Journal of Computational and Applied Mathematics 429 (2023) 115212

Contents lists available at ScienceDirect

Journal of Computational and Applied Mathematics

journal homepage: www.elsevier.com/locate/cam

# *khp*-adaptive spectral projection based discontinuous Galerkin method for the numerical solution of wave equations with memory

Stefano Giani [a,*], Christian Engström [b,c], Luka Grubišić [d]

[a] Department of Engineering, University of Durham, South Road, Durham DH1 3LE, UK
[b] Department of Mathematics, Linnaeus University, Sweden
[c] Centre for Mathematical Sciences, Lund University, Lund, Sweden
[d] Department of Mathematics, Faculty of Science, University of Zagreb, Croatia

ARTICLE INFO

*Article history:*
Received 29 September 2022
Received in revised form 18 February 2023

*Keywords:*
Automatic adaptivity
Inverse Laplace transform
Spectral projection
Wave equation with delay
Discontinuous Galerkin method

ABSTRACT

In this paper, we present an adaptive spectral projection based finite element method to numerically approximate the solution of the wave equation with memory. The adaptivity is not restricted to the mesh (*hp*-adaptivity), but it is also applied to the size of the computed spectrum (*k*-adaptivity). The meshes are refined using a residual based error estimator, while the size of the computed spectrum is adapted using the $L^2$ norm of the error of the projected data. We show that the approach can be very efficient and accurate.



## 1. Introduction

Adaptivity is a key ingredient in modern finite element methods (FEMs), because it improves the accuracy of computations at a reasonable computational cost. The two main types of adaptivity are *h*-adaptivity, which consists in subdividing or "refining" a portion of the elements in the mesh to improve the solution, and *hp*-adaptivity, which also adjusts the order of the polynomials on a portion of the mesh [1–4].

In [5], the authors presented a discontinuous Galerkin (DG) spectral projection-based method to solve the wave equation with memory. In this paper, an adaptive version of the method is presented.

The model problem considered in the paper is the wave equation with memory, which is a source problem. However, the focus of this paper is the adaptive strategy to compute a portion of the spectrum of the eigenvalue problem which is used for spectral projection. A posteriori error estimators for eigenvalue problems [6–9] started to appear much later compared to source problems.

The present paper extends the concept of *hp*-adaptivity adding a new dimension of *k*-adaptivity. *k*-adaptivity is the process of automatically adapting the size *k* of the computed spectrum. The accuracy of a spectral projection method does not depend only on the quality of the mesh as for standard finite element methods applied to steady state problems, but also on the size and quality of the computed spectrum used to construct the numerical solution. If the computed spectrum is too small, the numerical solution might be poor because the number of modes used in the approximation is very limited. On the other hand, if the computed spectrum is too large, the method is computationally very expensive to run and a lot

* Corresponding author.
*E-mail addresses:* stefano.giani@durham.ac.uk (S. Giani), christian.engstrom@math.lth.se (C. Engström), luka.grubisic@math.hr (L. Grubišić).

https://doi.org/10.1016/j.cam.2023.115212

of the modes may not add much to the numerical solution. For these reasons, it is a good idea to adapt the size of the computed spectrum to every problem.

The model problem considered in this paper is the following wave equation problem with memory

$$\begin{aligned} u_{tt}(\mathbf{x},t) - \Delta u(\mathbf{x},t) + \int_0^t \kappa(t-s)\Delta u(\mathbf{x},s)\,ds = q(\mathbf{x},t)\,, && \text{in } \Omega\times(0,T]\,,\\ u(\mathbf{x},t) = 0\,, && \text{on } \partial\Omega\times(0,T]\,,\\ u(\mathbf{x},0) = u_0(\mathbf{x})\,, && \text{in } \Omega\,,\\ u_t(\mathbf{x},0) = v_0(\mathbf{x})\,, && \text{in } \Omega\,, \end{aligned} \tag{1}$$

where $\Omega$ is a bounded connected Lipschitz domain in $\mathbb{R}^2$ and $T > 0$. Problem (1) can be used to model a variety of physical systems like heat transfer with finite propagation speed, systems with thermal memory, viscoelastic materials with a long memory, and acoustic waves in composite media [10,11]. Several authors have considered numerical methods based on the inverse Laplace transform [12,13]. This is understandable because the Laplace transform of a differential equation in time becomes an algebraic equation. However, there are important differences between the diffusion equation and wave equations (with or without memory).

The Laplace transform of the diffusion equation $u_t - \Delta u = f$ with $u(\mathbf{x},0) = u_0(\mathbf{x})$ can be written in the form $A_1(s)\hat{u} = \hat{f} + u_0$, with $\hat{u}$ the Laplace transform of $u$ and where $A_1(s) = s - \Delta$. It is then clear that $A_1(s)^{-1} = (s-\Delta)^{-1}$ can be defined everywhere except at a discrete set of points on the negative real axis (at the eigenvalues of $\Delta$). Hence, $\hat{u} = A(s)^{-1}(\hat{f} + u_0)$ if $s$ is not an eigenvalue of $\Delta$ and we have for $r > 0$ the following representation

$$u(\mathbf{x},t) = \frac{1}{2\pi i}\int_{r-i\infty}^{r+i\infty} e^{st}A_1(s)^{-1}h(\mathbf{x},s)\,ds, \quad h(\mathbf{x},s) = \hat{f}(\mathbf{x},s) + u_0(\mathbf{x}),$$

in terms of the standard (Bromwich) integral. This integral oscillates rapidly, which causes numerical difficulties. Talbot's approach is to deform the standard contour in the Bromwich integral into $\Gamma$, where the contour $\Gamma$ begins and ends in the left half-plane. Let $s : (-\pi,\pi) \to \mathbb{C}$ denote a parametrization of $\Gamma$, such that $\mathrm{Re}\, s(\alpha) \to -\infty$ when $|\alpha| \to \pi$ and assume that $\Gamma$ is a contour around the spectrum of $A_1$. Then, we have a representation of the form

$$u(\mathbf{x},t) = \frac{1}{2\pi i}\int_{-\pi}^{\pi} e^{s(\alpha)t}s'(\alpha)A_1^{-1}(s(\alpha))h(\mathbf{x},s(\alpha))\,d\alpha,$$

where, since $A_1(s)^{-1}$ is defined everywhere of $\mathbb{R}^-$, we have large freedom in choosing $\Gamma$.

The situation changes completely for the wave equation $u_{tt} - \Delta u = f$. The Laplace transform of the wave equation can be written in the form $A_2(s)\hat{u} = \hat{f} + su_0 + v_0$, where $A_2(s) = s^2 - \Delta$. It is then clear that $A_2(s)^{-1} = (s^2-\Delta)^{-1}$ can be defined everywhere except at a discrete set of points on $i\mathbb{R}$. Hence, a contour $\Gamma$ that crosses the imaginary axis at a finite point cannot go around the spectrum of $A_2$. Moreover, it is possible for the corresponding discrete problem to choose a contour around the spectrum of $A_2$, but it will in general be numerically very inefficient. The situation with a memory term (1) is similar to the standard wave equation, but the spectral properties depend on $\kappa$ [14]. In the letter [5] we showed that regardless of the above spectral properties it is possible to apply Talbot's approach to (1).

The paper is organized as follows. In Section 2 we present the numerical methods used to approximate the inverse Laplace transform and the adaptive algorithm. In Section 3 numerical results are presented exploring the challenges of numerically inverting the Laplace transform using Talbot's method and a series of examples of the application of the adaptive methods to the wave equation with memory. Conclusions are stated in Section 4.

## 2. The method

Under the assumption of separation of variables between time and space, problem (1) is solved using the same method as in [5].

For the purpose of clarity, let us consider a simpler problem:

$$\begin{aligned} u_{tt}(\mathbf{x},t) - \Delta u(\mathbf{x},t) = 0\,, && \text{in } (0,1)^2\times(0,T]\,,\\ u(\mathbf{x},t) = 0\,, && \text{on } \partial(0,1)^2\times(0,T]\,,\\ u(\mathbf{x},0) = u_0(\mathbf{x})\,, && \text{in } (0,1)^2\,,\\ u_t(\mathbf{x},0) = v_0(\mathbf{x})\,, && \text{in } (0,1)^2\,. \end{aligned} \tag{2}$$

Problem (2) is simpler than (1) in the sense that there is no memory term, the rhs is zero and the domain is the unit square.

By applying the Laplace transform with respect to time to the PDE in (2) we obtain the time-independent problem

$$s^2\hat{u} - su_0 - v_0 - \Delta\hat{u} = 0\,. \tag{3}$$

The solution $\hat{u}$ can be expressed as a linear combination of the eigenfunctions $\phi_j$ of the Laplace operator with Dirichlet boundary condition

$$\hat{u} = \sum_{j=1}^{\infty} \hat{T}_j(s)\phi_j(\mathbf{x}) = \sum_{j=1}^{\infty} \hat{T}_j(s)\sin(\pi n_j x)\sin(\pi m_j y) ,$$

with $\mathbf{x} \equiv (x, y)$.

Similarly, also the initial data can be projected onto the spectrum of the Laplace operator

$$u_0(\mathbf{x}) = \sum_{j=1}^{\infty} a_j\phi_j(\mathbf{x}) ,$$

$$v_0(\mathbf{x}) = \sum_{j=1}^{\infty} b_j\phi_j(\mathbf{x}) .$$

Substituting the expansions back into (2) we obtain

$$(s^2\hat{T}_j(s) - sa_j - b_j + \lambda_j\hat{T}_j(s))\phi_j(\mathbf{x}) = 0 , \tag{4}$$

where $\lambda_j$ is the eigenvalue of $\phi_j$. From (4) is clear that $\hat{T}_j(s) = \frac{sa_j+b_j}{s^2+\lambda_j}$ for any value $j$. Then, the solution $u$ can be found by applying the inverse Laplace transform

$$u(\mathbf{x}, t) = \sum_{j=1}^{\infty} T_j(t)\phi_j(\mathbf{x}) ,$$

where $T_j$ is calculated from $\hat{T}_j$. Note that $\hat{T}_j$ is singular at $s = \pm i\sqrt{\lambda_j}$. The singularities are important for choosing of contour for the numerical inverse Laplace transform considered in the following subsection. For the more general case (1), the same method can be applied as described in [5]. When $q$, the rhs of problem (1), is not zero, we have to assume that $\hat{q}(\mathbf{x}, s)$, the Laplace transform of $q$, can be expressed as $\hat{q}(\mathbf{x}, s) = \sum_{i=1}^{R} \hat{q}_i(s)g_i(\mathbf{x})$, using a finite number of functions $g_i$ depending only on $\mathbf{x}$, see [5].

