





RESEARCH PAPER

A new algorithm using L-BFGS for two-parameter eigenvalue problems from Lamé's system and simulation

Hayati Ünsal Özer ^{1,2,†} and Ahmet Duran ^{1,*,†}

¹Istanbul Technical University, Department of Mathematics Engineering, 34469 Istanbul, Türkiye,

²Yıldız Technical University, Department of Mathematics Engineering, 34220 Istanbul, Türkiye

*Corresponding Author

†ozzerh@itu.edu.tr (Hayati Ünsal Özer); aduran@itu.edu.tr (Ahmet Duran)

Abstract

We deal with the challenges and solutions for two-parameter eigenvalue problems (TPEPs) involving large-scale various dense coefficient matrices using several numerical methods. We propose a new method, via fused parameter optimization (*fusedparopt*) algorithm using limited-memory Broyden-Fletcher-Goldfarb-Shanno (L-BFGS) having several variations, for TPEPs. We combine the advantages of the tensor Rayleigh quotient (RQ), Newton (N) and L-BFGS methods, while avoiding the disadvantages of each method. They are designed for certain TPEPs having real eigenvalue tuples. We test the performance of our algorithm and compare them with state-of-art algorithms such as *twopareigs* from MultiParEig toolbox in Matlab, tensor Rayleigh quotient-Newton (RQ_N) and L-BFGS alone, by using the coefficient matrices coming from Lamé system and simulations via randomly generated matrices. We also obtain convergence diagrams for the *fusedparopt_LBFGS* to understand the convergence behavior for the number of iterations and computational times via Monte Carlo simulation. We observe that our algorithms can reduce computation time, diminish divergence problems, and give more stable solutions for our data set including various matrices for TPEPs. To the best of our knowledge, we perform the first study including *fusedparopt_LBFGS* method in this type of eigenvalue problem.

Keywords: Simulation; Lamé system; numerical linear algebra and matrix theory; algorithm for two-parameter eigenvalue problem; numerical parameter optimization

AMS 2020 Classification: 65F15; 65L10; 65C05; 65F05; 90C31

1 Introduction

In this study, we deal with two-parameter eigenvalue problems (TPEPs) and we propose a new method via a parameter optimization algorithm for solving TPEPs in large-scale systems. We validate the proposed method on different testing problems.

Let the structure of the TPEP be defined as below

$$A_1 \mathbf{x} = \lambda B_1 \mathbf{x} + \mu C_1 \mathbf{x}, \quad (1)$$

$$A_2 \mathbf{y} = \lambda B_2 \mathbf{y} + \mu C_2 \mathbf{y}, \quad (2)$$

where $\lambda, \mu \in \mathbb{R}$ are eigenvalues of the problem and $\mathbf{x} \in \mathbb{R}^{n_1} \setminus \{\mathbf{0}\}$, $\mathbf{y} \in \mathbb{R}^{n_2} \setminus \{\mathbf{0}\}$ are corresponding eigenvectors. Also, $A_1, B_1, C_1 \in \mathbb{R}^{n_1 \times n_1}$, $A_2, B_2, C_2 \in \mathbb{R}^{n_2 \times n_2}$ are coefficient matrices. For more details and fundamental theorems about TPEPs we refer to the works [1–3].

Deterministic grid multi-start approach and random multi-start approach are proposed in order to find all or selected number of different eigenvalue tuples for three-parameter eigenvalue problems via our fused algorithm having steepest descent (SD) technique in [4] where we apply the approaches for the coefficient matrices coming from ellipsoidal wave equation. Unlike [4], one of the contributions of this paper is that we propose *fusedparopt_LBFGS* with a focus for Lamé's system. Lamé's system enables applications in engineering and applied physics and it is used for elasticity theory [5], such as in a plane orthotropic medium [6]. The system is obtained by applying separation of variables to the Helmholtz equation in sphero-conal coordinates [7, 8]. Moreover, we compare our method with the known algorithm RQ_N on a real application using Lamé's system in an electrostatic field for charge-singularity problem at the corner of a flat plate [9]. Also, we contrast these methods with a state-of-art method of *twopareigs* from *MultiParEig* toolbox [10] in Matlab. Morrison and Lewis [9], Plestenjak et al [7], Bailey [11] and Ji [12] consider the numerical solution of the charge-singularity problem for a relatively smaller size (on 60 points) than our simulations (2000 points).

TPEPs appear in numerous science and engineering problems. For example, they emerge from discretized boundary value problems solved by the separation of variables method in mathematical physics [3]. In addition, the TPEPs have been used for the stability of delay-differential equations [13]. They are also employed in power system models such as the power flow equations having two bus power systems in electrical engineering [14].

