2024, Vol. 2, pp. 41–55
doi: 10.30970/ana.2024.2.41

Modified heuristic criterion for parameter choice for one stabilization scheme of the finite element method

R. Drebotiy, H. Shynkarenko

*Ivan Franko National University of Lviv,
Universytetska Str. 1, 79000, Lviv, Ukraine*

Анотація. Розглядається модифікація евристичного критерію оптимального вибору параметра стабілізації в певній скінченно-елементній схемі для сингулярно збуреної задачі дифузії-адвекції-реакції. Цей критерій базується на задачі мінімізації функції параметра стабілізації та відповідному алгоритмі знаходження мінімуму. Функція зазвичай демонструє дві точки екстремуму: локальний мінімум і локальний максимум. Це змушує алгоритм неодноразово використовувати додаткову контрольовану евристичну процедуру «спробуй і перевір», яка в деяких випадках може бути неефективною. У цій статті ми пропонуємо прості евристичні модифікації критерію для усунення другої максимальної точки, роблячи звичайний метод розділення навпіл застосовним до процедури пошуку мінімуму.

Abstract. We consider a modification to a heuristic criterion for the optimal choice of the stabilization parameter in a certain finite element scheme for the singularly perturbed diffusion-advection-reaction problem. This criterion is based on a minimization problem of the function of the stabilization parameter and the corresponding algorithm for finding the minimum. The function generally exhibits two extremum points: a local minimum and a local maximum. This forces the algorithm to repeatedly use an additional controlled heuristic "try-and-check" procedure, which can be inefficient in some cases. In this article, we propose simple heuristic modifications to the criterion to eliminate the second maximum point, making ordinary bisection method applicable to the minimum-finding procedure.

1 Introduction

In this article, we consider the Dirichlet problem for the diffusion-advection-reaction (DAR) equation with dominant advection. Problems of this kind are called *singularly perturbed.* Classical finite element schemes [2, 14] using coarse uniform meshes do not provide meaningful results for such problems [13–15]. The main reason is the presence of artificial parasitic oscillations in the approximate finite element solutions obtained. Various finite element algorithms have been developed to address this issue, including adaptive schemes [5, 13, 15], discontinuous finite elements [8], and stabilized schemes [1, 14].

Although adaptive schemes are mostly used for this purpose, we study one stabilization scheme here due to certain benefits. In short, the main idea of stabilization is to add additional penalty terms to the corresponding variational formulation to compensate for advection dominating over other processes, thereby mitigating the parasitic high-frequency oscillations. Naturally, such stabilization procedures may involve some parameters. These parameters determine the balance between the level of applied smoothing of the solution and the overall accuracy.

In [6], we constructed a stabilization scheme for the finite element method based on Tikhonov-type regularization [10]. The scheme involves one positive regularization parameter, which must be chosen appropriately to obtain meaningful results. To select this parameter, we proposed a heuristic criterion in [4], formulated as an optimization problem for a loss function constructed as

2020 *Mathematics Subject Classification*: 65N30.
Key words: finite element method, diffusion-advection-reaction model, stabilization scheme, Cauchy problem, discrete Laplace operator.

a composition of a certain linear functional and the obtained approximation. Since the introduced function exhibits a special structure with several extremum points, we employed a simple heuristic search algorithm to locate the interval where bisection method can then be used to find the needed minimum. However, this procedure may not converge in some cases, resulting in an infinite loop.

It is important to note that the proposed parameter estimation algorithm was designed with the potential for quantum computing applications in finite element methods. Specifically, it allows for the possible use of the Harrow-Hassidim-Lloyd (HHL) linear solver [9] in certain parts of the algorithm, which we will point out later.

In this work, we present several comparable and straightforward ways to modify the mentioned heuristic criterion's loss function, making it suitable for direct application of the bisection algorithm.

The paper is structured as follows: First, we define the model DAR problem. Next, we present and review the algorithm from [6]. Then, we introduce the heuristic algorithm from [4]. Finally, we provide the proposed heuristic modifications and test them on the same sample problem, which was used in the numerical experiment in [4].

2 Diffusion-advection-reaction problem

We consider the following Dirichlet problem for the diffusion-advection-reaction equation

$$\begin{cases} \text{find function } \; u : \bar{\Omega} \to \mathbb{R} \; \text{ such that:} \\ -\mu\Delta u + \vec{\beta} \cdot \nabla u + \sigma u = f \quad \text{in } \; \Omega \subset \mathbb{R}^m, \quad m = 1, 2, \\ u = 0 \quad \text{on } \; \Gamma = \partial\Omega, \end{cases} \tag{2.1}$$

where Ω is a bounded domain with a Lipschitz boundary $\Gamma = \partial\Omega$, $\mu = const > 0$ and $\sigma = const > 0$ are coefficients of diffusion and reaction respectively, function $f = f(x)$ and vector $\vec{\beta} = (\beta_1(x), \beta_2(x))$ represent the sources and advection flow velocity respectively. We will consider incompressible flow, i.e., $\nabla \cdot \vec{\beta} = 0$ in Ω. In case of $m = 1$ we consider $\Omega = (0, 1)$ and the condition of flow incompressibility can be omitted.

The boundary value problem (2.1) admits the following variational formulation

$$\begin{cases} \text{find } \; u \in V := H_0^1(\Omega) \; \text{ such that,} \\ a(u, v) = \langle l, v \rangle \quad \forall v \in V, \end{cases} \tag{2.2}$$

where

$$\begin{cases} a(u, v) = \int\limits_{\Omega} (\mu \nabla u \cdot \nabla v + \vec{\beta} v \cdot \nabla u + \sigma u v) dx \quad \forall u, v \in V, \\ \langle l, v \rangle = \int\limits_{\Omega} f v dx \quad \forall v \in V. \end{cases}$$

Here and below, we assume that the problem data satisfy the conditions of the Lax-Milgram lemma, ensuring that it has a unique weak solution $u \in V$.