### 2.1. The numerical inverse Laplace transform

The inverse Laplace transform is for $r > r_0$, $|f(t)| \leq Ce^{r_0 t}$ given by the Bromwich integral

$$f(t) = \frac{1}{2\pi i}\int_{r-i\infty}^{r+i\infty} \hat{f}(s)e^{st}\, ds. \tag{5}$$

Talbot's approach to numerically computing the inverse Laplace transform is to deform the standard contour in the Bromwich integral. The choice of the contour has implications for the numerical stability of the resulting numerical method. Several contours and quadrature formulas have been proposed; see [15,16] and the references therein. In this paper, we will use Talbot's original contour [17] since it is already widely used in the community. However, the most efficient contour will in particular depend on the spectrum of $\hat{T}$ where the Laplace transform of problem (1) can be written as $\hat{T}(s)\hat{u} = \hat{q}$. Note that we target wave equations with memory terms. The optimal contour will then depend on the particular kernel used in the memory term.

In Section 3.1, numerical instability due to a *poorly* chosen contour is analysed. It is clear that the method can be saved by increasing the numerical precision of the machine. The algorithm used in this paper is based on [18], but the projection approach for the numerical solution of the time-dependent problem is independent on which method to approximate the Bromwich integral is used. Assume that $r = 0$ and consider Talbot's contour parametrized by

$$s(\alpha) = \beta\alpha(\cot(\alpha) + i), \quad -\pi < \alpha < \pi .$$

Furthermore, we assume that $|\hat{f}(s)| \to 0$ uniformly in $\Re s \leq 0$ as $|s| \to \infty$. Then

$$f(t) = \frac{1}{2\pi i}\int_{-\pi}^{\pi} \hat{f}(s(\alpha))e^{s(\alpha)t}s'(\alpha)\, d\alpha. \tag{6}$$

The parameter $\beta$ should, if possible, be chosen such that no singularity of $\hat{f}$ is crossed in the deformation of the original integration path. According to Cauchy's theorem, the integrals (5) and (6) are equal.

In the following, we will use $\beta = 2M/(5t)$, which is taken from [18] and when necessary adjust $M$ such that $s$ encloses all relevant singularities in the inverse Laplace transform. Note that $s(0) = \beta$, $s(\pi/2) = i\beta\pi/2$, $s(3\pi/2) = 3\pi(-1+i)\beta/4$.

Assume that $f$ is real-valued. Then, the Talbot algorithm for numerically approximating $f(t)$ can be written as

$$f_M(t) = \frac{2}{5t}\sum_{k=0}^{M-1} \mathrm{Re}\left(\gamma_k\hat{f}\left(\frac{\delta_k}{t}\right)\right),$$

where for $k = 0, 1, \ldots, M$, we have

$$\delta_0 = \frac{2M}{5}, \quad \delta_k = \frac{2k\pi}{5}\left(\cot\left(\frac{k\pi}{M}\right) + i\right),$$
$$\gamma_0 = \frac{1}{2}e^{\delta_0}, \quad \gamma_k = \left[1 + i\frac{k\pi}{M}\left(1 + \cot^2\left(\frac{k\pi}{M}\right)\right) - i\cot\left(\frac{k\pi}{M}\right)\right]e^{\delta_k}$$

The described algorithm is presented in pseudo-language in Algorithm 1.

**Algorithm 1** The Talbot Algorithm for Inverting the Laplace Transform from [18]

**Input**

$M$ number of terms to use.
Value of $t$ to use in approximating the inverse of the Laplace transform.
Coordinates $\mathbf{x}$ for which invert the Laplace transform.
$F$ function to use in the inverse of the Laplace transform.

**Output**

$f_M(t)$ approximation of the inverse of the Laplace transform of $F$ at $t$ using $M$ terms.

1: Define $\delta_0 := \frac{2M}{5}$ and $\delta_j := \frac{2j\pi}{5}(\cot(j\pi/M) + i)$, for $0 < j < M$ and where $i$ is the imaginary unit.
2: Define $\gamma_0 := \frac{1}{2}e^{\delta_0}$ and $\gamma_j := \big(1 + i(j\pi/M)\big(1 + (\cot(j\pi/M))^2\big) - i\cot(j\pi/M)\big)e^{\delta_j}$, for $0 < j < M$ and where $i$ is the imaginary unit.
3: Compute $f_M(\mathbf{x}, t) := \frac{2}{5t}\sum_{j=0}^{M-1} \mathrm{Re}\big(\gamma_j F(\mathbf{x}, \delta_j/t)\big)$.

To explain how to apply the Talbot method, problem (2) is considered for simplicity. Assuming that $(\lambda_{h,j}, \phi_{h,j})_{j=1}^{K}$ are computed approximations of the $K$ lowest eigenpairs of the Laplace operator with Dirichlet boundary conditions, then we have that for any value $\mathbf{x}$ and $t$, the solution $u(\mathbf{x}, t)$ is approximated using

$$u_h(\mathbf{x}, t) = \sum_{j=1}^{K} \mathrm{Talbot}_M(F_j(\mathbf{x}, s), \mathbf{x}, t)\,, \quad F_j(\mathbf{x}, s) = \hat{T}_{h,j}(s)\phi_{h,j}(\mathbf{x}) \tag{7}$$

where $\mathrm{Talbot}_M$ is Algorithm 1 using $M$ terms and $\hat{T}_{h,j}$ is the numerical approximation of $\hat{T}_j$ in (4). The Talbot method has to be applied every time a new value of $\mathbf{x}$ or $t$ is considered.

Due to its simple structure, the computational complexity of Algorithm 1 can be analysed. The computationally most expensive part is step 3, which becomes dominant for large $M$. It has to be noted that an efficient way to evaluate $F_j(\cdot,\cdot)$ can decrease the overall computational time. Denoting $n_{t,\mathbf{x}}$ the number of distinct pairs $(t, \mathbf{x})$ to be used as input for Algorithm 1, we have that the asymptotic complexity of a sequential implementation of algorithm is $\mathcal{O}(Mn_{t,\mathbf{x}})$. Since the approximation of the solution is computed independently for each pair $(t, \mathbf{x})$, it is straightforward to run multiple instances of Algorithm 1 on different pairs $(t, \mathbf{x})$ at the same time to reduce the computational time. Also, step 3 in Algorithm 1 can be parallelized, but a reduction operation is needed.

The advantage of this method compared to standard FEMs for non-stationary problems is that the solution can be approximated for any value of $t$ without the need to go through all time steps from the initial time to $t$. To compute an approximation of the solution $u$ on the entire domain and for a given value of $t$, the function $u$ is projected onto the $L^2(\Omega)$ space using (7) to sample the values of the function.

### 2.2. The SIPG method

The spectrum of the operator is approximated using the symmetric interior penalty discontinuous Galerkin (SIPDG) method [19]. We assume that the domain $\Omega$ is partitioned into a shape-regular mesh $\mathcal{T}$ consisting of either triangles or quadrilaterals. At most we allow one hanging node per edge. Let us define $\mathbf{h} = \{h_\tau : \tau \in \mathcal{T}\}$ the vector containing all the elemental diameters in the mesh. Denote by $E$ an interior edge of $\mathcal{T}$ if $E = \partial\tau^+ \cap \partial\tau'-$ for two neighbouring elements $\tau^+, \tau^- \in \mathcal{T}$ whose intersection has a positive measure. The set of all interior mesh edges is denoted by $\mathcal{E}_I(\mathcal{T})$.

Analogously, assume that the intersection $E = \partial\tau \cap \partial\Omega$ of the boundary of an element $\tau \in \mathcal{T}$ and $\partial\Omega$ is of positive measure. Then, the set of all boundary mesh edges of $\mathcal{T}$ is denoted by $\mathcal{E}_B(\mathcal{T})$ and we set $\mathcal{E}(\mathcal{T}) = \mathcal{E}_I(\mathcal{T}) \cup \mathcal{E}_B(\mathcal{T})$.

The diameter of an edge $E$ is denoted by $h_E$.

The jumps of an elementwise smooth scalar function $v$ and the averages of an elementwise smooth vector function $\mathbf{g}$ across interior edges of the mesh $\mathcal{T}$ are defined as

$$\{\{\mathbf{g}\}\} = \frac{1}{2}(\mathbf{g}^+|_E + \mathbf{g}^-|_E), \qquad [\![v]\!] = v^+|_E\,\mathbf{n}^+ + v^-|_E\,\mathbf{n}^-,$$

where $\mathbf{n}^+$ and $\mathbf{n}^-$ denote the unit outward normal vectors on the boundary of elements $\tau^+$ and $\tau^-$, respectively. Moreover, $v^+|_E$ and $v^-|_E$ are the trace on $E$ taken from inside either $\tau^+$ and $\tau^-$. The traces are defined similarly for $\mathbf{g}^+|_E$ and $\mathbf{g}^-|_E$.

On a boundary edge $E \in \mathcal{E}_B(\mathcal{T})$, we accordingly set $\{\{\mathbf{g}\}\} = \mathbf{g}$ and $[\![v]\!] = v\mathbf{n}$, with $\mathbf{n}$ denoting the unit outward normal vector on $\partial\Omega$.

