

IMPEDANCE BOUNDARY CONDITIONS IN
TIME DOMAIN
SEMESTER THESIS

Author: Alberto David Maria Paganini
Supervisor: Prof. Dr. Ralf Hiptmair
ETHZ

March 12, 2011

Contents

1	Introduction	3
2	Simple model problem	4
3	Convolution Quadrature	6
4	Test of convolution quadrature	9
5	Discretization of exterior BVP	17
6	Test for exterior BVP	21
6.1	Analytic solution	21
6.2	Mesh Generation	22
6.3	Computation of mass matrix	23
6.4	Computation of the numerical solution	24
6.5	Collection of the errors	26
6.6	Interpretation of the results	27
7	References	28

1 Introduction

Alternating electromagnetic fields decay exponentially when penetrating a good conductor (skin effect). Therefore, a reasonable approximation of the electromagnetic Dirichlet-to-Neumann map of the interior of a good conductor is provided by the impedance boundary conditions

$$\mathbf{E}_t = \frac{1}{2}\sqrt{2}(1-i)\sqrt{\frac{\mu}{\sigma\omega}}(\mathbf{H} \times \mathbf{n}), \quad (1)$$

where $\omega > 0$ is a fixed angular frequency, characterizing the temporal variation of all electromagnetic fields. The conductivity σ and permeability μ are known material parameters.

The relationship (1) is valid in the frequency domain only. However, often sinusoidal temporal variation of the fields cannot be assumed, which forces us to resort to time domain methods. However, when formulating impedance boundary conditions in the time domain, we encounter temporal convolutions of the form

$$\mathbf{E}_t(\mathbf{x}, t) = \int_{t_0}^t k(\mu, \sigma, \tau - t)(\mathbf{H} \times \mathbf{n})(\mathbf{x}, \tau) d\tau. \quad (2)$$

In word, the boundary conditions become non-local in time. This renders a straightforward discretization of (2) prohibitively expensive, if many timesteps are to be carried out.

Instead of the eddy current problem we will consider a second-order parabolic problem with a boundary condition involving convolution in time, in which just the Laplace transform of the kernel is known. To approximate this kind of convolution we will discuss the algorithm developed by Lubich and see how it applies. We will conclude with a concrete numerical experiment based on the library LehrFEM.

2 Simple model problem

We consider the following parabolic PDE

$$\begin{cases} \frac{\partial}{\partial t}(\sigma u) - \Delta u = f & \text{in } \Omega \times]0; T], \\ u = 0 & \text{in } \Omega \times \{t = 0\}, \\ u = 0 & \text{on } \partial\Omega \times]0; T], \end{cases}$$

where

$$\sigma(x) = \begin{cases} c_0 \gg 1 & \text{in } \Omega_0 \subset\subset \Omega, \\ 0 & \text{in } \Omega \setminus \Omega_0 \end{cases}$$

and with $\text{supp}(f) \subset \Omega \setminus \Omega_0$, where $\Omega \setminus \Omega_0$ is an annulus around zero.

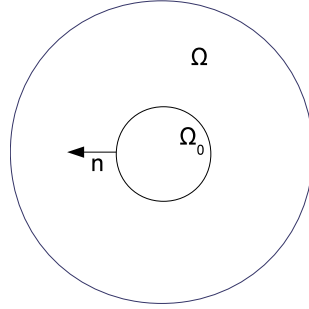


Figure 1: Domain of the PDE.

We want to compute the solution u in $\Omega \setminus \Omega_0$. For this we need an additional boundary condition on $\partial\Omega_0$. Since c_0 is very large, an impedance boundary condition provides a good approximation. To derive it we consider the 1D-problem (we assume that $\text{Re}(s) > 0$; $\mathcal{L}(u)$ denotes the Laplace transform of u)

$$\begin{cases} sc_0 \mathcal{L}(u) - \partial_\xi^2 \mathcal{L}(u) = 0 & \text{for } \xi > 0, \\ u(\xi) \rightarrow 0 & \text{for } \xi \rightarrow \infty. \end{cases}$$

A solution can be found with the Ansatz $\mathcal{L}(u)(\xi) = e^{\lambda\xi}$. With the decay condition at infinity we obtain $\mathcal{L}(u)(\xi) = e^{-\sqrt{sc_0}\xi}$. If we differentiate $\mathcal{L}(u)$ one time we have $\partial_\xi \mathcal{L}(u)(\xi) = -\sqrt{sc_0} \mathcal{L}(u)(\xi)$ and thus, by taking the limit $\xi \rightarrow 0$, we obtain the relation

$$\partial_\xi \mathcal{L}(u)(0) = -\sqrt{sc_0} \mathcal{L}(u)(0).$$

This motivates the following impedance boundary condition

$$\nabla \mathcal{L}(u) \cdot -\mathbf{n} = -\sqrt{sc_0} \mathcal{L}(u) \quad \text{on } \partial\Omega_0.$$

The previous PDE becomes

$$\begin{cases} -\Delta u = f & \text{in } \Omega \setminus \Omega_0 \times]0; T], \\ u = 0 & \text{in } \Omega \setminus \Omega_0 \times \{t = 0\}, \\ u = 0 & \text{on } \partial\Omega \times]0; T], \\ \nabla u \cdot -\mathbf{n} = \int_0^t k(t - \tau) \cdot u(\tau) d\tau & \text{on } \partial\Omega_0 \times]0; T], \end{cases}$$

where $\mathcal{L}(k) = -\sqrt{sc_0}$.

The corresponding variational problem is

$$\int_{\Omega \setminus \Omega_0} \nabla u \cdot \nabla v \, dx - \int_{\partial\Omega_0} \nabla u \cdot -\mathbf{n} \cdot v \, dS = \int_{\Omega \setminus \Omega_0} f \cdot v \, dx$$

for every $v \in H^1(\Omega \setminus \Omega_0)$ with $v = 0$ on $\partial\Omega$ and for every $t \in]0; T]$.

