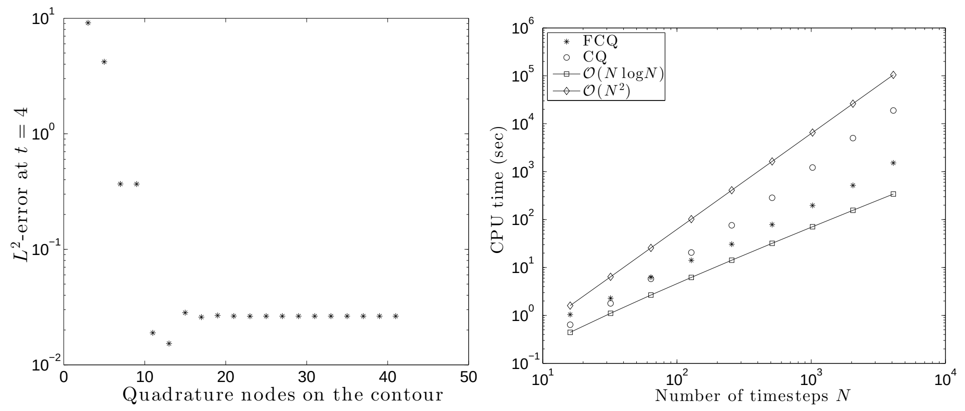

FIG. 7.2. Left: *Impact of FOCQ on the total error in the* $L^2(\widetilde{D})$ *at a fixed time* $t = 4$ *(*right*) for the coupling of FEM and FCQ base on the implicit Euler method.* Right: *Cpu time in seconds versus the number of time steps for Algorithms 1 and 2.*

In Figure 7.2 (right) we compare the time required for solving (2.5) with the Algorithms 1 and 2. We see that the growth of the computation time coincides asymptotically with the theoretical growths (6.3) and (6.4).

We perform a second numerical test and this time the convolution is approximated by using the FOCQ based on the 2-stage RadauIIA method (FOCQ of order 3). Again we measure both the $\ell^2_{\Delta t}([0,4], H^1(\widetilde{D}))$-error and the $L^2(\widetilde{D})$-error in space at a fixed time $t = 4$ for several meshes and timesteps. We expect that the convolution quadrature error contributes to the total error with a cubic algebraic rate in Δt. This is only partially confirmed by the experiment because the total error is almost always dominated by the discretization error in space, as we can see in Figure 7.3.

8. Conclusion.

We have investigated a numerical scheme for solving the eddy current problem (2.4). The scheme has computational complexity $\mathcal{O}(N \log N)$, where N denotes the number of timesteps, and inherits the stability properties of the convolution quadrature. For spatial and temporal discretization error we have observed algebraic decay in terms of mesh width and timestep size, respectively. The error due to the fast and oblivious approximation

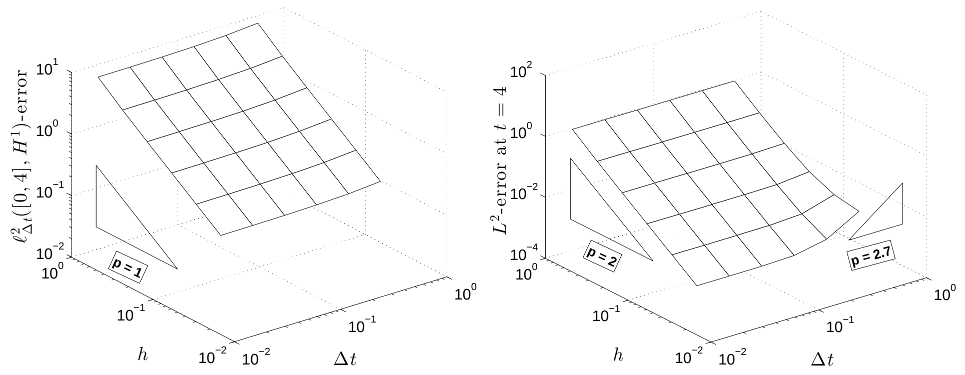

FIG. 7.3. *Error in the* $\ell^2_{\Delta t}([0,4], H^1(\widetilde{D}))$*-norm (*left*) and in the* $L^2(\widetilde{D})$ *at a fixed time* $t = 4$ *(*right*) for the coupling of FEM and FOCQ based on the 2-stage RadauIIA method.*

decays exponentially in a discretization parameter and is negligible compared to the other error contributions. For an a priori convergence analysis of a fully discrete oblivious finite element CQ for transient eddy current problems we refer the reader to (Hiptmair et al. 2013).

REFERENCES

A. Alonso-Rodriguez and A. Valli (2010), *Eddy Current Approximation of Maxwell Equations*, Vol. 4 of *Modelling, Simulation & Applications*, Springer, Milan.

L. Banjai, C. Lubich and J. M. Melenk (2011), 'Runge-Kutta convolution quadrature for operators arising in wave propagation', *Numer. Math.* **119**(1), 1–20.

V. De Santis, S. Cruciani, M. Feliziani and M. Okoniewski (2012), 'Efficient low order approximation for surface impedance boundary conditions in finite-difference time-domain method', *Magnetics, IEEE Transactions on* **48**(2), 271 –274.

E. Hairer and G. Wanner (2010), *Solving ordinary differential equations. II*, Vol. 14 of *Springer Series in Computational Mathematics*, Springer-Verlag, Berlin. Stiff and differential-algebraic problems, Second revised edition, paperback.

R. Hiptmair (2002), 'Finite elements in computational electromagnetism', *Acta Numerica* **11**, 237–339.

R. Hiptmair, A. Paganini and M. López-Férnandez (2013), Fast convolution quadrature based impedance boundary conditions, Technical Report 2013-02, Seminar for Applied Mathematics, ETH Zürich, Switzerland.

M. López-Fernández, C. Lubich, C. Palencia and A. Schädle (2005), 'Fast Runge-Kutta approximation of inhomogeneous parabolic equations', *Numer. Math.* **102**(2), 277–291.

M. López-Fernández, C. Palencia and A. Schädle (2006), 'A spectral order method for inverting sectorial Laplace transforms', *SIAM J. Numer. Anal.* **44**(3), 1332–1350 (electronic).

C. Lubich (1988*a*), 'Convolution quadrature and discretized operational calculus. I', *Numer. Math.* **52**(2), 129–145.

C. Lubich (1988*b*), 'Convolution quadrature and discretized operational calculus. II', *Numer. Math.* **52**(4), 413–425.

C. Lubich and A. Ostermann (1993), 'Runge-Kutta methods for parabolic equations and convolution quadrature', *Math. Comp.* **60**(201), 105–131.

K. S. Oh and J. Schutt-Aine (1995), 'An efficient implementation of surface impedance boundary conditions for the finite-difference time-domain method', *Antennas and Propagation, IEEE Transactions on* **43**(7), 660 –666.

A. Schädle, M. López-Fernández and C. Lubich (2006), 'Fast and oblivious convolution quadrature', *SIAM J. Sci. Comput.* **28**(2), 421–438 (electronic).

T. Senior (1960), 'Impedance boundary conditions for imperfectly conducting surfaces', *Applied Scientific Research, Section B* **8**, 418–436. 10.1007/BF02920074.