Let us define $p_\tau$ the order of the element $\tau$. The order of the elements is allowed to vary across the mesh under the assumption that its variation across neighbour elements is not more than 1. For a mesh edge $E \in \mathcal{E}(\mathcal{T})$, we introduce the face polynomial degree $p_E$ by

$$p_E = \begin{cases} \max\{p_{\tau^+}, p_{\tau^-}\}, & \text{if } E = \partial\tau^+ \cap \partial\tau^- \in \mathcal{E}_I(\mathcal{T}), \\ p_\tau, & \text{if } E = \partial\tau \cap \partial\Omega \in \mathcal{E}_B(\mathcal{T}). \end{cases} \tag{8}$$

The DG finite element space with the degree vector $\mathbf{p} = \{ p_\tau \,:\, \tau \in \mathcal{T} \}$ is defined as

$$S_{\mathbf{p}}(\mathcal{T}) = \{ v \in L^2(\Omega) \,:\, v|_\tau \in \mathcal{H}_{p_\tau}(\tau),\ \tau \in \mathcal{T} \}, \tag{9}$$

for triangular elements, $\mathcal{H}_{p_\tau}(\tau) \equiv \mathcal{P}_{p_\tau}(\tau)$ is the space of polynomials at most of order $p_\tau$ on $\tau$, and for quadrilateral elements, $\mathcal{H}_{p_\tau}(\tau) \equiv \mathcal{Q}_{p_\tau}(\tau)$ is the space of polynomials at most of order $p_\tau$ in each directions.

In view of (1), the spectrum of the operator can be approximated by solving numerically the problem

$$\begin{aligned} -\Delta\phi &= \lambda\phi, && \text{in } \Omega, \\ \phi &= 0, && \text{on } \partial\Omega. \end{aligned} \tag{10}$$

The SIPDG discrete version of the eigenvalue problem (10) is: find $(\lambda_h, \phi_h) \in \mathbb{R} \times S_{\mathbf{p}}(\mathcal{T})$ such that

$$A(\phi_h, v_h) = \lambda_h\, b(\phi_h, v_h), \qquad \forall\, v_h \in S_{\mathbf{p}}(\mathcal{T}), \tag{11}$$

with $\|\phi_h\|_{0,\Omega} = 1$. The bilinear form $A(\cdot,\cdot)$ is given by

$$\begin{aligned} A(w, v) = &\sum_{\tau\in\mathcal{T}} \int_\tau \nabla w \cdot \nabla v \, dx - \sum_{E\in\mathcal{E}(\mathcal{T})} \int_E \Big( \{\{\nabla w\}\} \cdot [\![v]\!] + \{\{\nabla v\}\} \cdot [\![w]\!] \Big)\, ds \\ &+ \sum_{E\in\mathcal{E}(\mathcal{T})} \frac{\gamma p_E^2}{h_E} \int_E [\![w]\!] \cdot [\![v]\!]\, ds, \end{aligned} \tag{12}$$

where the broken gradient operator $\nabla$ is defined elementwise and the parameter $\gamma > 0$ is the interior penalty parameter, and where the bilinear form $b(\cdot,\cdot)$ is the inner-product of $L^2(\Omega)$.

The DG norm can therefore be defined as:

$$\| \phi \|^2_{\mathrm{DG},\mathcal{T}} = \sum_{\tau\in\mathcal{T}} \|\nabla\phi\|^2_{0,\tau} + \sum_{E\in\mathcal{E}(\mathcal{T})} \frac{\gamma p_E^2}{h_E} \|[\![\phi]\!]\|^2_{0,E}\,, \tag{13}$$

where $\|\cdot\|_{0,\tau}$ and $\|\cdot\|_{0,E}$ are respectively the $L^2$ norm on an element $\tau$ and on an edge $E$.

### 2.3. khp-adaptive algorithm

In [5], we proved an error estimator for the solution of the wave equation in $L^2(\Omega)$ for any given value of $t$:

$$\|u(t) - u_h(t)\|_0 \lesssim C_0(t)\varepsilon_0 + C_1(t)\varepsilon_1 + \sum_{i=1}^{R} C_{g_i}(t)\varepsilon_{g_i}\,, \tag{14}$$

where $\varepsilon_0$, $\varepsilon_1$ and $\varepsilon_{g_i}$ for all $i$ are computable quantities defined as:

$$\left\| u_0 - \sum_{j=1}^{K} P_j^h u_0 \right\|_0 \le \varepsilon_0\,, \quad \left\| v_0 - \sum_{j=1}^{K} P_j^h v_0 \right\|_0 \le \varepsilon_1\,, \quad \left\| g_i - \sum_{j=1}^{K} P_j^h g_i \right\|_0 \le \varepsilon_{g_i}\,,$$

where $\|\cdot\|_0$ is the $L^2$ norm on the entire domain $\Omega$ and where $P_j^h$ is the projection operator onto the eigenspace of the computed eigenpair $(\lambda_{h,j}, \phi_{h,j})$. The functions $C_0(\cdot)$, $C_1(\cdot)$ and $C_{g_i}(\cdot)$, for $i$ from 1 to $R$, do not depend on the size of the elements or the order of polynomials used, but they depend on $t$, on the portion of spectrum computed and on $\kappa(\cdot)$, see [5] for more details.

The error estimator (14) inspired the *khp*-adaptive algorithm, Algorithm 2. From (14), the projection errors of the data must be controlled in order to control the error of the solution. From now on, the projection error onto the computed spectrum of a function $w$ is going to be called total error in view of the fact that it is going to be decomposed into parts.

Considering the total error for a function $w$:

$$\underbrace{\left\| w - \sum_{j=1}^{K} P_j^h w \right\|_0^2}_{\text{Total Error}} = \left\| w - \sum_{j=1}^{N} c_{h,j}\phi_{h,j} \right\|_0^2$$

$$\lesssim \underbrace{\left\| \sum_{j=1}^{K}(c_j - c_{h,j})\phi_j + \sum_{j=K+1}^{+\infty} c_j\phi_j \right\|_0^2}_{\text{Truncation Error}} + \underbrace{\left\| \sum_{j=1}^{K} c_{h,j}(\phi_j - \phi_{h,j}) \right\|_0^2}_{\text{Approximation Error}},$$

where $c_{h,j}$ are the coefficients projecting $w$ onto the eigenspaces of the computed eigenpairs $(\lambda_{h,j}, \phi_{h,j})$ and where $c_j$ are the coefficients projecting $w$ onto the eigenspaces of the exact eigenpairs $(\lambda_j, \phi_j)$. The first term on the last line is called truncation error in view of the fact that taking the limit on the mesh refinements:

$$\lim_{h\to 0, p\to+\infty} \left\| \sum_{j=1}^{K}(c_j - c_{h,j})\phi_j + \sum_{j=K+1}^{+\infty} c_j\phi_j \right\|_0 = \left\| \sum_{j=K+1}^{+\infty} c_j\phi_j \right\|_0 .$$

Algorithm 2 is designed to control the total errors for all projected functions and the approximation errors. This is because even if the total errors are what is used to construct the upper bound in (14), good approximations cannot be found without a good approximation of the spectrum to start with. The approximation error can be controlled not only with the $L^2$ norm. In [20], an asymptotically reliable error estimator for the DG norm for problem (11) is presented:

$$\| \phi_j - \phi_{h,j} \|_{\mathrm{DG},\mathcal{T}} \lesssim \eta_{h,j} ,$$

where

$$\eta_{h,j}^2 = \sum_{\tau\in\mathcal{T}} \eta_{j,\tau}^2, \quad \eta_{j,\tau}^2 = \eta_{j,R_\tau}^2 + \eta_{j,F_\tau}^2 + \eta_{j,J_\tau}^2 . \tag{15}$$

The terms in (15) are define as:

$$\eta_{j,R_\tau}^2 = p_\tau^{-2} h_\tau^2 \|\lambda_{h,j}\phi_{h,j} + \Delta\phi_{h,j}\|_{0,\tau}^2 ,$$

$$\eta_{j,F_\tau}^2 = \frac{1}{2} \sum_{E\in\mathcal{E}_I(\tau)} p_E^{-1} h_E \|[\![\nabla\phi_{h,j}]\!]\|_{0,E}^2 ,$$

$$\eta_{j,J_\tau}^2 = \frac{1}{2} \sum_{E\in\mathcal{E}_I(\tau)} \frac{\gamma^2 p_E^3}{h_E} \|[\![\phi_{h,j}]\!]\|_{0,E}^2 + \sum_{E\in\mathcal{E}_B(\tau)} \frac{\gamma^2 p_E^3}{h_E} \|[\![\phi_{h,j}]\!]\|_{0,E}^2,$$

where $\mathcal{E}_I(\tau)$ is the set of faces of the element $\tau$ in the interior of the domain and $\mathcal{E}_B(\tau)$ is the set of faces of the element $\tau$ on the boundary of the domain.

In Algorithm 2, the definition of the total error for a function $w$ is:

$$e_{\mathrm{total},K}(w) = \left\| w - \sum_{j=1}^{K} c_{h,j}\phi_{h,j} \right\|_0 ,$$

and the definition of the approximation error is:

$$e_{\mathrm{approx},K}(w) = \sum_{j=1}^{K} |c_{h,j}|\eta_{h,j} .$$

$e_{\mathrm{approx},K}$ uses the projection coefficients $c_{h,j}$ to focus on those eigenpairs that are more important for the approximation of $w$. In this way, little computational power is used to improve the accuracy of eigenpairs not carrying much of the energy of $w$.

In Algorithm 2, there are two points to exit the algorithm, namely lines 12 and 18. Line 12 is used when the accuracy is reached before exhausting the number of meshes. Line 18 is used when the maximum number of meshes is reached before getting to the required accuracy. The FE space $S_{\mathbf{p}}(\mathcal{T})$ is adapted until the approximation errors for all functions $w_i$ are below the tolerance. At that point, if the total error is still not below the tolerance, the size of the spectrum is increased since clearly more eigenfunctions are needed to approximate the functions $w_i$. The spectrum and projection coefficients computed in Algorithm 2 are used in (7) to compute an approximation of the solution to the wave equation.