We substitute $\nabla u \cdot -\mathbf{n}$, then the variational form becomes

$$\int_{\Omega \setminus \Omega_0} \nabla u \cdot \nabla v \, dx - \int_{\partial\Omega_0} \left(\int_0^t k(t - \tau) \cdot u(\tau) d\tau \right) \cdot v \, dS = \int_{\Omega \setminus \Omega_0} f \cdot v \, dx \quad (3)$$

with $v \in H^1(\Omega \setminus \Omega_0)$, $v = 0$ on $\partial\Omega$ and for every $t \in]0, T]$.

3 Convolution Quadrature

In the variational formulation we have to compute a convolution in which only the Laplace transform of the kernel is known. C. Lubich has developed an efficient algorithm for the approximation of this kind of convolution. We now recall the general statement.

We want to approximate a continuous convolution

$$\int_0^t f(\tau)g(t-\tau)d\tau \quad \text{for } t > 0 \quad (4)$$

where the Laplace transform $F(s) := \mathcal{L}(f)$ and $g(t)$ are known functions.

Notation: it is useful to introduce the operational calculus notation, which sets

$$F(\partial)g(t) := \int_0^t f(\tau)g(t-\tau)d\tau.$$

First of all, we assume that we are handling with a sectorial Laplace transform $F(s)$.

Assumption 1.

- $F(s)$ is analytic in a sector $|\arg(s-c)| < \pi - \varphi$ with $\varphi < \frac{1}{2}\pi$ and real c (see Figure 2),
- $|F(s)| \leq M|s|^{-\mu}$ for some real positive μ and M .

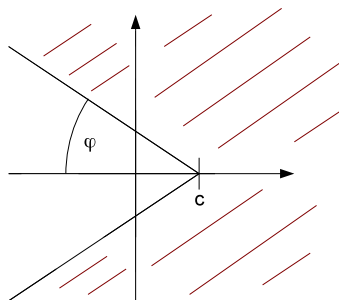


Figure 2: Domain of analyticity of $F(s)$.

The convolution quadrature is based on a linear multistep method for which we assume strong $A(\alpha)$ -stability and order p . Linear multistep methods are of the form

$$\sum_{j=0}^k \alpha_j \cdot y_{n+j-k} = h \cdot \sum_{j=0}^k \beta_j (\lambda \cdot y_{n+j-k} + g((n+j-k) \cdot h)) \quad n \geq 0$$

where the starting values $y_{-k}, y_{-k+1}, \dots, y_{-1}$ are equal zero while the function g is extended by zero to the negative real axis (k is the number of steps of the method). These methods are represented by two characteristic polynomials

$$\rho(\zeta) := \sum_{j=0}^k \alpha_j \cdot \zeta^j$$

and

$$\sigma(\zeta) := \sum_{j=0}^k \beta_j \cdot \zeta^j.$$

We define

$$\delta(\zeta) := \frac{\rho(\zeta)}{\sigma(\zeta)},$$

then the following assumption on $\delta(\zeta)$ guarantees the strong $A(\alpha)$ -stability and the order of convergence p of the linear multistep method.

Assumption 2.

- $\delta(\zeta)$ is analytic and without zeros in a neighbourhood of the closed unit disk $|\zeta| \leq 1$, except for a zero at $\zeta = 1$,
- $|\delta(\zeta)| \leq \pi - \alpha$ for $|\zeta| < 1$ and $\alpha > \varphi$,
- $\frac{1}{h}\delta(e^{-h}) = 1 + \mathcal{O}(h^p)$ with the order $p \geq 1$.

Now we can define the convolution quadrature with a step size $h > 0$, $h|t$, as

$$F(\partial_h)g(t) := \sum_{n=0}^M w_n \cdot g(t - nh)$$

where $M := \frac{t}{h}$ and $g(t)$ is extended by zero on the negative real axis. The weights w_n can be derived from the Taylor series

$$F\left(\frac{\delta(\zeta)}{h}\right) = \sum_{n=0}^{\infty} w_n \zeta^n.$$

The above defined convolution quadrature satisfies the following error bound.

Theorem 1. *Under the two assumptions and for real $\beta > 0$ we have*

$$|F(\partial)t^{\beta-1} - F(\partial_h)t^{\beta-1}| \leq \begin{cases} Ct^{\mu-1}h^\beta & \text{for } 0 < \beta \leq p, \\ Ct^{\mu-1+\beta-p}h^p & \text{for } \beta \geq p, \end{cases}$$

where the constant C is independent of h and t .

Remark: the theorem is still valid for $\mu \leq 0$ (in Assumption 1). In this case the continuous convolution (4) is considered as

$$\partial^k \tilde{F}(\partial)g(t) := \left(\frac{d}{dt}\right)^k \int_0^t \tilde{f}(\tau)g(t-\tau)d\tau$$

where k is the smallest integer such that $\mu + k > 0$ and

$$\tilde{f}(t) := \mathcal{L}^{-1}(F(s)s^{-k}).$$

The convolution quadrature would then be

$$\partial_h^k \tilde{F}(\partial_h)g(t) := \sum_{n=0}^M \tilde{w}_n \cdot \partial_h^k g(t-nh).$$

4 Test of convolution quadrature

We first compute analytically the convolution involving the identity function

$$g(t) = \sqrt{t}$$

and a kernel which the Laplace transform is

$$F(s) = -\sqrt{sc_0}.$$

Since

$$|F(s)| = |-\sqrt{sc_0}| \leq \sqrt{c_0}|s|^{-\left(-\frac{1}{2}\right)},$$

to compute the analytic convolution we have to take the Hadamard finite part of it with $k = 1$, id est

$$\begin{aligned} F(\partial)g(t) &= \left(\frac{d}{dt}\right) \int_0^t \tilde{f}(\tau)g(t-\tau)d\tau \\ &= \left(\frac{d}{dt}\right) \int_0^t \mathcal{L}^{-1}(F(s)s^{-1})g(t-\tau)d\tau \\ &= \left(\frac{d}{dt}\right) \int_0^t \mathcal{L}^{-1}\left(-\sqrt{\frac{c_0}{s}}\right)g(t-\tau)d\tau \\ &= \left(\frac{d}{dt}\right) \int_0^t -\sqrt{\frac{c_0}{\tau\pi}} \cdot \sqrt{t-\tau}d\tau \\ &= -\sqrt{\frac{c_0}{\pi}} \left(\frac{d}{dt}\right) \left[\tau\sqrt{\frac{t-\tau}{\tau}} - \frac{1}{2}t \arctan\left(\frac{\sqrt{\frac{t}{\tau}-1}(t-2\tau)}{2(t-\tau)}\right) \right]_0^t \\ &= -\sqrt{\frac{c_0}{\pi}} \left(\frac{d}{dt}\right) \left[\sqrt{\tau(t-\tau)} - \frac{1}{2}t \arctan\left(\frac{t-2\tau}{2\sqrt{\tau(t-\tau)}}\right) \right]_0^t \\ &= -\sqrt{\frac{c_0}{\pi}} \left(\frac{d}{dt}\right) \left(t\frac{\pi}{2}\right) \\ &= -\frac{\sqrt{c_0\pi}}{2}. \end{aligned}$$