It is important to develop numerical solution methods for different kinds of TPEPs because TPEP can be non-singular, singular, right definite, small or large. The corresponding matrices may have various properties. In recent years, various numerical methods have been proposed for solving TPEPs, e.g., continuation method [15], Jacobi-Davidson approach [16, 17], Sylvester-Arnoldi type method [18], transformation into a non-linear problem [19], steepest descent (called gradient search method as well) [20–22], homotopy method for quadratic problems [23], alternating method using affine transformation and Helmholtz equations [24] and studies for singular TPEPs [25, 26]. In addition, [7, 27, 28] are among the studies in numerical solution methods for three-parameter and multi-parameter eigenvalue problems. Each method has advantages and limitations under certain conditions. For example, homotopy methods such as [23, 29] can be considered to compute all eigenvalues of a multi-parameter eigenvalue problem, while subspace methods such as [17] may be preferred to compute some selected eigenvalues. Computing the eigenvalues of a non-symmetric matrix is still a challenging problem. If a matrix A is symmetric or normal, then its eigenvalues are well-conditioned. However, the problem of computing the eigenvalues of a non-symmetric matrix is often ill-conditioned [30]. We also deal with nonsymmetric matrices obtained from real eigenvalue tuples in this paper.

Newton's method as a solver is an important part of our proposed approach. Generally, it is known that Newton's method converges rapidly when the iterations start close enough to a solution. First, we consider the tensor Rayleigh quotient-Newton (RQ_N) algorithm [15] in Figure 1 for the TPEPs such as Eqs. (1) and (2). In brief, the algorithm starts with coefficient matrices $A_1, B_1, C_1, A_2, B_2, C_2$

and initial eigenvectors $\mathbf{x}_0, \mathbf{y}_0$. Then, the tensor Rayleigh quotient (RQ) [15] computes initial eigenvalues λ_0, μ_0 (from Definition 1) with these coefficient matrices and initial eigenvectors for the starting eigenvalues in Newton's method. Using all these elements, Newton's method tries to approximate the eigenvalues of the problem until the test condition is satisfied. When the test condition is valid, RQ_N algorithm stops and gives approximate eigenvalues. Also, more details on RQ_N algorithm can be found in [15]. RQ_N can be useful for the moderate (or small) size of TPEPs [15, 17]. However, RQ_N method occasionally has limitations for large-scale problems. In our application on Lamé's system and simulations (Section 4), we encounter convergence problems and high-cost solutions in large-scale problems via RQ_N. The main reason for this situation is that it becomes difficult to determine sufficiently accurate initial conditions when the size of the problem grows in Newton's method. Although the RQ is a good alternative for determining initial eigenvalue approximations, we observe that it is not always sufficient for large-scale problems in our simulation. Therefore, this situation emerges the requirement to generate an effective solution for large-scale problems.

Parameter optimization algorithms have various important applications such as physics, engineering, mathematical finance and economics [31–33]. Convergence diagrams via Monte Carlo simulation are important tools to understand the convergence behavior of iterative methods in such applications [32–34]. Optimization studies for certain large-scale eigenvalue problems are also existent [35]. They study the minimization or maximization of the j^{th} largest eigenvalue of an analytic and Hermitian matrix-valued function and describe subspace procedures [35].

In the literature, we see very interesting optimization studies using a kind of limited memory BFGS (L-BFGS) method in large-scale problems arising in machine learning applications from speech recognition, sensor networks, network traffic and internet search [36].

L-BFGS is an alternative method to the SD method. It is an updated BFGS method for large-scale unconstrained optimization problems, by Nocedal [37]. Besides large-scale problems, it is also used for parameter estimation [38]. Moreover, it is proposed to use in combination with Cholesky factorization to develop an algorithm for generalized inverse eigenvalue problems [39]. This method is remarkable for its low storage and cost in computations [40]. Therefore, we first consider the L-BFGS method alone in the solution of TPEPs. We find that L-BFGS alone does not converge and is not effective in TPEPs for our data set. Consequently, we also propose L-BFGS method in our fused algorithm, as an alternative to the *fusedparopt_SD* method in [4]. It is called *fusedparopt_LBFGS* method.

Outline. The remainder of the paper is organized as follows: In the following subsections, we present the preliminary including the related methods and theorems for this paper. In Section 2, our new method *fusedparopt_LBFGS* via fused parameter optimization algorithm for TPEPs is presented. In Section 3, we provide the related computational complexity of the proposed algorithm. In Section 4, we illustrate the numerical experiments for performance comparison of the related methods. First, we consider L-BFGS method alone in the solution of TPEPs. We compare *fusedparopt_LBFGS* method versus *fusedparopt_SD* method for the solution of TPEPs. Then, we make an application on the trigonometric form of Lamé's system using *fusedparopt_LBFGS*, RQ_N, *fusedparopt_SD* and state-of-art twopareigs methods. Finally, we present the convergence diagrams for *fusedparopt_LBFGS*, RQ_N and *fusedparopt_SD* methods. Section 5 concludes the paper.

Tensor Rayleigh quotient - Newton algorithm

The flowchart of RQ_N algorithm [15] is shown in Figure 1.