The method can also be implemented in a different way called online/offline, designed specifically for solving several times the same wave equation on the same domain for a series of different initial conditions. This version takes advantage of the fact that the spectrum remains the same no matter the initial conditions. The online part consists in using (7) to

**Algorithm 2** *khp*-adapt Algorithm for the Spectrum

**Input**
Initial FE space $S_{\mathbf{p}}(\mathcal{T})$.
Maximum number of adaptive steps $n_{\max}$.
Tolerance tol.
Initial size of the computed spectrum $K$.
Increment size of the compute spectrum $K_{\text{inc}}$.
List of functions $\{w_i\}$ to approximate using the spectrum.

**Output**
Approximated spectrum.
Projection coefficients for each $w_i$.

$n := 1$
**while** $n < n_{\max}$ **do**
  Computing $K$ eigenpairs for the problem.
  **for all** $w_i$ **do**
    Compute the projection coefficients for $w_i$.
    Compute the total errors $e_{\text{total},K}(w_i)$.
    Compute the approximation errors $e_{\text{approx},K}(w_i)$.
  **end for**
  **if** For at least one $w_i$, $e_{\text{approx},K}(w_i) >$ tol **then**
    *hp*-adapt $S_{\mathbf{p}}(\mathcal{T})$ using as error estimator $\eta_h = \sqrt{\sum_{j=1}^{K} \eta_{h,j}^2}$.
  **else if** For all $w_i$, $e_{\text{total},K}(w_i) <$ tol **then**
    Return current spectrum and projection coefficients for each $w_i$.
  **else**
    Increase $K$ by $K_{\text{inc}}$
  **end if**
  $n := n + 1$
**end while**
Return current spectrum and projection coefficients for each $w_i$.

compute an approximation of the solution to the wave equation for any given set of initial conditions. In the online/offline algorithm, the spectrum has to be adapted without knowing what functions $w_i$ are going to be approximated. For this reason, we cannot use any more the above definition of the approximation error $e_{\text{approx},K}$.

In Algorithm 3, only the error estimators from eigenpairs still above the tolerance are used to refine the mesh. If Algorithm 3 is successful, all returned eigenpairs are approximated using an accuracy of at least tol. Algorithm 3 also returns the error estimators $\eta_{h,j}$ for the entire computed spectrum to control the error during the online part. In the online part, the functions $w_i$ to approximate are know, therefore, using the values $\eta_{h,j}$, the approximation errors $e_{\text{approx},K}(w_i)$ and the total errors $e_{\text{total},K}(w_i)$ can be computed. This can be used to check if the computed spectrum is delivering a good approximation of the functions $w_i$ and to decide whether to accept the computed solutions.

## 3. Numerics

All the simulations are run using ARPACK [21] to compute the eigenpairs and MUMPS [22] to solve the linear systems as part of the iterative loop of ARPACK. For simplicity, all simulations are run on the sequential version of the code, i.e. only one core of the CPU has been used. For consistency, all simulations are run on a 100 MHz Intel Xeon CPU E5-2620 with 64GiB of DDR3 RAM.

### 3.1. The Talbot Algorithm

In [18], it is stated that Algorithm 1 might push to the limit the precision of the machine for large values of $M$. In this section, numerical instability due to a *poorly* chosen contour is analysed. In exact arithmetic, the accuracy of the Talbot method increases with $M$ and decreases with $t$. For a fixed $t$, there is a minimum value of $M$ to get accurate results. In order to keep the results accurate when $t$ increases, $M$ might have to be increased. In practice, small values of $M$ can deliver accurate results in double precision. However, to compute accurately values of the solution for larger $t$, $M$ has to be increased and this might lead to bad accuracy due to the limitation of double precision. In order to better understand this issue, we tested the Talbot method on the wave equation on the unit square $(0, 1)^2$ without memory with initial velocity zero and initial displacement equals to a $u_0(\mathbf{x}) = u(\mathbf{x}, 0) = \sin(\pi x)\sin(\pi y)$. The exact

**Algorithm 3** *khp*-Adapt Algorithm for the Spectrum for the Offline version

**Input**
Initial FE space $S_{\mathbf{p}}(\mathcal{T})$.
Maximum number of adaptive steps $n_{\max}$.
Tolerance tol.
Size of the computed spectrum $K$.
**Output**
Approximated spectrum.
Error estimators $\eta_{h,j}$.

$n := 1$
**while** $n < n_{\max}$ **do**
  Computing $K$ eigenpairs for the problem.
  $\mathcal{R} := \emptyset$.
  **for all** Eigenpair $j$ with $\eta_{h,j} >$ tol **do**
    Add $j$ to $\mathcal{R}$
  **end for**
  **if** $\mathcal{R}$ not empty **then**
    $hp$-adapt $S_{\mathbf{p}}(\mathcal{T})$ using as error estimator $\eta_h = \sqrt{\sum_{j\in\mathcal{R}} |c_{h,j}|\eta_{h,j}^2}$.
  **else**
    Return current spectrum and current $\eta_{h,j}$ for the entire computed spectrum.
  **end if**
  $n := n + 1$
**end while**
Return current spectrum and current $\eta_{h,j}$ values for the entire computed spectrum.

solution is $u(\mathbf{x}, t) = \cos(\sqrt{\lambda}t)\sin(\pi x)\sin(\pi y)$, where $\lambda = 2\pi^2$. Therefore, the function passed to the Talbot method is $\hat{T}(s) := 1/(s^2 + \lambda)s$ for the approximated solution. We tested the impact of different implementations of the Talbot method in different precisions: double precision, quad precision, and arbitrary precision using the algorithm in [23]. In Fig. 1, we reported the relative error on the value of the solution at $(0.5, 0, 5)$ for different values of $t$ and $M$. The issue is very clear in Fig. 1(a) where double precision is used. For the same value of $t$, the error increases with $M$. This is clearly not supposed to happen and it does not happen in exact arithmetic. Exactly the opposite is supposed to happen, i.e. the accuracy should increase as $M$ is increasing. But, because of the limitations of double precision to cope with the terms in the series in the Talbot method, accuracy is poor for large enough $M$. The worse case is for $M = 100$ or higher, where for no values of $t$, the error is small. Switching to quad precision, Fig. 1(b), the situation is not improved. In Fig. 1(c), 1200 digits are used to compute the Talbot method, and errors behave correctly. For the same $t$, we see smaller errors increasing $M$ and even for large $t$, there are values of $M$ that deliver good results. It is important to notice that in all simulations, only the Talbot method has been implemented in different precisions. Terms used in the method like the projection of $u_0$ and the eigenvalues are still computed in double precision only. This is because it is not feasible to assume that for real problems, the entire adaptive algorithm can be implemented and run in any precision.

In Fig. 2, the same tests have been repeated for an initial solution $u_0$ equal to the 200th lowest eigenpair of the Laplace operator. The higher frequency of the solution is amplifying the issues. To keep a good accuracy for large $t$, more than 100 terms are needed. This is not a problem since we are far away from the limit of the used arbitrary precision. In conclusion, the Talbot algorithm may be unstable in double precision for large $t$ for problems of engineering interest. However, the possibility to switch to arbitrary precision removes the issue. In the worst-case scenario, the arbitrary precision can be pushed much higher than the 1200 digits used here.

### 3.2. Problem on the unit circle without memory

In this example, the domain is a disk of radius 1 centred in the origin with homogeneous Dirichlet boundary conditions and we assume zero initial velocity and initial displacement

$$u_0(\mathbf{x}) = u(\mathbf{x}, 0) = J_0(\alpha|\mathbf{x}|)\,,$$

with $J_0$ denoting the Bessel function of the first kind and $\alpha$ the first zero of $J_0$. We also assume $q$ and $\kappa$ in (1) to be zero. Under these assumptions, the analytical solution is

$$u(\mathbf{x}, t) = J_0(\alpha|\mathbf{x}|)\cos(\alpha t)\,.$$

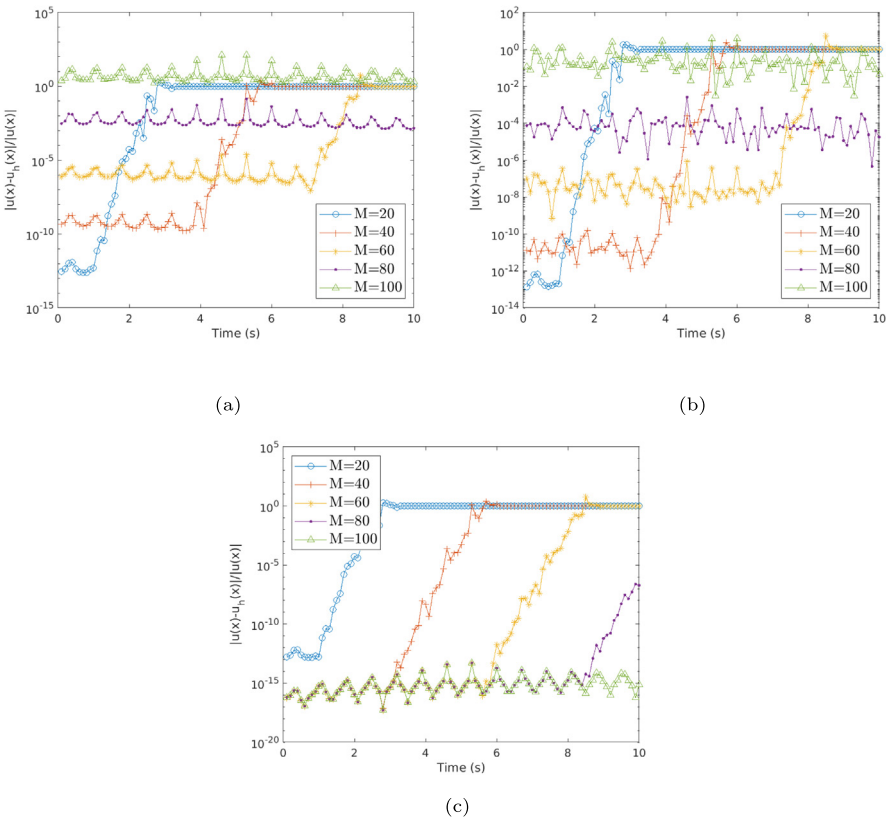


**Fig. 1.** Precision of the Talbot algorithm (a) in double precision, (b) in quad precision and (c) in arbitrary precision with 1200 digits for a problem involving the lowest eigenpair of the Laplace operator.