For the convolution quadrature we compute the weights for the backward Euler method (which is a strong $A(\alpha)$ -stable linear multistep method of order

1).

$$\begin{aligned}
\tilde{F}\left(\frac{\delta(\zeta)}{h}\right) &= -\sqrt{\frac{c_0 h}{\delta(\zeta)}} \\
&= -\sqrt{\frac{c_0 h}{1-\zeta}} \\
&= -\sqrt{c_0 h} \sum_{n=0}^{\infty} (-1)^n \binom{-\frac{1}{2}}{n} \zeta^n,
\end{aligned}$$

where

$$\begin{aligned}
\binom{-\frac{1}{2}}{n} &= \frac{\Gamma(-\frac{1}{2} + 1)}{\Gamma(n + 1)\Gamma(-\frac{1}{2} - n + 1)}, \\
&= \frac{\Gamma(\frac{1}{2})}{\Gamma(n + 1)\Gamma(\frac{1}{2} - n)}.
\end{aligned}$$

Thus

$$\tilde{w}_n = -\frac{\sqrt{c_0 h}(-1)^n \Gamma(\frac{1}{2})}{\Gamma(n + 1)\Gamma(\frac{1}{2} - n)}.$$

We compute the convolution quadrature and we investigate the error

$$|F(\partial)y(t) - F(\partial_h)y(t)|$$

for $t = 2$ performing seven step refinement. If we consider Theorem 1, we have $\beta = 2$, $p = 1$ and $\mu = -\frac{1}{2}$; thus we expect an algebraic convergence of order 1, which is confirmed by the test (see Figure 3). The error after the seventh refinement increases because, from w_{171} on, MATLAB sets the weights equal to infinity (to compute the weights we used the function *gamma*).

```

% test the convolution quadrature for backward Euler method using
% g(t)=sqrt(t) and F(s)=-sqrt(c_0 s)

function convtest2
5
% constant
c = 1;

% analytic convolution
10 f=@(t) -sqrt(c*pi)/2;

% function g
g=@(x) gi(x);

15 % number of refinement
p=7;

% final time
T=2;
20
% step size's vector
h=zeros(p,1);

% iteration's vector
25 k=zeros(p,1);

for i=1:p
    h(i,1)=0.5^i;
30    k(i,1)=T/h(i,1);
end

% convolution quadrature
y=zeros(p,1);
35
for j =1:p
    % weights
    w=zeros(k(j,1)+1,1);
40    for i=1:k(j,1)+1
        w(i,1)=-sqrt(c*h(j))*((-1)^(i-1))*gamma(1/2)/gamma(i-1+1)/gamma
            (1/2-(i-1));
    end
45    end

    % summing up
    for l=1:k(j,1)+1
50        y(j,1)=y(j,1)+w(l,1)*(g(T-(l-1)*h(j))-g(T-l*h(j)))/h(j);
    end
end

% error vector
55 err=abs(ones(p,1)*f(2)-y);
loglog(h,err,'*-');
title('Error of backward Euler');
xlabel('stepsize h');
ylabel('error');
60 grid on;
end

```

```

function y=gi(x)
65 %g is extended by zero on the negative real axis
   if x<0
       y=0;
   else
       y=sqrt(x);
70 end
end

```

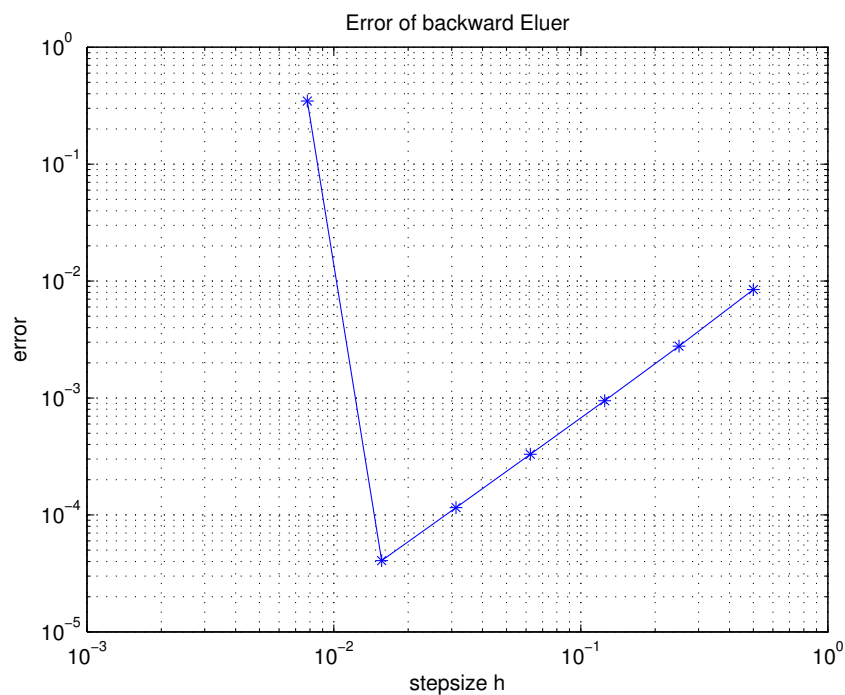


Figure 3: Error of convolution quadrature.