Main numerical quantity, that indicates, that the problem may be singularly perturbed is well-known Péclet number

$$Pe := \mu^{-1} \left\| \vec{\beta} \right\|_{\infty} diam \; \Omega.$$

Large values of this number indicate high domination of advection process over diffusion (which is expressed by the term with second-order derivatives). This means that the equation is nearly degenerate to a first-order hyperbolic equation. Such a structure induces boundary layers with high gradients in the solution.

3 Stabilization procedure

Let us recall a stabilization scheme, proposed in [6] for problems with large values of Pe. Let $\Gamma_0 := \{x \in \partial\Omega | \vec{n}(x) \cdot \vec{\beta}(x) < 0\}$, and $\vec{n}$ be an outward unit normal vector to the boundary of domain Ω. Consider the following *reduced problem*

$$\begin{cases} \text{find function} \ \ u_0 \in C^1(\Omega) \ \ \text{such that:} \\ \vec{\beta} \cdot \nabla u_0 + \sigma u_0 = f \quad \text{in} \ \ \Omega, \\ u_0|_{\Gamma_0} = 0, \end{cases} \tag{3.1}$$

and replace the original variational problem (2.2) with the following one

$$\begin{cases} \text{given parameter} \ \ \lambda = const \geq 0, \\ \text{given} \ \ u_0 \ \ \text{is the solution of (3.1)}, \\ \text{find} \ \ u^* \in V \ \ \text{such that:} \\ a(u^*, v) + \lambda(u^*, v)_V = \langle l, v \rangle + \lambda(u_0, v)_V \quad \forall v \in V. \end{cases} \tag{3.2}$$

To numerically solve the (3.2), we use the finite element method with linear elements. In 2D space, we employ a quasi-uniform mesh with triangular elements. In the 1D case, we use a uniform mesh. Additionally, in both cases, we definitely use coarse meshes.

To use the equation from the (3.2) we need to solve two additional problems. First of all, we need to find u_0 from (3.1). The second problem is to choose the value of the parameter λ. In the following chapters, we discuss both.

We should note that the provided stabilization procedure is similar in some ways to Tikhonov regularization. The purpose of the introduced procedure is to overcome the oscillatory nature of the finite element solution of the singularly perturbed problem obtained on coarse meshes. That is, we use stabilization/regularization to make the approximate solution precise enough and suitable for researchers without requiring any mesh refinement. Additionally, since we use a coarse mesh and the singularly perturbed problem will have a boundary layer, we will not be able to fit our approximation to that boundary layer structure. This is a drawback of the provided scheme compared to adaptive schemes, but when we consider, for example, air pollution migration problems modeled by the DAR equation, we may be primarily interested in the general solution structure, and having it imprecise only in the thin boundary layer zone is acceptable.

4 Solving auxiliary Cauchy problem

The problem (3.1) is actually Cauchy problem restricted to domain Ω. We suppose that there are no closed integral curves of field $\vec{\beta}$ which are entirely contained within the domain Ω. If there is a closed integral curve contained within the domain, it can be shown that, in general, a problem of this kind may not be well-posed.

To numerically solve (3.1), it is proposed in [6] to use method of characteristics to decompose the problem to a system of ordinary differential equations (ODEs) and then use some methods like Euler or Runge-Kutta for solving obtained system with some interpolation to calculate the solution value between the obtained curves. Algorithm is the following. Consider $\vec{\beta} \neq 0$ in Ω. Let us define parametrization of curve Γ_0 $[0, mes\Gamma_0] \ni \eta \mapsto \rho(\eta) = (\rho_1(\eta), \rho_2(\eta)) \in \bar{\Gamma}_0$ which maps parameter η bijectively onto the $\bar{\Gamma}_0$. For each value of η we can find integral curve of vector field $\vec{\beta}$: $x(t, \eta) = (x_1(t, \eta), x_2(t, \eta)) \in \Omega$ as a solution of the following Cauchy problem

$$\begin{cases} x_t'(t, \eta) = \vec{\beta}(x(t, \eta)), \\ x(0, \eta) = \rho(\eta). \end{cases} \tag{4.1}$$

Let us define the function $z(t,\eta) = u_0(x(t,\eta))$. Taking into account (4.1), we can derive the following $z'_t(t,\eta) = \vec{\beta}(x(t,\eta)) \cdot \nabla_x u_0(x(t,\eta))$. Using the last equality and (4.1), we can rewrite problem (3.1) as a Cauchy problem for the following system parametrized by η

$$\begin{cases} x'_t(t,\eta) = \vec{\beta}(x(t,\eta)), \\ z'_t(t,\eta) = f(x(t,\eta)) - \sigma z(t,\eta), \\ x(0,\eta) = \rho(\eta), \\ z(0,\eta) = 0. \end{cases} \tag{4.2}$$

We have just presented a well-known method of characteristics [7].

Now, we can consider the discrete set of points $\{\eta_1, ..., \eta_k\} \in [0, mes\Gamma_0]$ and apply the Runge-Kutta method to solve (4.2) for each of the values η_i. In that way we obtain the set of curves, which are lying on the surface $u_0(x)$ defined by (3.1). The last step would be to introduce some interpolation technique to interpolate found curves onto entire domain to obtain approximation to u_0.

5 Overview of the stabilization parameter choice algorithm

The aforementioned stabilization procedure relies on the parameter λ. Choosing this parameter is critical for providing an adequate approximation for advection-dominated problems. If the parameter is set too small, it will lead to an approximate solution with high-frequency oscillations, making it impractical. Conversely, if the parameter is set too large, it will result in a very smoothed solution that loses some of the structure of the exact solution.