For any computed eigenpair $(\lambda_{h,j}, \phi_{h,j})$, we can define the function

$$F(\mathbf{x}, s) = \hat{T}_j(s)\phi_{h,1}(\mathbf{x}) = \frac{1}{s^2 + \lambda_j} s P_j^h u_0(\mathbf{x}) \,,$$

where $P_j^h$ is the projection operator on the eigenspace of the $j$th computed eigenpair.

To correctly approximate the shape of the domain, the transfinite interpolation method [24] is used to bend the edges of the elements to exactly match the boundary of the domain.

The focus of this test is on the decay of the total errors $e_{\text{total},K}(u_0)$ using adaptivity. $h$-adaptivity is compared to $hp$-adaptivity using Algorithm 2 to adapt the spectrum targeting $u_0$, i.e. $w_1 \equiv u_0$ in Algorithm 2. The function $u_0$ belongs to the eigenspace of the lowest eigenpair, therefore, the lowest computed eigenpair is the most important in this example. For this reason, the size of the spectrum $K$ is not increased from its initial value of 5.

In Fig. 3, we can see the comparison for the total error using different adaptive schemes. Clearly, $hp$-adaptivity is much better than just $h$-adaptivity.

For each approximation of the spectrum, we compute the relative error of the approximated solution $u_h$ at the origin with $t = 1.0$ using the Talbot method, Algorithm 1, with $M = 35$. In Fig. 4, the comparison between the two adaptive schemes is presented. Clearly, better accuracy for the total error translates into a better approximation for $u_h$. This is further confirmed by Fig. 5 where the ratio between the total error for $u_0$ and the relative error for $u_h$ are presented. The two errors look to be linearly dependent no matter what adaptive scheme is used. Fig. 5 seems to suggest that the total error can also be used to control point-wise error. This is very interesting because it seems to extend the result (14) which holds for the $L^2$ norm of the error.

In Tables 1 and 2, the cumulative time for the most computational demanding parts of Algorithm 2 are reported for the two adaptive strategies. It is interesting to notice that even if the total computational time is similar for both $h$- and $hp$-adaptivity, the delivered accuracy is vastly different, see Figs. 4 and 5. Also, the most computationally demanding tasks are different for the two strategies. For $h$-adaptivity, most of the time is spent computing the residuals due to a large number of elements, while for $hp$-adaptivity, the approximation of the spectrum is the most demanding task.

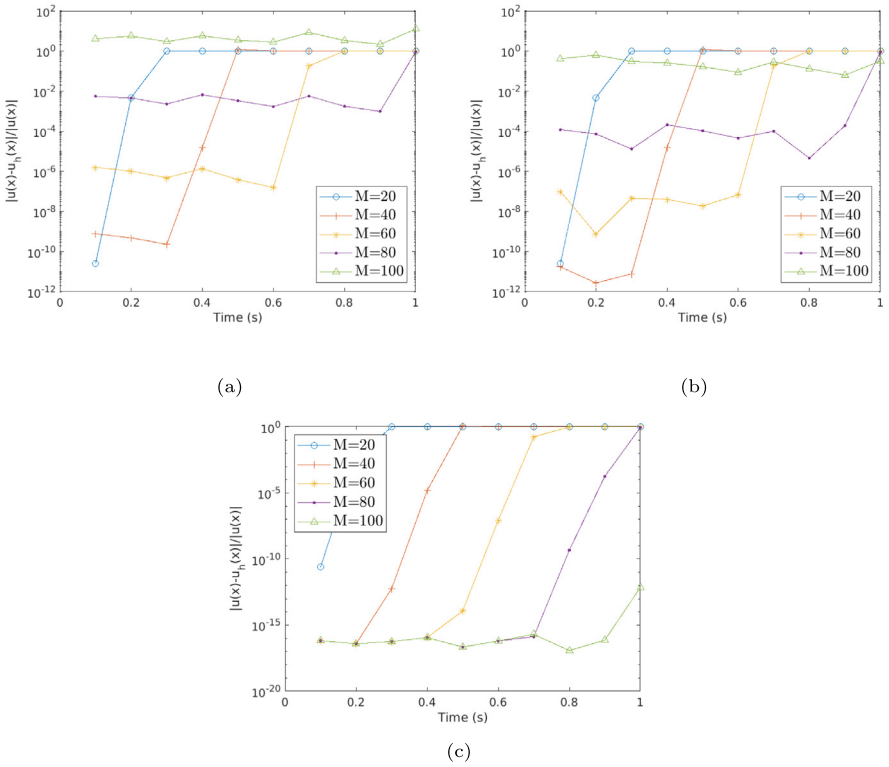


**Fig. 2.** Precision of the Talbot algorithm (a) in double precision, (b) in quad precision and (c) in arbitrary precision with 1200 digits for a problem involving the 200th lowest eigenpair of the Laplace operator.

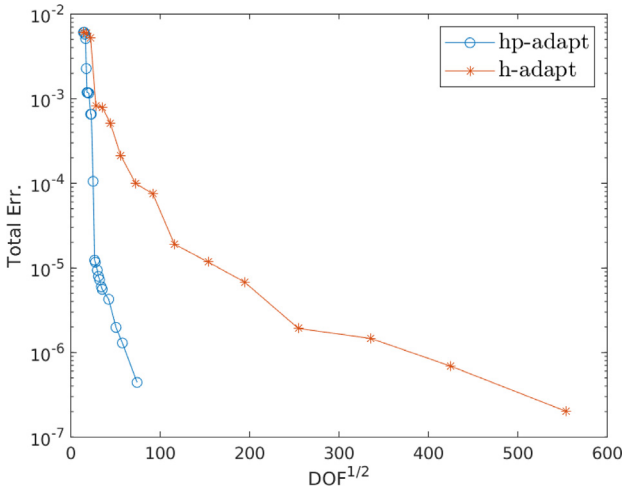


**Fig. 3.** Total error for the adaptive methods for the problem on the unit disk with no memory.

### 3.3. Problem on the unit circle with memory

In this example, we consider the same setting as in the previous example except that this time we have that $\kappa(t) = \frac{1}{2}e^{-t}$. This implies that the analytical solution is

$$u(\mathbf{x}, t) = (-t^2e^{-t} + 1)J_0(\alpha|\mathbf{x}|) ,$$

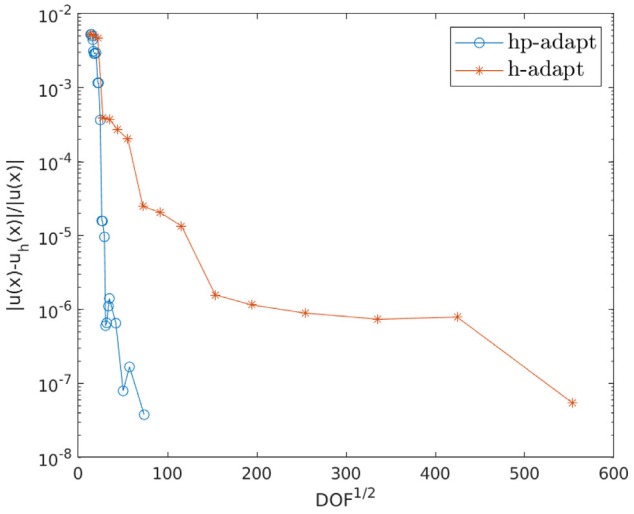


**Fig. 4.** Relative error for the adaptive methods for the problem on the unit disk with no memory.

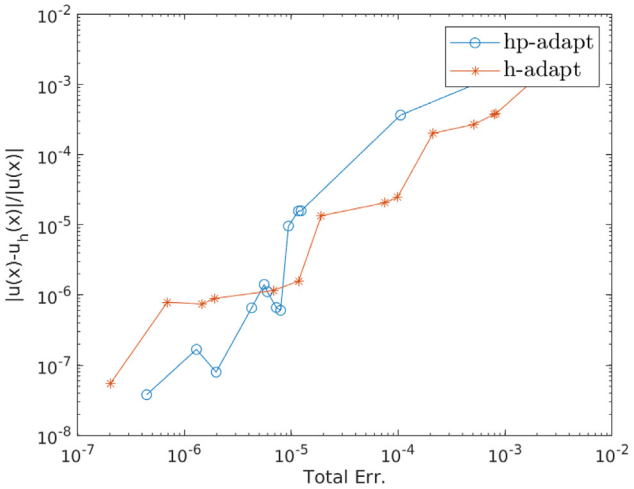


**Fig. 5.** Ratios between the total and relative error for the problem on the unit disk with no memory.

**Table 1**
Timing of the most demanding tasks in Algorithm 2 using *h*-adaptivity.

| Total computational time 1082.67 s | | |
|---|---|---|
| Task | Cumulative time (s) | Cumulative time (%) |
| **Calculation of the residuals** | **771.24** | **71.24** |
| Mesh adaptation | 136.94 | 12.65 |
| Approximation of the spectrum | 135.91 | 12.55 |