Now we try to solve the following integral equation

$$y(t) = \sqrt{t} + \frac{\sqrt{c_0\pi}}{2} + \int_0^t F(t-\tau)y(\tau)d\tau$$

The analytic solution of this problem is

$$y(t) = \sqrt{t}.$$

We apply the convolution quadrature to derive an algorithm which solves this problem. Let $y_k \doteq y(k \cdot h)$ and $f(t) := \sqrt{t} + \frac{\sqrt{c_0 \pi}}{2}$. Then

$$\begin{aligned}
y_k &= f(kh) + \sum_{i=0}^k \tilde{w}_i \delta_h y(kh - ih) \\
&= f(kh) + \sum_{i=0}^k \tilde{w}_i \frac{y(kh - ih) - y(kh - (i+1)h)}{h} \\
&= f(kh) + \tilde{w}_0 \frac{y(kh) - y(kh - h)}{h} + \sum_{i=1}^k \tilde{w}_i \frac{y(kh - ih) - y(kh - (i+1)h)}{h}.
\end{aligned}$$

For $k > 1$

$$\begin{aligned}
\left(1 - \frac{\tilde{w}_0}{h}\right)y_k &= f(kh) - \frac{\tilde{w}_0}{h}y_{k-1} + \sum_{i=1}^k \tilde{w}_i \frac{y_{k-i} - y_{k-(i+1)}}{h} \\
&= f(kh) - \frac{\tilde{w}_0}{h}y_{k-1} + \sum_{i=1}^{k-1} \tilde{w}_i \frac{y_{k-i} - y_{k-(i+1)}}{h}
\end{aligned}$$

because $y_0 = y_{-1} = 0$. Thus the discrete solution can be found with the following scheme:

$$\begin{aligned}
y_0 &= 0, \\
y_1 &= \frac{f(h)}{1 - \frac{\tilde{w}_0}{h}}, \\
y_k &= \frac{f(kh) - \frac{\tilde{w}_0}{h}y_{k-1} + \sum_{i=1}^{k-1} \tilde{w}_i \frac{y_{k-i} - y_{k-(i+1)}}{h}}{1 - \frac{\tilde{w}_0}{h}} \quad \text{for } k > 1.
\end{aligned}$$

```

% convolution integral equation:  $y(t) = \sqrt{t}$ 
% is the solution of  $y(t) = \sqrt{t} + \sqrt{\pi}/2 + \int_0^t k(t-x)y(x) dx$ 
% where  $L(k)(s) = -\sqrt{s}$ 

5 function [h,y]=convtest23(p)

% exact solution
f=@(t) sqrt(t);

10 % number of refinement
%p=3;

% step size's vector
h=0.25^p;
15 %final time
T=2;

% number of iterations
20 k=T/h;

%weights
w=zeros(1,k+1);

25 for j=1:k+1

    w(1,j)=-sqrt(h)*(-1)^(j-1)*gamma(0.5)/gamma(j-1+1)/gamma(1/2-(j-1));

end
30

% numerical solution
y=zeros(1,k+1);

35 y(1,2)=(sqrt(h) + sqrt(pi)/2)/(1-w(1)/h);

for i=2:k
    sum=0;
    for l=1:i-1
40        sum=sum+w(l+1)*(y(1,i-l+1)-y(i-1))/h;
    end

    y(1,i+1)=(sqrt(i*h) + sqrt(pi)/2-w(1)/h*y(1,i)+sum)/(1-w(1)/h);
end
45

%figure;
%plot(0:h:T,y(1,1:k+1),0:h:T,f(0:h:T));
%legend('ref 1','exact');
50 %xlabel('time');
%ylabel('obtained solution');
%grid on;
end

function convtest24

% exact solution
f=@(t) sqrt(t);
5 %final time
T=2;

```

```

%compute solutions
10 [h(1),y1]=convtest23(1);
    [h(2),y2]=convtest23(2);
    [h(3),y3]=convtest23(3);

15
%plot
figure;
plot(0:h(1):T,y1,0:h(2):T,y2,0:h(3):T,y3,0:h(2):T,f(0:h(2):T));
legend('ref 1','ref 2','ref 3','exact');
20 xlabel('time');
    ylabel('obtained solution');
    grid on;

%error vector
25 err=zeros(3,T/h(1));
    for i=1:T/h(1)
        err(1,i)=abs(f((i)*h(1))-y1(1+i));
        err(2,i)=abs(f((i)*h(1))-y2(1+i*4));
        err(3,i)=abs(f((i)*h(1))-y3(1+i*16));
30 end
    figure;
    plot(h(1):h(1):2,err);
    legend('ref 1','ref 2','ref 3');
    ylabel('error');
35 xlabel('time');
    grid on;
end

```

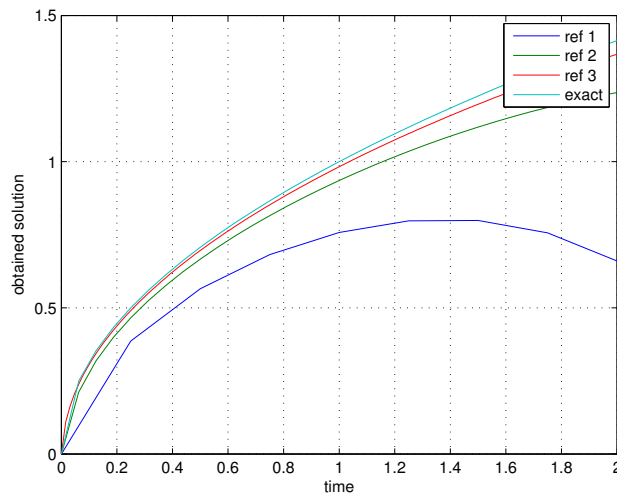


Figure 4: Evolution of the discrete solutions.

Figure 6 shows the first order algebraic convergence of the error for a fixed time. Figure 6 shows the first order algebraic convergence of the error for a fixed time.

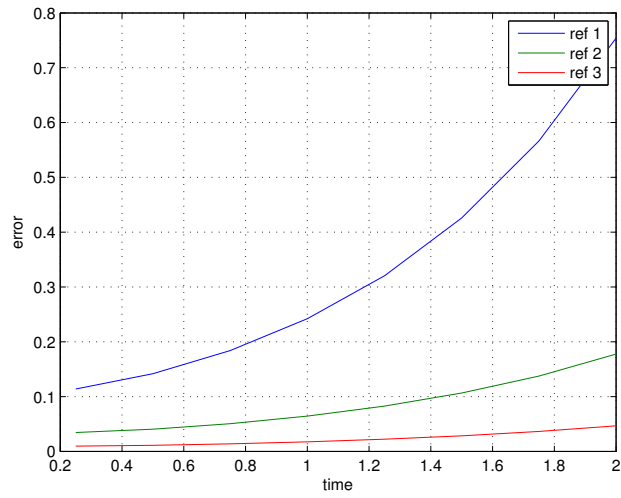


Figure 5: Evolution of the error.