That is, we need to choose the parameter as a point between the mentioned cases. For this purpose, in [4], we propose a certain heuristic optimization problem in which the minimum point of the defined loss function is the exact value of the parameter λ.

We consider the following optimization problem

$$\begin{cases} \text{given } \lambda_{\max} > 0, \\ \text{find } \ \lambda = \lambda^* \ \text{ such that} \\ F(\lambda) \to \text{local}\min \quad \text{on} \quad (0, \lambda_{\max}), \end{cases} \tag{5.1}$$

where

$$\lambda_{max} := \frac{\left\|\vec{\beta}\right\|_\infty diam\ \Omega}{Pe_{user}} \tag{5.2}$$

and Pe_{user} is a user-defined value of some "ordinary" Pe value, i.e. for which finite element method on coarse uniform mesh should work fine. For example we can use $Pe_{user} = 10$. Loss function $F(\lambda)$ is defined as follows.

Consider some triangulation of the domain Ω and the ordering of triangulation nodes: x_1, x_2, ..., x_N. For node i and function u let us define discrete Laplace operator approximation to $(\Delta u)(x_i)$ as [3]

$$\tilde{\Delta}(u,i) := \frac{1}{2A_i} \sum_{j \in N_{mesh}(i)} (\cot\alpha_{ij} + \cot\beta_{ij})(u(x_j) - u(x_i)), \tag{5.3}$$

where A_i is 1/3 of sum of areas of adjacent triangles, $N_{mesh}(i)$ is a set of indexes of all adjacent nodes for node i, α_{ij} and β_{ij} are angles opposite to edge (i,j). Defined discrete operator is an approximation to the continuous Laplace operator in selected point.

Let us denote by $u^*_h(x;\lambda)$ approximation $u^*_h(x)$ computed for certain λ. Let us denote by $\bar{N}_{mesh}(i) := N_{mesh}(i) \cup \{i\}$. Let us denote by $M \subset \bar{\Omega}$ the set of all nodes (points) of the mesh and by $\Gamma_{out} := \partial\Omega \setminus \Gamma_0$. If S is some set of node indexes, then we define $\bar{N}_{mesh}(S) := \cup_{j \in S} \bar{N}_{mesh}(j)$. If we have $R \subset \bar{\Omega}$, we can extend definition of $\bar{N}_{mesh}$ to the form $\bar{N}_{mesh}(R) := x[\cup_{j:x_j \in R} \bar{N}_{mesh}(j)]$, where $x[S] := \{x_j | j \in S\}$.

Let us define set of nodes $Q := M \setminus (\Gamma_0 \cup \bar{N}_{mesh}(\Gamma_{out}))$, i.e. Q will contain all internal nodes, excluding those nodes, which are adjacent to any nodes from Γ_{out}. Note, that Γ_{out} in the case of large Pe values will be the part of boundary, adjacent to the boundary layer with high gradient.

Let us define now loss function $F(\lambda)$

$$F(\lambda) := \frac{1}{\|u_h^*(\cdot\,;\lambda)\|_{vect}} \left| \sum_{j \in Q} sign(\tilde{\Delta}(u_h^*(\cdot\,;0), j)) \tilde{\Delta}(u_h^*(\cdot\,;\lambda), j) \right|, \tag{5.4}$$

where by $u_h^*(\cdot\,;\lambda)$ we denoted u_h^* as a function of first independent argument and fixed second argument. By $\|u_h^*\|_{vect}$ we denote the Euclidean norm of the vector of coefficients of the finite element approximation $u_h^*(\cdot\,;\lambda)$.

The general structure of the function $F(\lambda)$ and a minimum point is depicted on the Fig. 5.1.

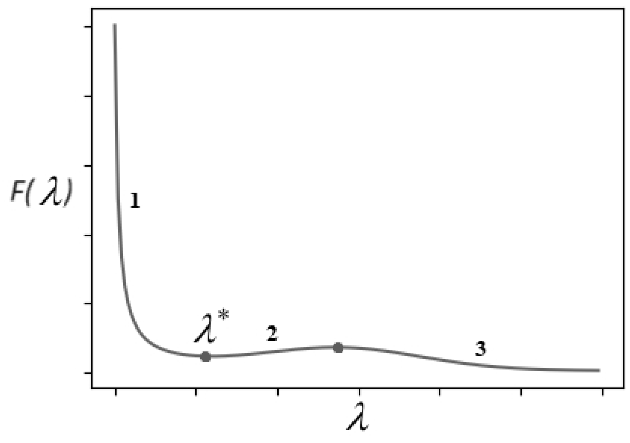

Fig. 5.1. General structure of the loss function [4]

Note, that for 1D case in the sum in expression (5.4) we use divided differences as an approximation to the Laplace operator.

6 Motivation and heuristic optimization

In this section, we briefly review the idea behind the proposed optimization strategy [4].

The main component of this optimization problem is the loss function (5.4). Let us analyze its structure. It is, in fact, a weighted sum of the values of the discrete Laplace operator, calculated at the triangular mesh nodes, excluding the zone with the boundary layer.

The common normalizing multiplier $1/\|u_h^*(\cdot\,;\lambda)\|_{vect}$ is necessary for the possible computation of this loss function on a quantum computer, which we will discuss later.

Let us recall that for singularly perturbed problems with large Pe number, oscillations typically spread over the entire domain, making the finite element approximation on a coarse mesh completely unusable. As λ increases, we will observe a decrease in uniform oscillations. In Fig. 6.1, we show the typical evolution of the solution with increasing values of λ.

Since the Laplace operator measures the curvature of the function graph in a certain sense, the set of all weights $\{sign(\tilde{\Delta}(u_h^*(\cdot\,;0), j))\}$ represents the actual oscillation pattern of the original (non-stabilized) finite element solution with $\lambda = 0$. We use this pattern as a reference. On the other hand, the values $\{\tilde{\Delta}(u_h^*(\cdot\,;\lambda), j)\}$ represent the actual oscillation pattern (and its amplitude) of the stabilized solution for a given λ.