**Table 2**
Timing of the most demanding tasks in Algorithm 2 using *hp*-adaptivity.

| Total computational time 1636.90 s | | |
|---|---|---|
| Task | Cumulative time (s) | Cumulative time (%) |
| **Approximation of the spectrum** | **929.15** | **56.76** |
| Calculation of the residuals | 392.29 | 23.97 |
| Mesh adaptation | 274.33 | 16.76 |

and

$$q(\mathbf{x}, t) = -t^2 e^{-t} + 1 \, .$$

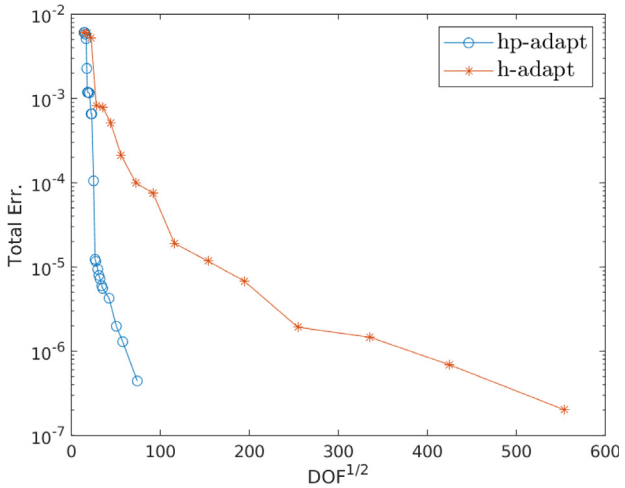


**Fig. 6.** Total error for the adaptive methods for the problem on the unit disk with memory.

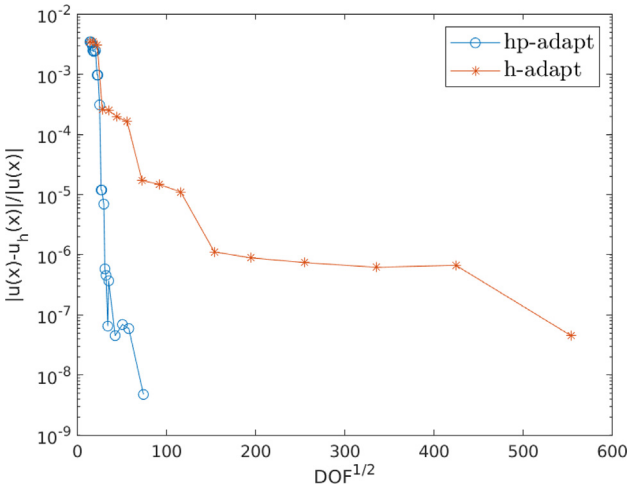


**Fig. 7.** Relative error for the adaptive methods for the problem on the unit disk with memory.

For any computed eigenpair $(\lambda_{h,j}, \phi_{h,j})$, we can define the function

$$F_j(\mathbf{x}, s) = \hat{T}_j(s)\phi_{h,j}(\mathbf{x}) = \frac{1}{s^2 + \mu_j(1-\hat{\kappa}(s))}\left(\hat{q}(s)s^2 P_j^h u_0(\mathbf{x}) - (\hat{\kappa}(s)-1)\hat{q}(s)\lambda_{h,j}P_j^h u_0(\mathbf{x})\right),$$

where

$$\hat{q}(s) = \frac{1}{s} - \frac{2}{(s+1)^3}, \quad \hat{\kappa}(s) = \frac{1}{2(s+1)},$$

are respectively the Laplace transforms of $q$ and $\kappa$.

For this example $K = 5$ which is not changed during the simulation because the solution is proportional to the first eigenfunction.

In Figs. 6–8 the behaviours of the total error, the relative error at the origin with $t = 1$ and their ratio are presented. The presence of memory does not affect the results much.

In Tables 3 and 4, the cumulative time for the most computational demanding parts of Algorithm 2 are reported for the two adaptive strategies. The same observations discussed in the previous example are applicable here.

In Tables 3 and 4, the inverse of the Laplace transform is calculated in double precision. The reported running times have to be interpreted in view of the fact that the delivered accuracy is very high. In view of the results in Section 3.1, in Table 5 the computational time to run Algorithm 1 on one pair of values $(\mathbf{x}, t)$ and with $M = 35$ is reported for

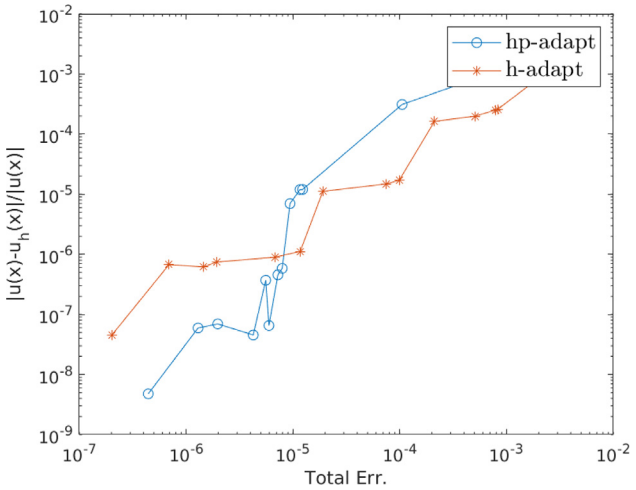


**Fig. 8.** Ratios between the total and relative error for the problem on the unit disk with memory.

**Table 3**
Timing of the most demanding tasks in Algorithm 2 using *h*-adaptivity.

| Total computational time 1045.62 s | | |
|---|---|---|
| Task | Cumulative time (s) | Cumulative time (%) |
| **Calculation of the residuals** | **738.59** | **70.64** |
| Mesh adaptation | 134.53 | 12.87 |
| Approximation of the spectrum | 134.20 | 12.83 |

**Table 4**
Timing of the most demanding tasks in Algorithm 2 using *hp*-adaptivity.

| Total computational time 1617.08 s | | |
|---|---|---|
| Task | Cumulative time (s) | Cumulative time (%) |
| **Approximation of the spectrum** | **924.24** | **57.16** |
| Calculation of the residuals | 375.89 | 23.25 |
| Mesh adaptation | 275.41 | 17.03 |

**Table 5**
Timing to run Algorithm 1 in different precisions. 1200 digits is the default accuracy in the library [23].

| Precision | | Time (s) |
|---|---|---|
| Double | | 4.00e−4 |
| Quadruple | | 1.20e−3 |
| Precision | Digits | Time (s) |
| Arbitrary | 34 | 2.00e−1 |
| Arbitrary | 100 | 2.91e−1 |
| Arbitrary | 500 | 1.29 |
| Arbitrary | 1200 | 4.96 |

different accuracies. Arbitrary precision makes the code much slower especially considering that 34 digits is about the same precision as quadrupole precision. The main reason for the discrepancy is the fact that quad precision is supported by the hardware of modern CPUs while arbitrary precision is done in software.

### *3.4. Error estimator for the wave equation*

In [5] an asymptotic error estimator for the error $\|u - u_h\|_0$ in the $L^2$ norm was presented for the spectral projection based method without adaptivity, see Theorem 2 in [5]. The following results show the accuracy of the error estimator in the adaptive case. The problem considered here is the wave equation with memory on the square domain $(0, 1)^2$ with homogeneous Dirichlet boundary conditions and with zero initial velocity and initial displacement

$$u_0(\mathbf{x}) = u(\mathbf{x}, 0) = \sin(2\pi x)\sin(2\pi y)\ ,$$

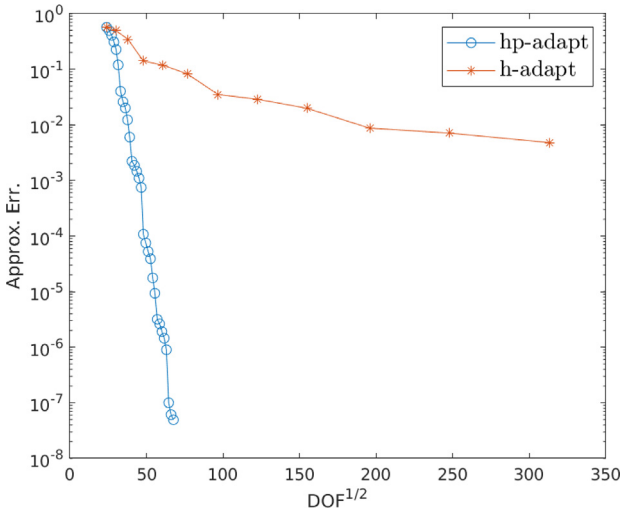


**Fig. 9.** Approximation error for the adaptive methods for the problem on the unit square with memory.

**Table 6**
Timing of the most demanding tasks in Algorithm 2 using *h*-adaptivity.

| Total computational time 1421.07 s | | |
|---|---|---|
| Task | Cumulative time (s) | Cumulative time (%) |
| **Talbot method** | **1217.05** | **85.64** |
| Approximation of the spectrum | 95.90 | 6.74 |
| Calculation of the residuals | 77.34 | 5.44 |

We also assume that $\kappa(t) = \frac{1}{2}e^{-t}$ and $q(\mathbf{x}, t) = -t^2e^{-t} + 1$. Under these assumptions, the analytical solution is

$$u(\mathbf{x}, t) = (-t^2e^{-t} + 1)\sin(2\pi x)\sin(2\pi y)\ .$$

For any computed eigenpair $(\lambda_{h,j}, \phi_{h,j})$, we can define the

$$\begin{aligned} F_j(\mathbf{x}, s) &= \hat{T}_j(s)\phi_{h,j}(\mathbf{x}) \\ &= \frac{1}{s^2 + \mu_j(1 - \hat{\kappa}(s))}\left(\hat{q}(s)s^2P_j^h u_0(\mathbf{x}) - (\hat{\kappa}(s) - 1)\hat{q}(s)\lambda_{h,j}P_j^h u_0(\mathbf{x})\right)\ , \end{aligned}$$

where

$$\hat{q}(s) = \frac{1}{s} - \frac{2}{(s+1)^3}\ , \quad \hat{\kappa}(s) = \frac{1}{2(s+1)}\ ,$$

are respectively the Laplace transforms of $q$ and $\kappa$.

To adapt the spectrum, we used Algorithm 2 with tol $= 1e-50$, $n_{\max} = 30$, $K = 10$, $K_{\text{inc}} = 10$ and $w_1 \equiv u_0$. The values of $n_{\max}$ and tol are chosen in such a way that the tolerance is never reached before running out of meshes. In particular, tol is so small that is unreachable in double precision using 30 meshes, giving full control on the number of meshes used in the simulation. The initial mesh is structured consisting of 64 square elements and the initial order of polynomials is two. The function $u_0$ belongs to the eigenspace of the fourth lowest eigenpair, for this reason, the size of $K$ remains equal to 10 and it is not increased by Algorithm 2 during the simulation.

In the following figures, the performances of Algorithm 2 using either *h*-adaptivity or *hp*-adaptivity are compared.

In Fig. 9, the approximation error $e_{\text{approx},K}(u_0)$ for the different adaptive scheme is presented. Clearly, the *hp*-adaptive method outperforms the *h*-adaptive method.