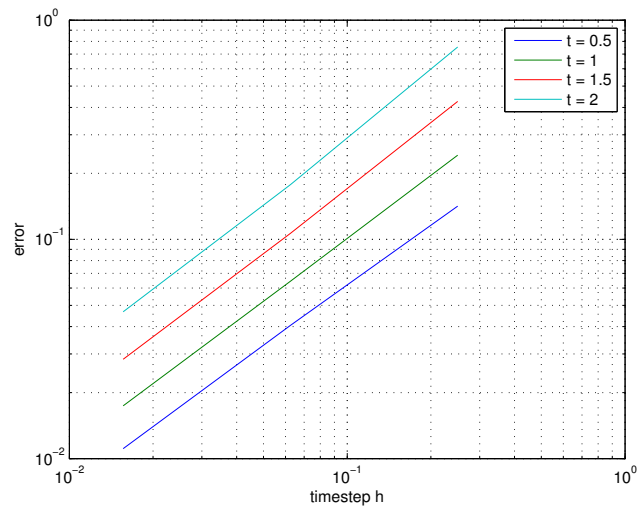


Figure 6: Error at fixed time.

5 Discretization of exterior BVP

We want to discretize the variational form (3). The first step is to approximate the convolution $\int_0^t k(t-\tau) \cdot u(\tau) d\tau$ using the convolution quadrature. In this case $F(s) = -\sqrt{sc_0}$, which is analytic everywhere except for the negative real line and thus satisfies Assumption 1 for any real $c > 0$, $0 < \varphi < \frac{1}{2}\pi$ and $\mu = -\frac{1}{2}$. By taking backward Euler as linear multistep method, we have

$$\int_0^t k(t-\tau) \cdot u(\tau) d\tau \approx \sum_{n=0}^M w_n(h_t) \cdot \partial_h u(x, t - nh_t),$$

where

$$\partial_h u(x, t - nh_t) := \frac{u(x, t - nh_t) - u(x, t - (n+1)h_t)}{h_t}.$$

Thus the variational form (3) becomes

$$\int_{\Omega \setminus \Omega_0} \nabla u \cdot \nabla v \, dx - \int_{\partial\Omega_0} \sum_{n=0}^M w_n(h_t) \cdot \partial_h u(x, t - nh_t) \cdot v \, dS = \int_{\Omega \setminus \Omega_0} f \cdot v \, dx$$

with $v \in H^1(\Omega \setminus \Omega_0)$, $v = 0$ on $\partial\Omega$ and for every $t \in]0, T]$.

Now we apply the finite linear element method over a simplicial mesh. Let N be the number of nodes of the mesh and $\{b_N^j | j = 1, \dots, N\}$ be the set of hat functions. The discretized variational problem reads to find u_N in $\text{span}\{b_N^j | j = 1, \dots, N\}$ such that

$$\int_{\Omega \setminus \Omega_0} \nabla u_N \cdot \nabla b_N^j \, dx - \int_{\partial\Omega_0} \sum_{n=0}^M w_n(h_t) \cdot \partial_h u_N(x, t - nh_t) \cdot b_N^j \, dS = \int_{\Omega \setminus \Omega_0} f \cdot b_N^j \, dx$$

for $j = 1, \dots, N$ and for every $t \in]0; T]$.

The solution u_N can be written as a linear combination of the hat functions

$$u_N(x, t) = \sum_{i=1}^N \mu_i(t) b_N^i$$

where the coefficients $\mu_i(t)$ are time dependent. We will now consider the three terms of the discretized variational problem separately.

$$\begin{aligned}
\int_{\Omega \setminus \Omega_0} \nabla u_N \cdot \nabla b_N^j \, dx &= \int_{\Omega \setminus \Omega_0} \nabla \left(\sum_{i=1}^N \mu_i(t) b_N^i \right) \cdot \nabla b_N^j \, dx \\
&= \int_{\Omega \setminus \Omega_0} \sum_{i=1}^N \mu_i(t) \nabla b_N^i \cdot \nabla b_N^j \, dx \\
&= \sum_{i=1}^N \left(\int_{\Omega \setminus \Omega_0} \nabla b_N^i \cdot \nabla b_N^j \, dx \right) \mu_i(t)
\end{aligned}$$

By setting

$$\boldsymbol{\mu}(t) := (\mu_i(t))_{i=1}^N$$

and

$$\mathbf{A} := \left(\int_{\Omega \setminus \Omega_0} \nabla b_N^i \cdot \nabla b_N^j \, dx \right)_{i,j=1}^N$$