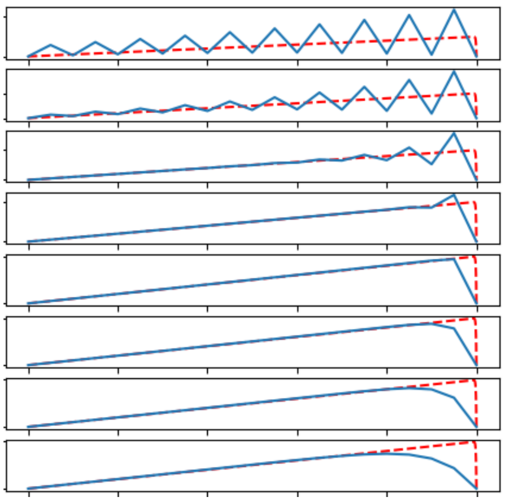
Fig. 6.1. Finite element approximations of 1D problem on unit interval with $\mu = \sigma = 1$, $\beta = f = 10^3$ for different λ: 0, 1, 5, 15, 25, 35, 50, 80 (top to down). Dashed line represents exact solution

Thus, the sum in the expression for the loss function is simply an inner product between the vectors of the reference oscillation pattern (with a high rate of oscillation) and the oscillation pattern for the stabilized solution. This inner product measures the "correlation" between the oscillation pattern that we want to avoid and the actual oscillation.

Taking this fact into account, we can predict the general structure of the function, which should have a short, rapidly descending part corresponding to the decrease in uniform oscillation amplitude as λ increases, followed by a long, slowly descending part after a certain point when high-frequency oscillations have already been eliminated.

Note that the graph in Fig. 5.1 also contains a zone depicted as "2". Adding this relatively short but slowly increasing zone is achieved by excluding the nodes from the sum in the loss function that are adjacent to the finite elements containing the boundary layer. This approach is precisely analyzed in [4].

Also, note that such an ascending zone is necessary to locate the required point (minimum), which, as experiments show, corresponds to the optimal λ value. This is the value for which we have already eliminated parasitic oscillations, while the solution structure corresponds to the exact solution over the entire domain, except for the thin zone consisting only of finite elements containing the boundary layer.

7 Possibility of computing the loss function on a quantum computer

An important note should be considered regarding the computation of $F(\lambda)$. The discrete Laplace operator (5.3) is constructed from the values of the finite element approximation at the mesh nodes. If we expand each term $\tilde{\Delta}(u_h^*(\cdot\,;\lambda), j)$ with that operator in the expression (5.4) and regroup the sum, we will obtain the normalized inner product of a constant vector (with respect to the mesh) and the vector of nodal values of the approximation $u_h^*(\cdot\,;\lambda)$.

Now, taking into account that the vector of nodal values of the approximation is obtained from the linear system of algebraic equations, we see that the loss function $F(\lambda)$ is actually proportional to the projection of the solution of the linear system onto a fixed predefined direction.

Let us recall the problem of solving the linear system on the quantum computer, specifically the well-known Harrow-Hassidim-Lloyd algorithm (HHL) [9]. Suppose that we have built a quantum scheme that can represent our matrix in a suitable form for the mentioned algorithm. After applying HHL, we will obtain a vector proportional to the solution of the linear system encoded as a quantum state.

From the nature of quantum computations, we know that it is not possible to practically reconstruct all individual components (amplitudes) of the obtained vector (quantum state), but it is possible to compute some integral quantity – specifically, the inner product with a predefined vector. For that purpose, we can use the SWAP test [11]. This is exactly what we need, as the structure of $F(\lambda)$ defines the same inner product.

8 Heuristic method for optimization problem

In short, to solve (5.1) in [4], we proposed using two phases. The first phase is to find the upper bound a of the interval $(0, a)$, which lies somewhere below the zone "2" of the curve in Fig. 5.1. The second phase involves running bisections to find the minimum on $(0, a)$.

For the first phase, we employ a simple heuristic geometrically stepped approach to move the point sequentially to the left until we find the ascending part "2" of the loss function.

To be more precise, we will next cite the chapter from [4] that describes the simple heuristic algorithm we used. This algorithm employs certain heuristic rules because the structure of $F(\lambda)$ does not allow us to localize the zone near the minimum, in which there is a unique extremum point. The algorithm consists of two steps

Algorithm 1 Solving optimization problem

```
1: Find point $\lambda_{loc} \in (0, \lambda_{\max}]$ such that $F'(\lambda_{loc}) > 0$;
2: Solve the optimization problem (5.1) on interval $(0, \lambda_{loc})$.
```

Step 1 tries to localize zone around local minimum point, in which this minimum is the only extremum point. This is needed, since from Fig. 5.1 we see that local maximum between zones 2 and 3. For this step we use two heuristic procedures. The step 1 can be detailed as follows

Algorithm 2 Solving optimization problem (**Algorithm** 1 – **Step 1**)

```
1: $\lambda_{loc}$ :=ProcSimple($\lambda_{\max}$)
2: if $\lambda_{loc} > 0$ then
3:     return $\lambda_{loc}$
4: else
5:     return ProcRefined($\lambda_{\max}$)
6: end if
```

In "simple" procedure we use bisections with selecting always left subintervals to try to find ascending part of $F(\lambda)$.

Algorithm 3 Procedure **ProcSimple**($\lambda_{\max}$)

```
Require: $\delta > 0$, $MaxIter \in \mathbb{N}$;
1: initialization: $\lambda_{loc} := \lambda_{\max}$;
2: while $F(\lambda_{loc} - \delta) \geq F(\lambda_{loc} + \delta)$ and $MaxIter > 0$ do
3:     $\lambda_{loc} := \lambda_{loc}/2$;
4:     $MaxIter := MaxIter - 1$;
5: end while
6: if $MaxIter \leq 0$ then
7:     return 0
8: end if
9: return $\lambda_{loc}$
```

Refined procedure has the same steps, but it extends step 3 to make additional "backsteps" to the right from current parameter value.