Similar conclusions can be drawn by looking at the comparison for the total error $e_{\text{total},K}(u_0)$ in Fig. 10.

The error estimator in [5] bounds the $L^2$ of the error for the wave equation. The Talbot method, Algorithm 1, with $M = 35$, is then used to compute an approximation $u_h$ of the solution for $t = 1$ on the entire domain. In Fig. 11, for each approximation of the spectrum, we compute the Error of the approximated solution $u_h$. As before, there is a clear gap between the performances of the two adaptive schemes, but more interesting is the fact that the error estimator is very accurate in both schemes.

In Tables 6 and 7, the cumulative time for the most computational demanding parts of Algorithm 2 are reported for the two adaptive strategies. Compared to the previous examples, here the two strategies have different run times. The

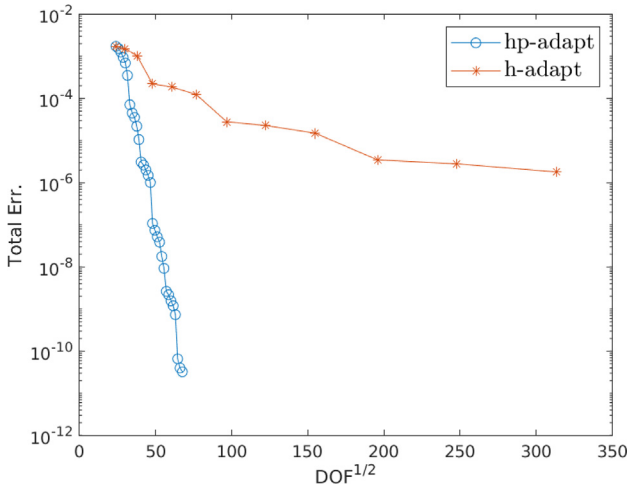


**Fig. 10.** Total error for the adaptive methods for the problem on the unit square with memory.

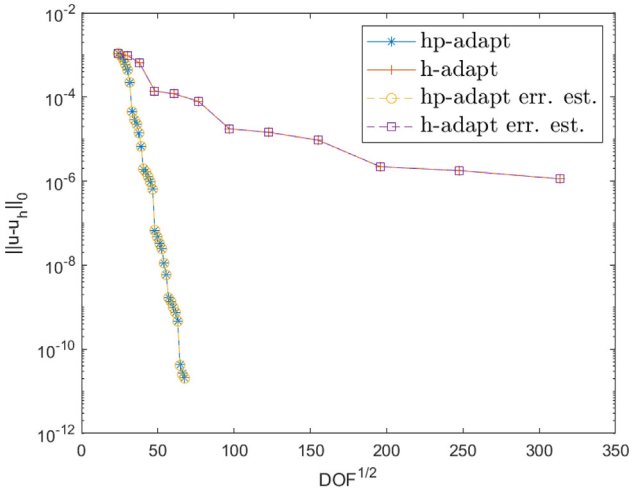


**Fig. 11.** Error and error estimator values for the adaptive methods for the problem on the unit square with memory.

**Table 7**
Timing of the most demanding tasks in Algorithm 2 using *hp*-adaptivity.

| Total computational time 125.48 s | | |
|---|---|---|
| Task | Cumulative time (s) | Cumulative time (%) |
| **Approximation of the spectrum** | **82.79** | **65.98** |
| Mesh adaptation | 19.12 | 15.23 |
| Calculation of the residuals | 11.6 | 9.27 |

*hp*-adaptivity is about ten times faster compared to the *h*-adaptivity delivering also much better results. For the *h*-adaptive method, the most demanding task is the Talbot method. This is understandable since to compute the $L^2(\Omega)$ of the error for $t = 1$, the solution is calculated on all integration points for all elements, resulting in a very large number of calls to the Talbot method on a fine mesh. As in previous examples, the approximation of the spectrum is the most demanding task for the *hp*-adaptive method. In contrast, the inversion of the Laplace transform takes only 2.16s (1.72%) of the time for the *hp*-adaptive method.

### 3.5. *k*-adaptivity

In this next example, the focus is on the *k*-adaptivity. In order to do that, the problem is designed to include a high number of eigenpairs. In practice, this is rarely the case since for most applications only a limited number of frequencies

**Table 8**
Timing of the most demanding tasks in Algorithm 2.

| Task | Cumulative time (s) | Cumulative time (%) |
|---|---|---|
| **Approximation of the spectrum** | **6648.08** | **92.65** |
| Calculation of the residuals | 400.59 | 5.58 |
| Projection of the spectrum | 50.98 | 0.71 |

have to be considered. However, in this section, we want to push the algorithm to the limit. The problem considered here is the wave equation with memory on the square domain $(0, 1)^2$ with homogeneous Dirichlet boundary conditions and with zero initial velocity and initial displacement

$$u_0(\mathbf{x}) = u(\mathbf{x}, 0) = (xy)^2(1-x)^2(1-y)^2 \,,$$

We also assume that $\kappa(t) = \frac{1}{2}e^{-t}$ and $q(\mathbf{x}, t) = -t^2e^{-t} + 1$. Under these assumptions, the analytical solution is

$$u(\mathbf{x}, t) = (-t^2e^{-t} + 1)(xy)^2(1-x)^2(1-y)^2 \,.$$

For any computed eigenpair $(\lambda_{h,j}, \phi_{h,j})$, we can define the function

$$\begin{aligned} F_j(\mathbf{x}, s) &= \hat{T}_j(s)\phi_{h,j}(\mathbf{x}) \\ &= \frac{1}{s^2 + \mu_j(1-\hat{\kappa}(s))}\left(\hat{q}(s)s^2P_j^h u_0(\mathbf{x}) - (\hat{\kappa}(s)-1)\hat{q}(s)\lambda_{h,j}P_j^h u_0(\mathbf{x})\right) \,, \end{aligned}$$

where

$$\hat{q}(s) = \frac{1}{s} - \frac{2}{(s+1)^3} \,, \quad \hat{\kappa}(s) = \frac{1}{2(s+1)} \,,$$

are respectively the Laplace transforms of $q$ and $\kappa$.

To adapt the spectrum, we used Algorithm 2 with *hp*-adaptivity and with $\mathrm{tol} = 1e-6$, $n_{\max} = 100$, $K = 50$, $K_{\mathrm{inc}} = 50$ and $w_1 \equiv u_0$. The initial mesh is structured consisting of 64 square elements and the initial order of polynomials is two. In contrast to previous examples, the function $u_0$ does not belong to any eigenspace but it takes contributions from many of them. For this reason, Algorithm 2 expands the size of the computed spectrum trying to capture enough of $u_0$ to reach the tolerance, see line 14 in Algorithm 2.

In Fig. 12, the behaviour of the total error in terms of the size of the computed spectrum and the number of DOF is presented. Looking at Fig. 12(a), it is clear that the decay of the error is linked to the number of eigenpairs used to approximate $u_0$. For the first few meshes, $K$ is not changed. Instead Algorithm 2 is refining the mesh. The plateaus in Fig. 12(b) correspond to those iterations in the main loop of Algorithm 2 when the method is refining the mesh rather than increasing $K$. In Fig. 12(c), the results are reported in terms of DOF times the number of eigenpairs computed which is interesting since $K$ is adapted during the simulation.

In Table 8, the cumulative time for the most computational demanding parts of Algorithm 2 are reported for this example. The approximation of the spectrum, line 3 in Algorithm 2, is clearly the bottleneck of the method in the current implementation. The remaining part of the Algorithm 2 takes less than 8% of the time to run the code.

### 3.6. Concentrated peak

In this next example, the method is applied to a problem without memory and with initial data concentrated in a small region in the domain $(0, 1)^2$

$$u_0(\mathbf{x}) = u(\mathbf{x}, 0) = e^{\frac{-(\mathbf{x}-\mathbf{x}_0)}{2c^2}} \sin(\pi x)\sin(\pi y) \,,$$

with $\mathbf{x} \equiv (x, y)$, $\mathbf{x}_0 \equiv (0.5, 0.5)$ and with $c > 0$. The exponential is multiplied by the function $\sin(\pi x)\sin(\pi y)$ to enforce $u_0 = 0$ on the boundary of the domain. We also assume $q$ and $\kappa$ in (1) to be zero. The solution of this problem is unknown, therefore the results are compared to a reference solution $u$ computed using the projection method on a very fine structured mesh of 4225 elements with order of polynomials 4 using $K = 500$ and for $t = 1$. In Table 9, we reported the $L^2$ norm of the error for $t = 1$ using the projection method on a structured mesh of 1089 elements with order of polynomials 3 for different choices of $c$ and $K$. Clearly, the strength of the peak impacts the convergence of the method in terms of both DOF and $K$.