we can summarize the first part with

$$\mathbf{A} \cdot \boldsymbol{\mu}(t).$$

For the second term we have

$$\begin{aligned}
& \int_{\partial\Omega_0} \sum_{n=0}^M w_n(h_t) \cdot \partial_h u_N(x, t - nh_t) \cdot b_N^j \, dS \\
&= \int_{\partial\Omega_0} \sum_{n=0}^M w_n(h_t) \cdot \frac{u_N(x, t - nh_t) - u_N(x, t - (n+1)h_t)}{h_t} \cdot b_N^j \, dS \\
&= \int_{\partial\Omega_0} \sum_{n=0}^M w_n(h_t) \cdot \frac{u_N(x, t - nh_t)}{h_t} \cdot b_N^j \, dS - \int_{\partial\Omega_0} \sum_{n=0}^M w_n(h_t) \cdot \frac{u_N(x, t - (n+1)h_t)}{h_t} \cdot b_N^j \, dS \\
&= \int_{\partial\Omega_0} w_0(h_t) \cdot \frac{u_N(x, t)}{h_t} \cdot b_N^j \, dS + \int_{\partial\Omega_0} \sum_{n=1}^M w_n(h_t) \cdot \frac{u_N(x, t - nh_t)}{h_t} \cdot b_N^j \, dS \\
&\quad - \int_{\partial\Omega_0} \sum_{n=0}^{M-1} w_n(h_t) \cdot \frac{u_N(x, t - (n+1)h_t)}{h_t} \cdot b_N^j \, dS \\
&= \int_{\partial\Omega_0} w_0(h_t) \cdot \frac{u_N(x, t)}{h_t} \cdot b_N^j \, dS \\
&\quad + \int_{\partial\Omega_0} \sum_{n=1}^M (w_n(h_t) - w_{n-1}(h_t)) \cdot \frac{u_N(x, t - nh_t)}{h_t} \cdot b_N^j \, dS \\
&= \int_{\partial\Omega_0} w_0(h_t) \cdot \frac{\sum_{i=1}^N \mu_i(t) b_N^i}{h_t} \cdot b_N^j \, dS \\
&\quad + \int_{\partial\Omega_0} \sum_{n=1}^M (w_n(h_t) - w_{n-1}(h_t)) \cdot \frac{\sum_{i=1}^N \mu_i(t - nh_t) b_N^i}{h_t} \cdot b_N^j \, dS \\
&= \frac{w_0(h_t)}{h_t} \sum_{i=1}^N \left(\int_{\partial\Omega_0} b_N^i \cdot b_N^j \, dS \right) \mu_i(t) \\
&\quad + \sum_{n=1}^M \frac{(w_n(h_t) - w_{n-1}(h_t))}{h_t} \sum_{i=1}^N \left(\int_{\partial\Omega_0} b_N^i \cdot b_N^j \, dS \right) \mu_i(t - nh_t)
\end{aligned}$$

By setting

$$\mathbf{B} := \left(\int_{\partial\Omega_0} b_N^i \cdot b_N^j \, dS \right)_{i,j=1}^N$$

this can be summarized with

$$\frac{w_0(h_t)}{h_t} \cdot \mathbf{B} \cdot \boldsymbol{\mu}(t) + \sum_{n=1}^M \frac{(w_n(h_t) - w_{n-1}(h_t))}{h_t} \cdot \mathbf{B} \cdot \boldsymbol{\mu}(t - n \cdot h_t).$$

At least, the right handside can be written as

$$\boldsymbol{\varphi}(t) := \left(\int_{\Omega \setminus \Omega_0} f \cdot b_N^j dx \right)_{j=1}^N.$$

By summing all the results, we obtain the following linear problem

$$\mathbf{A} \cdot \boldsymbol{\mu}(t) - \frac{w_0(h_t)}{h_t} \cdot \mathbf{B} \cdot \boldsymbol{\mu}(t) - \sum_{n=1}^M \frac{(w_n(h_t) - w_{n-1}(h_t))}{h_t} \cdot \mathbf{B} \cdot \boldsymbol{\mu}(t - n \cdot h_t) = \boldsymbol{\varphi}(t)$$

which we can rewrite as

$$\left(\mathbf{A} - \frac{w_0(h_t)}{h_t} \cdot \mathbf{B} \right) \cdot \boldsymbol{\mu}(t) = \boldsymbol{\varphi}(t) + \sum_{n=1}^M \frac{(w_n(h_t) - w_{n-1}(h_t))}{h_t} \cdot \mathbf{B} \cdot \boldsymbol{\mu}(t - n \cdot h_t). \quad (5)$$

6 Test for exterior BVP

6.1 Analytic solution

To test the algorithm we consider the PDE

$$\begin{cases} -\Delta u = 0 & \text{in } \Omega \times]0; T], \\ u = \frac{4\sqrt{t}}{\sqrt{\pi c_0}} + \log(4) & \text{on } \partial\Omega, \\ \nabla u \cdot -\mathbf{n} = \int_0^t k(t-\tau) \cdot u(\tau) d\tau & \text{on } \partial\Omega_0, \end{cases}$$

where $\mathcal{L}(k) = -\sqrt{sc_0}$.

In the Laplace domain the PDE reads

$$\begin{cases} -\Delta \mathcal{L}(u) = 0 & \text{in } \Omega \times]0; T], \\ \mathcal{L}(u) = \frac{2+\log(4)\sqrt{sc_0}}{\sqrt{sc_0}} \cdot \frac{1}{s} & \text{on } \partial\Omega, \\ \nabla \mathcal{L}(u) \cdot -\mathbf{n} = -\sqrt{sc_0} \cdot \mathcal{L}(u) & \text{on } \partial\Omega_0. \end{cases}$$

We make the Ansatz of separation of variables $\mathcal{L}(u) := u_r(r) \cdot u_\varphi(\varphi)$. A natural Ansatz is then $u_\varphi(\varphi) := \frac{2+\log(4)\sqrt{sc_0}}{\sqrt{sc_0}} \cdot \frac{1}{s}$. The Laplace equation becomes

$$-\frac{1}{r} \frac{\partial}{\partial r} r \frac{\partial}{\partial r} u_r(r) = 0.$$

Its solution is $u_r(r) = A \log(r) + B$. From the Dirichlet boundary condition we find $u_r(2) = 1$ and thus $B = 1 - \log(2)A$. For the impedance boundary condition we have

$$\begin{aligned} 0 &= u_r'(0.5) - \sqrt{sc_0} u_r(0.5) \\ &= 2A - \sqrt{sc_0}(A \log(0.5) + B) \\ &= 2A - \sqrt{sc_0}(-\log(2)A + 1 - \log(2)A) \\ &= A(2 + 2 \log(2)\sqrt{sc_0}) - \sqrt{sc_0} \end{aligned}$$

and thus $A = \frac{\sqrt{sc_0}}{2+2\log(2)\sqrt{sc_0}}$. Then

$$\begin{aligned} \mathcal{L}(u) &= u_r(r) \cdot u_\varphi(\varphi) \\ &= \left(\frac{\sqrt{sc_0}}{2 + 2 \log(2)\sqrt{sc_0}} \log(r) + (1 - \log(2)) \frac{\sqrt{sc_0}}{2 + 2 \log(2)\sqrt{sc_0}} \right) \frac{2 + \log(4)\sqrt{sc_0}}{\sqrt{sc_0}} \cdot \frac{1}{s} \\ &= \frac{\log(r)}{s} + \frac{2 + \log(4)\sqrt{sc_0}}{\sqrt{sc_0}} \cdot \frac{1}{s} - \frac{\log(2)}{s} \\ &= \frac{\log(r)}{s} + \frac{2}{\sqrt{sc_0}} \cdot \frac{1}{s} + \frac{\log(2)}{s}. \end{aligned}$$