Algorithm 4 Procedure **ProcRefined**($\lambda_{\max}$)

```
Require: $\delta > 0$, $MaxIter \in \mathbb{N}$;
1: initialization: $\lambda_{loc} := \lambda_{\max}$;
2: while $F(\lambda_{loc} - \delta) \geq F(\lambda_{loc} + \delta)$ and $MaxIter > 0$ do
3:   $\lambda_{back}$ :=ProcRefinedBacksteps($\lambda_{loc}$)
4:   if $\lambda_{back} > 0$ then
5:     return $\lambda_{back}$
6:   end if
7:   $\lambda_{loc} := \lambda_{loc}/2$;
8:   $MaxIter := MaxIter - 1$;
9: end while
10: if $MaxIter \leq 0$ then
11:   return 0
12: end if
13: return $\lambda_{loc}$
```

Algorithm 5 Procedure **ProcRefinedBacksteps**(λ_{loc})

```
Require: $\delta > 0$, $MaxIter \in \mathbb{N}$;
1: initialization: $k := 1$;
2: initialization: $\lambda_{back} := \lambda_{loc}/2$;
3: while $F(\lambda_{back} - \delta) \geq F(\lambda_{back} + \delta)$ and $k < MaxIter$ do
4:   $\lambda_{back} := \lambda_{loc}\left(1 - \frac{1}{2}\left(\frac{2}{3}\right)^k\right)$;
5:   $k := k + 1$;
6: end while
7: if $k \geq MaxIter$ then
8:   return 0
9: end if
10: return $\lambda_{back}$
```

Note, that formula $\lambda_{back} := \lambda_{loc}\left(1 - \frac{1}{2}\left(\frac{2}{3}\right)^k\right)$ used in **ProcRefinedBacksteps** represents bisection, with choosing always right subinterval for next step with midpoint always shifted to the left.

For Algorithm 1 step 2 we use standard bisections to find minimum with given precision.

9 Modifications to the loss function

We can see from Fig. 5.1 that to apply a simple bisection algorithm to find the minimum, we can somehow eliminate the part "3" from the $F(\lambda)$ graph. Experiments show that the part "2" can be quite small, so capturing it through the stepped point movement (which resembles a "brute force" search) described in Algorithm 2 can, in some cases, skip that "2" region, thus leading us to an infinite loop (which we will technically stop by maximum iteration condition, resulting in stopping the search). Additionally, when the ascending part of the graph is short and lies near the left side, we have a long tail behaving like a convex function that converges to a horizontal asymptote. It is not hard to see from the stabilization procedure that the function needs to have that asymptote.

To change the shape of the loss function, we propose using simple graph shifting. Note that such a modification should not significantly change the minimum point of the function.

We can shift the graph in different ways.

Algorithm 6 Solving optimization problem (**Algorithm** 1 – **Step 2**)

```
Require: $\delta > 0$, $Tol > 0$, $MaxIter \in \mathbb{N}$;
 1: initialization: $a := 0$;
 2: initialization: $b := \lambda_{loc}$;
 3: while $b - a \geq Tol$ do
 4:     $\lambda_{mid} := \frac{a+b}{2}$
 5:     if $F(\lambda_{mid} - \delta) < F(\lambda_{mid} + \delta)$ then
 6:         $b := \lambda_{mid}$
 7:     else
 8:         $a := \lambda_{mid}$
 9:     end if
10: end while
11: return $\lambda_{mid}$
```

9.1 Linear shifting

Let us introduce three points $\lambda_i = \lambda_{max}(i+1)/4$ for $i = 1, 2, 3$. Consider positive user-defined constant "shifting factor" K which is relatively large. For example, we can use $K = 10$.

We define now linearly shifted loss function in the following manner

$$H_l(\lambda) := F(\lambda) + K\lambda \max\{|F'(\lambda_i)|\}_{i=1}^3.$$

In H_l we just add linear function based on some slope coefficient, which we want to set much greater than average slope coefficient on the tail part of the graph of $F(\lambda)$, to progressively compensate and overcome the negative slope of the $F(\lambda)$.

To calculate $F'(\lambda_i)$ we can just use finite difference approximation.

Note, that if we detect that any of the values $F'(\lambda_i)$, $i = 1, 2, 3$ is greater than zero, we can reassign λ_{max} to that value and start bisections algorithm for the corresponding interval.

9.2 Quadratic shifting

Another way of shifting is the following. As in previous case, if we detect that any of the values $F'(\lambda_i)$, $i = 1, 2, 3$ is greater than zero, we can reassign λ_{max} to that value and start bisections for the corresponding interval. Otherwise, let us define values $y_i := F(\lambda_i)$, where λ_i are the same as for linear shifting. K is also the same user-defined constant. Let us define

$$K_q := \frac{16(2y_2 - y_1 - y_3)}{2\lambda_{max}^2},$$

$$C_q := 2y_1 - y_3 - K_q\lambda_{max}^2/2.$$

If $K_q \geq 0$ we fallback to usage of the linear shifting, otherwise we introduce auxiliary quadratic function

$$Q(\lambda) := C_q - y_1 - \frac{1}{\lambda_{max}}(y_3 - y_1)(2\lambda - \lambda_{max}) - \frac{K_q}{2}(\lambda_{max} - \lambda)(2\lambda - \lambda_{max})$$

and define new loss function

$$H_q(\lambda) := F(\lambda) + KQ(\lambda).$$

If we have a function $F(\lambda)$ with a relatively larger maximum (see Fig. 5.1) between zones "2" and "3", then the curvature of the function in zone "3" will also be larger. Applying linear shifting requires a larger shifting factor K, which can lead to greater changes in the minimum point of λ to the left. To overcome this effect, we first build a quadratic interpolation function $g(\lambda)$ that fits the $F(\lambda)$ curve exactly at the points $(\lambda_i, y_i)_{i=1}^{3}$ to approximately "capture" the curvature of $F(\lambda)$ in its tail zone "3". Then we consider the function $Q(\lambda) = K(g(0) - g(\lambda))$, which is defined as above. It is, in fact, a vertically reflected function of $g(\lambda)$ that passes through the origin. This vertical reflection is intended to compensate for the curvature of the descending part "3" of the $F(\lambda)$ graph.