**Remark 1.** We finish the discussion on *k*-adaptivity by recalling the paper(s) by S. Sauter [25], and the references therein.[1] The goal of the series of papers was to ascertain how many eigenvalues can be trusted for a given FEM space. Or better

[1] See also https://www.math.uzh.ch/fileadmin/math/preprints/17_07_rev.pdf

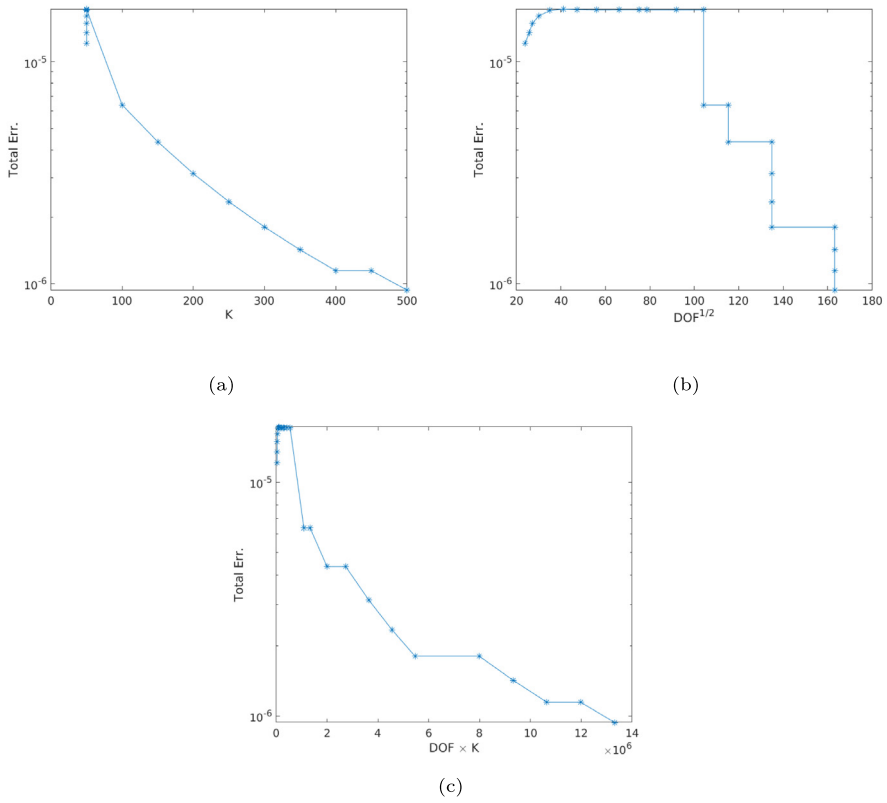


**Fig. 12.** Decay of the total error (a) in terms of the size $K$ of the computed spectrum and (b) in terms of DOF and (c) in terms of DOF times the number of eigenpairs computed on each mesh.

**Table 9**
Error measured in the $L^2$ norm for $t = 1$.

| $c$ | $K$ | $\|u - u_h\|_0$ |
|---|---|---|
| 0.1 | 50 | 7.9527E−3 |
| 0.1 | 100 | 4.7307E−4 |
| 0.1 | 200 | 2.4513E−6 |
| 0.05 | 50 | 3.9207e−2 |
| 0.05 | 100 | 1.5648e−2 |
| 0.05 | 200 | 3.3528e−3 |
| 0.01 | 50 | 1.7146e−2 |
| 0.01 | 100 | 1.6512e−2 |
| 0.01 | 200 | 1.5508e−2 |

to say, how far are we allowed to go with $k$. To avoid the technical details, we quote the result [25, Remark 6.1] in the case of $p = 1$ and using $h$-adaptivity. In this case, only eigenvalue approximations computed from the space whose $h < \lambda_k^{-1}$ can be trusted to be approximate eigenvalues of the Laplace operator (or stated in another way, to have a physical meaning). In [25, Remark 6.1, Table 2] there are bounds related to other choices of the polynomial degree $p$. This should be interpreted in a way that this paper does not give an asymptotic result (in $k$), but rather a practical algorithm that is limited by the size of the finite element space (and the associated mass and stiffness matrices) which can be fitted into the memory and subsequently the class of functions which can be resolved by such a piecewise polynomial space.

## 4. Conclusions

We have presented an adaptive numerical method for the wave equation with memory. The proposed method projects the data on the span of approximated eigenvectors. Both the meshes used to approximate the eigenvectors and the portion of the spectrum approximated by the method are automatically adapted to improve efficiency.

The proposed method can be used for 3D problems without modifications, the only difference is that a 3D eigenvalue problem must be solved. The Talbot method (Algorithm 1) is the same for all numbers of dimensions. The computational

cost of running Algorithm 1 for given pair $(\mathbf{x}, t)$ is determined by the number of terms $M$ and not the number of dimensions. The adaptive method (Algorithm 2) applies to 3D problems because the error estimator 2 works in both 2D and 3D.

The method can also be extended to variable diffusion coefficient in space since residual based error estimators, like (15), are reliable as shown in [26,27].

As discussed in Section 2, the method can be divided into an offline part approximating the spectrum only once and an online part approximating the solution of the problem for multiple values of $t$ and $\mathbf{x}$ and possibly for various initial conditions. As seen in the examples, the computation of spectra can be computationally demanding compared to the application of the Talbot method. Therefore, it makes sense to relegate the offline part to HPC machines and the online on desktop machines.

The proposed method is particularly efficient for applications where the spectrum computed in the offline part can be reused several times. For example, the optimization of initial conditions or the right-hand-side of problem (1) to achieve certain aims can be solved by computing the spectrum only once and then running the optimization only on the online part. Different initial conditions or right-hand-side candidates can be quickly tested by projecting them onto the computed spectrum and running Talbot's method to approximate the solution.

## Data availability

No data was used for the research described in the article.

## Acknowledgements

LG was supported by the Hrvatska Zaklada za Znanost (Croatian Science Foundation) under the grant IP-2019-04-6268 -Randomized low-rank algorithms and applications to parameter dependent problems. CE was supported by the Swedish Research Council under Grant No. 2021-04537.

## References

[1] S. Giani, L. Grubišić, J.S. Ovall, Benchmark results for testing adaptive finite element eigenvalue procedures part 2 (conforming eigenvector and eigenvalue estimates), Appl. Numer. Math. 102 (2016) 1–16.
[2] P. Solin, S. Giani, An iterative adaptive finite element method for elliptic eigenvalue problems, J. Comput. Appl. Math. 236 (18) (2012) 4582–4599.
[3] P. Solin, S. Giani, An iterative adaptive hp-FEM method for non-symmetric elliptic eigenvalue problems, Computing 95 (2013) 183–213.
[4] S. Giani, L. Grubišić, J.S. Ovall, Benchmark results for testing adaptive finite element eigenvalue procedures, Appl. Numer. Math. 62 (2) (2012) 121–140.
[5] C. Engström, S. Giani, L. Grubišić, A spectral projection based method for the numerical solution of wave equations with memory, Appl. Math. Lett. 127 (2022) 107844.
[6] R.G. Durán, C. Padra, R. Rodríguez, A posteriori error estimates for the finite element approximation of eigenvalue problems, Math. Models Methods Appl. Sci. 13 (8) (2003) 1219–1229.
[7] V. Heuveline, R. Rannacher, A posteriori error control for finite element approximations of elliptic eigenvalue problems, J. Adv. Comp. Math. 15 (2001) 107–138.
[8] M.G. Larson, A posteriori and a priori error analysis for finite element approximations of self-adjoint elliptic eigenvalue problems, SIAM J. Numer. Anal. 38 (2000) 608–625.
[9] T.F. Walsh, G.M. Reese, U.L. Hetmaniuk, Explicit a posteriori error estimates for eigenvalue analysis of heterogeneous elastic structures, Comput. Methods Appl. Mech. Engrg. 196 (37) (2007) 3614–3623.
[10] R. Dautray, J.-L. Lions, Mathematical Analysis and Numerical Methods for Science and Technology. Vol. 1, Springer-Verlag, Berlin, 1990, p. xviii+695, Physical origins and classical methods, With the collaboration of Philippe Bénilan, Michel Cessenat, André Gervat, Alain Kavenoky and Hélène Lanchon, Translated from the French by Ian N. Sneddon, With a preface by Jean Teillac.
[11] M.E. Gurtin, A.C. Pipkin, A general theory of heat conduction with finite wave speeds, Arch. Ration. Mech. Anal. 31 (2) (1968) 113–126.
[12] W. Mclean, V. Thomée, Numerical solution via Laplace transforms of a fractional order evolution equation, J. Integral Equations Appl. 22 (1) (2010) 57–94.
[13] S.-L. Wu, Laplace inversion for the solution of an abstract heat equation without the forward transform of the source term, J. Numer. Math. 25 (3) (2017) 185–198.
[14] C. Engström, Spectra of gurtin-pipkin type of integro-differential equations and applications to waves in graded viscoelastic structures, J. Math. Anal. Appl. 499 (2) (2021) 125063, 14.
[15] J.A.C. Weideman, Gauss-Hermite quadrature for the bromwich integral, SIAM J. Numer. Anal. 57 (5) (2019) 2200–2216.
[16] N. Guglielmi, M. López-Fernández, G. Nino, Numerical inverse Laplace transform for convection-diffusion equations, Math. Comp. 89 (323) (2020) 1161–1191.
[17] A. Talbot, The accurate numerical inversion of Laplace transforms, J. Inst. Math. Appl. 23 (1) (1979) 97–120.
[18] J. Abate, W. Whitt, A unified framework for numerically inverting Laplace transforms, INFORMS J. Comput. 18 (4) (2006) 408–421.
[19] D. Arnold, F. Brezzi, B. Cockburn, L. Marini, Unified analysis of discontinuous Galerkin methods for elliptic problems, SIAM J. Numer. Anal. 39 (5) (2002) 1749–1779.
[20] S. Giani, E.J.C. Hall, An a posteriori error estimator for hp-adaptive discontinuous galerkin methods for elliptic eigenvalue problems, Math. Models Methods Appl. Sci. 22 (10) (2012) 1250030.
[21] G. Acosta, T. Apel, R.G. Durán, A.L. Lombardi, Anisotropic error estimates for an interpolant defined via moments, Computing 82 (1) (2008) 1–9.
[22] P.R. Amestoy, I.S. Duff, J.-Y. L'Excellent, Multifrontal parallel distributed symmetric and unsymmetric solvers, Comput. Methods in Appl. Mech. Eng. 184 (2000) 501–520.
[23] D.H. Bailey, A thread-safe arbitrary precision computation package, 2022, https://www.davidhbailey.com/dhbpapers/mpfun2015.pdf. (Online; Accessed 28 September2022).

[24] P. Solin, K. Segeth, I. Dolezel, Higher-Order Finite Element Methods, Chapman and Hall/CRC, 2003.
[25] S. Sauter, hp-finite elements for elliptic eigenvalue problems: Error estimates which are explicit with respect to λ, h, and p, SIAM J. Numer. Anal. 48 (1) (2010) 95–108.
[26] S. Giani, Reliable anisotropic-adaptive discontinuous Galerkin method for simplified P N approximations of radiative transfer, 337 (2018) 225–243.
[27] S. Giani, An a posteriori error estimator for hp-adaptive continuous Galerkin methods for photonic crystal applications, 95 (5) (2013) 395–414.