This means that

$$u = \log(r) + \frac{4\sqrt{t}}{\sqrt{\pi c_0}} + \log(2).$$

6.2 Mesh Generation

We generate the meshes using uniform refinement.

```
% It generates the meshes for an annulus with internal radius = 0.5 and
% external radius = 2. The initial step size is H0=0.5.
% The meshes are then saved in Coordinates.dat and Elements.dat.

5 function premesh

    BBOX = [-2 -2; 2 2];
    H0=0.5;
    DHANDLE = @(x) dist_diff(dist_circ(x,[0 0],2),dist_circ(x,[0 0],0.5));
10 HHANDLE = @(x) ones(size(x,1),1);
    FIXEDPOS = [];
    DISP = 0;

    Mesh = init_Mesh(BBOX,H0,DHANDLE,HHANDLE,FIXEDPOS,DISP);
15 save_Mesh(Mesh,'Coordinates1.dat','Elements1.dat');

    Mesh = add_Edges(Mesh);
    Loc = get_BdEdges(Mesh);
    Mesh.BdFlags = zeros(size(Mesh.Edges,1),1);
20 Mesh.BdFlags(Loc) = -1;

    Mesh = refine_REG(Mesh,DHANDLE);
    save_Mesh(Mesh,'Coordinates2.dat','Elements2.dat');

25 Mesh = refine_REG(Mesh,DHANDLE);
    save_Mesh(Mesh,'Coordinates3.dat','Elements3.dat');

    Mesh = refine_REG(Mesh,DHANDLE);
    save_Mesh(Mesh,'Coordinates4.dat','Elements4.dat');
30 Mesh = refine_REG(Mesh,DHANDLE);
    save_Mesh(Mesh,'Coordinates5.dat','Elements5.dat');

end
```

6.3 Computation of mass matrix

To compute $\mathbf{B}(i, j) = \left(\int_{\partial\Omega_0} b_N^i \cdot b_N^j dS \right)$ we consider three cases:

- $i = j$,
- $i \neq j$ and the nodes i, j belong to the same edge,
- $i \neq j$ and the nodes i, j don't belong to the same edge.

For the first case

$$\mathbf{B}(i, i) = \frac{1}{3}(\text{lenght}(\partial\Omega_0 \cap \text{supp}(b_N^i))).$$

For the second case

$$\mathbf{B}(i, j) = \frac{1}{6}(\text{lenght}(\partial\Omega_0 \cap \text{supp}(b_N^i) \cap \text{supp}(b_N^j))).$$

For the third case

$$\mathbf{B}(i, j) = 0.$$

```

%Computes the mass matrix but just integrating on the inner boundary.
function M = Mass_on_bdry (Mesh)
5 M=zeros (size (Mesh. Coordinates , 1) , size (Mesh. Coordinates , 1));
  nEdges = size (Mesh. Edges , 1);
  for i = 1:nEdges
10   if (Mesh. BdFlags (i) == -2)
      dist=norm (Mesh. Coordinates (Mesh. Edges (i , 1) , :) - Mesh. Coordinates (Mesh.
          Edges (i , 2) , :));
      M (Mesh. Edges (i , 1) , Mesh. Edges (i , 2)) = dist / 6;
      % the matrix is symmetric
15   M (Mesh. Edges (i , 2) , Mesh. Edges (i , 1)) = M (Mesh. Edges (i , 1) , Mesh. Edges (
          i , 2));
      M (Mesh. Edges (i , 1) , Mesh. Edges (i , 1)) = M (Mesh. Edges (i , 1) , Mesh. Edges (i , 1)
          ) + dist / 3;
20   M (Mesh. Edges (i , 2) , Mesh. Edges (i , 2)) = M (Mesh. Edges (i , 2) , Mesh. Edges (i , 2)
          ) + dist / 3;
  end
end
end

```

6.4 Computation of the numerical solution

Let consider the equation (5). We will solve it for $T = 2$ and $h_t = 0.2$. The equation becomes then a system of equation

$$\left(\mathbf{A} - \frac{w_0(h_t)}{h_t} \cdot \mathbf{B}\right) \cdot \boldsymbol{\mu}(jh_t) = \varphi(jh_t) + \sum_{n=1}^M \frac{(w_n(h_t) - w_{n-1}(h_t))}{h_t} \cdot \mathbf{B} \cdot \boldsymbol{\mu}(jh_t - n \cdot h_t)$$

where $j = 1 : k$ and $k = \frac{T}{h_t}$. Since $\boldsymbol{\mu}(t) = 0$ for $t < 0$ the sum can be taken just until $n = j$. Then the system becomes

$$\left(\mathbf{A} - \frac{w_0(h_t)}{h_t} \cdot \mathbf{B}\right) \cdot \boldsymbol{\mu}(jh_t) = \varphi(jh_t) + \sum_{n=1}^j \frac{(w_n(h_t) - w_{n-1}(h_t))}{h_t} \cdot \mathbf{B} \cdot \boldsymbol{\mu}(jh_t - n \cdot h_t).$$