9.3 Rotation-projection shifting

The last approach is to apply a clockwise rotation to the coordinate system and then express the function graph in terms of that rotated system. This rotation should compensate for the descending part "3" of the graph of $F(\lambda)$.

Consider a user-defined small positive angle γ. Let s consider the line $y = -k\lambda$, where $k := \tan\gamma$ and y is the coordinate in which we show the values of $F(\lambda)$. We will treat this line as the new, rotated λ axis. We can calculate the projection of the point $(\lambda, F(\lambda))$ onto that line as follows

$$\begin{aligned} \lambda_{proj}(\lambda) &= \frac{\lambda - kF(\lambda)}{1+k^2}, \\ F_{proj}(\lambda) &= -k\lambda_{proj}(\lambda). \end{aligned}$$

Now we define new loss function in the following way

$$H_p(\lambda) := \|(\lambda, F(\lambda)) - (\lambda_{proj}(\lambda), F_{proj}(\lambda))\|,$$

where we used Euclidean norm $\|(y_1, y_2)\| := \sqrt{\sum_{i=1}^{2} y_i^2}$ in the second term to calculate the distance between the point and the rotated λ axis.

10 Numerical experiments

We consider two experiments: one for a 2D problem and another for a 1D problem. In both cases, the main purpose of the experiment is to observe how the optimal λ point changes when different types of shifting are applied with various values of the parameters. For each case, we provide the obtained optimal λ. In both numerical examples, we used Runge-Kutta method for the solution of obtained system of ODEs.

In the 1D experiment, we additionally provide graphs of the approximate solutions along with the corresponding exact solution graph, as the 1D case is more illustrative. We also demonstrate the influence of λ in the 1D example.

Let us consider as example 2D problem data from [4]

$$\begin{aligned} &\Omega = (0,1)^2, \mu = 1, \vec{\beta}(x,y) \equiv (10^3, 10^3), \sigma = 10^2, \\ &f(x,y) = 10^5 \cos(4.5\pi x/2)\cos(4.5\pi y/2). \end{aligned} \tag{10.1}$$

Table 10.1. Optimal parameter calculation results for the problem (10.1)

shifting type	K=10	K=100	$\gamma = 10°$	$\gamma = 20°$
linear	20.117	**20.117**	-	-
quadratic	20.117	**20.117**	-	-
projection	-	-	**20.117**	**20.117**

We use $Pe_{user} = 5$, making $\lambda_{max} = 400$. The tolerance for finding optimal λ is $\lambda_{max}/10^3$. Mesh is quasi-uniform with maximum triangle area 0.005. Without shifting we found $\lambda_{optimal} = 20.117$.

With different types of shifting and with different values of parameters we obtained the data placed in Table 10.1. The code is written in Python.

For solving linear systems we used standard function numpy.linalg.solve from NumPy lib, which under the hood calls LAPACK routine "gesv". This routine uses LU factorization to solve the linear system.

Obviously, due to tolerance selection as $\lambda_{max}/10^3$, for all cases mentioned in Table 10.1 the number of bisections is equal to 10.

By bolding, we indicate the cases where, after shifting, only the bisection algorithm was used without additional steps to find the interval. As we can see, the minimum point does not change with the application of different shifting strategies. This can be explained as follows. Despite being singularly perturbed, this problem does not have a particularly "hard" singular perturbation. Therefore, the optimal λ is located close to the 0 point, making it less sensitive to those shifts, which are progressive in the increasing λ. We also observe that the algorithm's behavior may depend on the configurable constants.

Consider another example – similar 1D problem but with larger Pe number

$$\Omega = (0,1), \mu = 1, \beta(x) = 10^4, \sigma = 10^2, f(x) = 10^6 \cos(4.5\pi x). \tag{10.2}$$

Computed $\lambda_{max} = 10^3$. We used uniform mesh with 20 elements. Obtained optimal $\lambda_{optimal} = 232.910$. The same experiment data is shown in Table 10.2.

Table 10.2. Optimal parameter calculation results for problem (10.2)

shifting type	K=10	K=100	$\gamma = 10°$	$\gamma = 20°$
linear	**232.91**	**145.99**	-	-
quadratic	**232.91**	**133.30**	-	-
projection	-	-	**228.02**	**184.08**

As in 2D example, we choose the tolerance for finding lambda in the same way, as $\lambda_{max}/10^3$, making bisections count equal to 10.

For solving linear system we used Thomas algorithm, since obtained matrices are tridiagonal.

In all cases where shifting was applied, we obtained a function that has one extremum point, meaning that only the bisection method was used to approximate that minimum point. We observe that for larger values of Pe, the obtained λ values are more sensitive to the shifting. For linear and quadratic shifting with $K = 100$, as well as for projection shifting with a larger angle, we see a decrease in the obtained parameter.

In Fig. 10.1, we present the finite element solution graphs for Table 10.2.