We see that $\boldsymbol{\mu}(jh_t)$ can be computed iteratively.

```
function [y,h]=prova11

%constant of the kernel
c=10^7;
5 %radius of the internal circle
r=0.5;
%final time
T=2;
%size of h_t
10 h_t=0.2;

k = T/h_t;

%weights of implicit Euler
15 weights=zeros(k+1,1);
for n=1:k+1

    weights(n)=-sqrt(c*h_t)*((-1)^(n-1))*gamma(1/2)/gamma(n-1+1)/gamma(1/2-(
        n-1));
end
20

%set the quadrature rule
QuadRule = P7O6();

25 %load the mesh
Mesh = load_Mesh('Coordinates1.dat','Elements1.dat');
Mesh = add_Edges(Mesh);

%set the flags of points on the exterior boundary to -1 and on the interior
to -2
30 nCoordinates = size(Mesh.Coordinates,1);
U = zeros(nCoordinates,1);
Loc = get_BdEdges(Mesh);
BE = Mesh.Edges(Loc,:);
Mesh.BdFlags = zeros(size(Mesh.Edges,1),1);
35 for i=1:size(Loc,1)
```

```

    if (Mesh.Coordinates(BE(i,1),1)^2 + Mesh.Coordinates(BE(i,1),2)^2 > (r
        ^2+0.5))
40     Mesh.BdFlags(Loc(i))=-1;
    else
        Mesh.BdFlags(Loc(i))=-2;
    end
end
45 %Extract Dirichlet nodes and collects the degree of freedoms
for j = -1

    DEdges = Loc(Mesh.BdFlags(Loc) == j);
50     DNodes = unique([Mesh.Edges(DEdges,1); Mesh.Edges(DEdges,2)]);

    %Compute Dirichlet boundary conditions

    U(DNodes)=log(4);
55 end

    FreeDofs = setdiff(1:nCoordinates, DNodes);

    %assemble the stiffness matrix
60 A = assemMat_LFE(Mesh, @STIMA_Lapl_LFE);

    %assemble the mass matrix on the boundary
    M = Mass_on_bdry(Mesh);

65 %solve the linear system
    U1=zeros(size(Mesh.Coordinates,1),k+1);
    for j=1:k+1
        U1(:,j)=U;
    end
70
    U1(:,1)=solution(Mesh.Coordinates,0,c);

    for j = 1:k
75     Upast=zeros(size(Mesh.Coordinates,1),1);

        for p = 1:j

            Upast = Upast + (weights(p+1)-weights(p))/h_t*M*U1(:,j-p+1);
80     end

        %assemble the load vector
        U(DNodes)=4*sqrt(j*h_t/pi/c)+log(4);
        U1(:,j+1)=U;
85     Fmod=A*U;

        U1(FreeDofs,1+j) = (A(FreeDofs,FreeDofs)...
            -weights(1)/h_t*M(FreeDofs,FreeDofs))\((Fmod(FreeDofs)+Upast(FreeDofs)
            ));
    end
90
    %computes the L2-error
    ErrHandle =@(x) [solution(x,T,c)];
    y = L2Err_LFE(Mesh,U1(:,k+1),QuadRule,ErrHandle);
    h = get_MeshWidth(Mesh);
95
end

```

```

function y=solution(x,t,c)
R=sqrt(x(:,1).^2+x(:,2).^2);
100 y=log(R)+4*sqrt(t/pi/c)+log(2);
end

```

6.5 Collection of the errors

We perform six spacestep refinement and seven timestep refinement and we collect the L^2 error with respect to the analytic solution. The values can be found in the following table.

$h \setminus h_t$	2^0	2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}
0.8468	0.0931	0.1176	0.1291	0.1340	0.1362	0.1371	0.1375
0.4234	0.0404	0.0177	0.0239	0.0282	0.0302	0.0311	0.0316
0.2117	0.0559	0.0190	0.0058	0.0047	0.0062	0.0070	0.0074
0.1059	0.0605	0.0233	0.0088	0.0031	0.0011	0.0012	0.0016
0.0529	0.0616	0.0245	0.0100	0.0041	0.0016	$5.9417 \cdot 10^{-4}$	$0.2659 \cdot 10^{-3}$

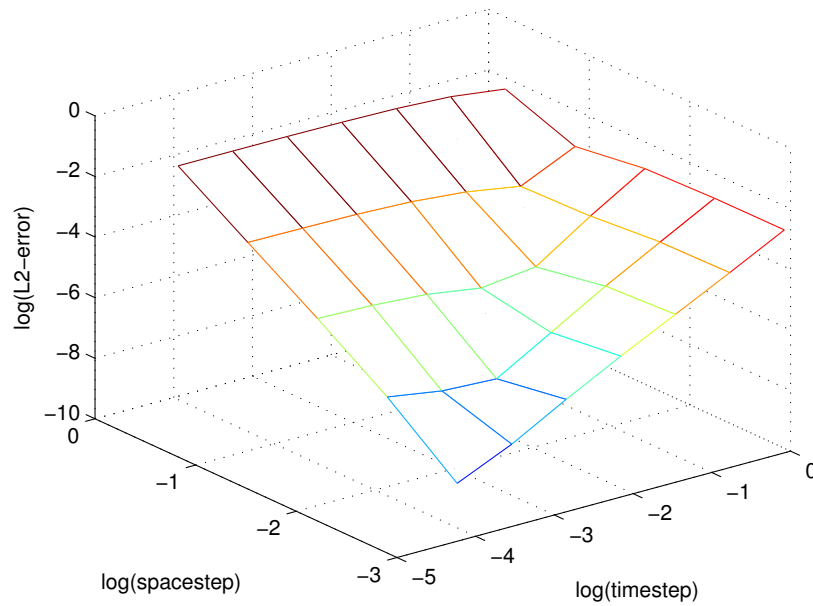


Figure 7: L^2 Error versus time and space.

6.6 Interpretation of the results

The global error of the algorithm is a sum of the convolution quadrature error and of the Lagrangian finite element error. We have led five mesh-refinement and seven time-refinement. As figure 7 suggests, there is an optimal way to choose the time stepsize and the space stepsize. From the table above, we see that the best result are obtained with

h	h_t
0.8468	2^0
0.4234	2^{-1}
0.2117	2^{-3}
0.1059	2^{-4}
0.0529	2^{-6}

The experiment suggests that the optimal ratio should be $\frac{h_t}{h} = 1.2801$.

7 References

1. C. Lubich, *Convolution quadrature and discretized operational calculus. I.*, Numer. Math., 52 (1988), pp. 129-145.
2. C. Lubich, *Convolution quadrature and discretized operational calculus. II.*, Numer. Math., 52 (1988), pp. 413-425.
3. C. Lubich, *Convolution quadrature revisited*, BIT, 44 (2004), pp. 503-514.
4. J. Ostrowski, *Boundary element methods for inductive hardening*, PhD thesis, Fakultät für Mathematik und Physik, Tübingen, November 2002.