As we see on Fig. 10.1, the graphs of obtained finite element approximations $u_h^*(\cdot\,;\lambda_{optimal})$ are slightly different only around the last element near the boundary layer. Other part of the solution still keep the same shape. Due to our recent ongoing investigation, we can combine h-adaptivity with described stabilization, making the overall algorithm much less sensitive on the parameter value, since we will neglect that by using adaptivity.

In general, introduced shifting strategies are comparable and can be used with some automatic code, which will try different methods and use them appropriately on demand (as many popular systems like Wolfram Mathematica do).

It is not clear what is the best way to choose parameters of the shifting. At least, we can see from the experiments that there is way less ambiguity in selection of those parameters, rather then manually selecting λ. For now we leave the selection of those parameter to the researcher.

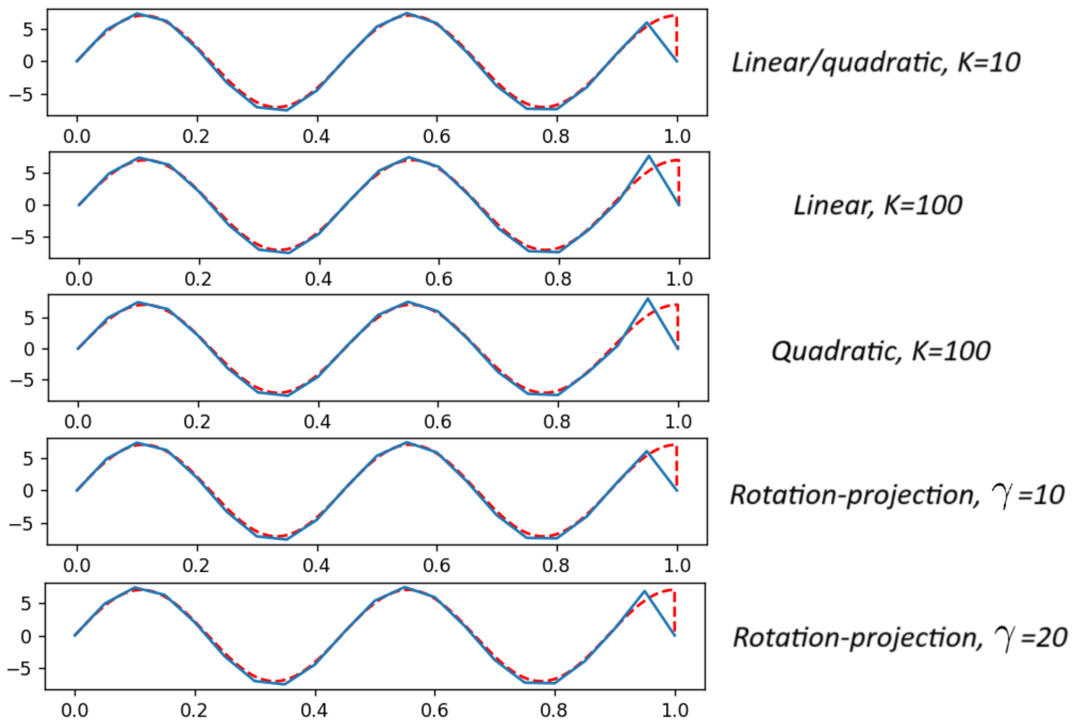

Fig. 10.1. Finite element approximations of 1D problem with data (10.2) with different type of shifting used (corresponding to Table 10.2). Dashed line represents exact solution

11 Advantages of the method and general remarks

Nowadays, the primary and most commonly used approach to dealing with singularly perturbed problems is mesh adaptation. This approach is widely adopted due to its universality. On the other hand, the development of quantum computing has sparked natural interest in investigating the possibility of combining it with the finite element method. Since the most computationally intensive part of the finite element method is solving linear systems, it is natural to attempt to exploit the currently available algorithms for solving linear systems on a quantum computer. Despite this, the probabilistic nature of quantum computations, particularly wave function collapse, makes it impossible to use the quantum linear solver directly in the same way as it is used in classical FEM schemes. These and other related questions are extensively discussed in the articles [11, 12].

The main point regarding the quantum linear solver is that the solution is encoded in the amplitudes of the quantum state. Thus, we can obtain some integral quantity by measuring, but not the entire vector of all individual amplitudes at once.

To overcome the oscillatory nature of the solution to the singularly perturbed problem, a typical adaptive scheme implements a loop of mesh refinement. In each iteration, we calculate the finite element solution on the current mesh by leveraging a linear solver. Then, we find the distribution of error indicators and refine the mesh based on that. In the next iteration, we start with the solution on the new, refined mesh, and so on. The process finishes when some stopping criterion, based on a posteriori error estimate, is met.

For the researcher, each of the intermediate steps of the adaptive scheme may not be interesting. Despite this, we still need to have the entire set of individual nodal values at each step, since we need to compute per-element error indicators based on the local values of the finite element solutions. Moreover, we need to implement a comparison procedure to select the appropriate subset of elements based on their error indicator values. Therefore, the direct use of the HHL algorithm in adaptive schemes seems impractical.

In the proposed Algorithm 1, similarly to adaptive schemes, we also have a loop, at each step of which we need to compute the value of the loss function $F(\lambda)$ and the approximation $u_h^*(\,\cdot\,;\lambda)$. As discussed previously, we can leverage the HHL algorithm together with the SWAP test to compute the value of $F(\lambda)$. Thus, the value of the loss function is the only quantity based on the finite element solution $u_h^*(\,\cdot\,;\lambda)$ that we wish to extract at each step. Therefore, we definitely do not need to extract the entire vector of nodal values at the intermediate steps. Only in the final step we need to solve the FEM linear system on a classical computer to provide the researcher with the solution. Note that, due to the use of a fixed coarse mesh, the dimension of the linear system in the final step will be the same as in the initial step of the adaptive algorithm, making the proposed algorithm more computationally efficient even in the classic sense.

Thus, in the proposed algorithm, each of the intermediate linear slover steps for obtaining the finite element solution can be implemented on a quantum computer, which is its main advantage compared to the most widely used adaptive schemes.

It is important to note that the implementation of this algorithm on a real quantum computer is not yet practical due to limited resources. The currently available number of qubits in existing hardware is insufficient to handle computations for real problems. Additionally, implementing error-tolerant computations with error-correcting codes presents challenges due to the large depth of overall circuits and the relatively short lifespan of quantum states.

One general note on the asymptotic number of steps that the algorithm will take: due to the expression of the form (5.2), we can see that the overall number of steps will be $O(\log(Pe))$.

12 Drawbacks of the algorithm and notes on error estimates

We encounter three main drawbacks of the algorithm.

The first drawback of the algorithm is that it is heuristic. Despite all the steps being logically derived and making sense, there is no strict proof of the structure of the graph of the loss function.

The second drawback is that by using coarse meshes, we cannot fit the approximation to the thin boundary layer. We would say this is not a drawback, but rather a property of the stabilization scheme, which by definition uses a *fixed* mesh with a *low* element count. Further improvement of the approximation is achieved not through mesh adaptation, but by changing the variational problem with added penalizing terms. This limitation can be mitigated by the limited use of h-adaptivity in combination with the described algorithm, and this is a topic of our ongoing research, with some results currently under review.

The last drawback is the absence of general a priori estimates for the stabilization procedure itself. So far, we have not derived any a priori estimates. Despite this, in the same ongoing research, we have derived some special *a posteriori* error estimates to guide the combined stabilization-adaptation process. In short, those estimates combine the standard technique of obtaining residual-based estimates simultaneously with two variational problems, (2.2) and (3.2), to yield a single vector of per-element error indicators.

13 Conclusions

In this article, we constructed a modified heuristic procedure for finding the optimal value of the regularization parameter for a certain stabilized finite element scheme. We proposed three similar shifting strategies to improve the structure of the loss function and prevent the overall procedure from entering infinite loops.

Numerical results are provided for both 2D and 1D problems.

Declarations

Conflict of Interest.
The authors have no conflict of interest to declare that are relevant to the content of this article.
Funding.
This study was not supported by any sponsor or funder.
Author Contributions.
Authors contributed equally to the research and preparation of this article.

Additional information

H.S. is a member of the Editorial Board for JANA. The paper was handled by another Editor and has undergone a rigorous peer review process. H.S. was not involved in the journal's peer review, or decisions related to this manuscript.

References

1. Bartels, S.: Numerical Approximation of Partial Differential Equations. Springer, 541 (2016)
2. Brenner, S., Scott L.: The Mathematical Theory of Finite Element Methods. Springer, 404 (2008)
3. Crane, K., de Goes, F., Desbrun, M., Schröder, P.: Digital geometry processing with discrete exterior calculus. ACM SIGGRAPH. Courses. SIGGRAPH '13. **7**, 1-126 (2013). doi:10.1145/2504435.2504442.
4. Drebotiy, R., Shynkarenko, H.: Heuristic choice of the regularization parameter for optimal stabilization of the finite element approximations. Mathematical Methods and Physicomechanical Fields. Pidstryhach Institute for Applied Problems of Mechanics and Mathematics, National Academy of Sciences of Ukraine. Lviv. **66** (1-2), 206-221 (2023)
5. Drebotiy, R., Shynkarenko, H.: On the application of the one hp-adaptive finite element strategy for nonsymmetric convection-diffusion-reaction problems. Journal of Numerical and Applied Mathematics, ISSN: 0868-6912. Kyiv. **3** (126), 48-61 (2017)
6. Drebotiy, R., Shynkarenko, H.: Regularized finite element method for singular perturbed convection-diffusion-reaction models with nonuniform sources. Visnyk of the Lviv University. Series Appl.Math.and Informatics, ISSN: 2078-5097. Lviv. **29**, 27-36 (2021)
7. Fritz, J.: Partial differential equations (4th ed.). Springer, ISBN 978-0-387-90609-6
8. Feng, X., Karakashian, O., Xing, Y.: Recent Developments in Discontinuous Galerkin Finite Element Methods for Partial Differential Equations. The IMA Volumes in Mathematics and its Applications. **157** (2012)
9. Harrow, A. W., Hassidim, A., Lloyd, S.: Quantum algorithm for solving linear systems of equations. Physical Review Letters. **103** (15) 150502. arXiv:0811.3171
10. Lenzen, F., Scherzer, O.: Tikhonov type regularization methods: History and recent progress. Proceeding Eccomas (2004)
11. Montanaro, A., Pallister, S.: Quantum algorithms and the finite element method. ISSN:2469-9934, Physical Review A. **93** (3), American Physical Society (APS) (2016)
12. Scherer, A., Valiron, B., Mau, S.-C., Alexander, S., van den Berg, E., Chapuran, T. E.: Concrete resource analysis of the quantum linear system algorithm used to compute the electromagnetic scattering cross section of a 2D target. Quantum Inf Process. **16** (60) (2017). arXiv:1505.06552v2
13. Shynkarenko, H. A., Kozarevska, Y.: The regularization of numerical solutions of variational problems of impurities migration: an adaptive finite element method. Part 1. Visnyk of the Lviv University. Series Appl.Math.and Informatics. ISSN: 2078-5097. Lviv. **5**, 153-165 (2002)

14. Trushevsky, V. M., Shynkarenko, H. A., Shcherbyna, N. M.: Finite element method and artificial neural networks: theoretical aspects and application. Lviv, Ivan Franko National University of Lviv. ISBN 978-617-10-0127-5 (2014) (in Urkainian)

15. Verfürth, R.: Adaptive finite element methods. Ruhr-Universität Bochum: Lecture notes. 129 (2011)

Received 03.08.2024
Revised 05.10.2024