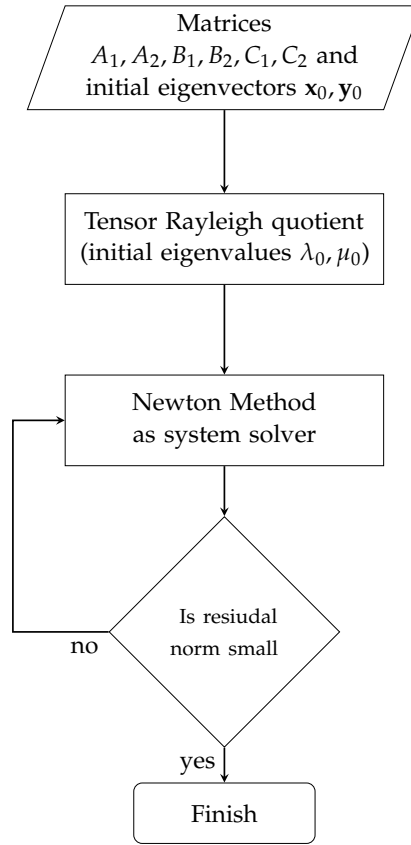


Figure 1. RQ_N algorithm

Definition 1 (from [15]) The RQ is 2-tuple (λ_0, μ_0) such that

$$\lambda_0 = \frac{(\mathbf{x}^T C_1 \mathbf{x})(\mathbf{y}^T A_2 \mathbf{y}) - (\mathbf{x}^T A_1 \mathbf{x})(\mathbf{y}^T C_2 \mathbf{y})}{(\mathbf{x}^T B_1 \mathbf{x})(\mathbf{y}^T C_2 \mathbf{y}) - (\mathbf{x}^T C_1 \mathbf{x})(\mathbf{y}^T B_2 \mathbf{y})},$$

$$\mu_0 = \frac{(\mathbf{x}^T B_1 \mathbf{x})(\mathbf{y}^T A_2 \mathbf{y}) - (\mathbf{x}^T A_1 \mathbf{x})(\mathbf{y}^T B_2 \mathbf{y})}{(\mathbf{x}^T B_1 \mathbf{x})(\mathbf{y}^T C_2 \mathbf{y}) - (\mathbf{x}^T C_1 \mathbf{x})(\mathbf{y}^T B_2 \mathbf{y})}.$$

Transformation for TPEPs

In order to implement L-BFGS or the SD technique on TPEP, the Eqs. (1) and (2) are transformed to the system of equations below.

$$\mathbf{F}(\mathbf{v}) = \begin{bmatrix} A_1 \mathbf{x} - \lambda B_1 \mathbf{x} - \mu C_1 \mathbf{x} \\ \frac{1}{2}(1 - \mathbf{x}^T \mathbf{x}) \\ A_2 \mathbf{y} - \lambda B_2 \mathbf{y} - \mu C_2 \mathbf{y} \\ \frac{1}{2}(1 - \mathbf{y}^T \mathbf{y}) \end{bmatrix} = \begin{bmatrix} f_1(\mathbf{v}) \\ f_2(\mathbf{v}) \\ f_3(\mathbf{v}) \\ f_4(\mathbf{v}) \end{bmatrix} = \mathbf{0}.$$

The system of equations has a solution at $\mathbf{v} = [\mathbf{x}, \lambda, \mathbf{y}, \mu]^T \in \mathbb{R}^{n_1+n_2+2}$ when the function g defined

by

$$\begin{aligned} g(\mathbf{v}) &= \sum_{i=1}^4 [f_i(\mathbf{v})]^2 \\ &= (A_1\mathbf{x} - \lambda B_1\mathbf{x} - \mu C_1\mathbf{x})^2 + \left(\frac{1}{2}(1 - \mathbf{x}^T\mathbf{x})\right)^2 \\ &\quad + (A_2\mathbf{y} - \lambda B_2\mathbf{y} - \mu C_2\mathbf{y})^2 + \left(\frac{1}{2}(1 - \mathbf{y}^T\mathbf{y})\right)^2, \end{aligned}$$

has the minimal value 0.

Let $\mathbf{v} = [\mathbf{x}, \lambda, \mathbf{y}, \mu]^T \in \mathbb{R}^{n_1+n_2+2}$ be a fixed point and let $\mathbf{v} + \Delta\mathbf{v} = [\mathbf{x} + \Delta\mathbf{x}, \lambda + \Delta\lambda, \mathbf{y} + \Delta\mathbf{y}, \mu + \Delta\mu]^T \in \mathbb{R}^{n_1+n_2+2}$ be a second point. Then the function $w = g(\mathbf{v})$ changes by an amount Δw in going from \mathbf{v} to $\Delta\mathbf{v}$:

$$\begin{aligned} \Delta w &= g(\mathbf{v} + \Delta\mathbf{v}) - g(\mathbf{v}) \\ &= g(\mathbf{x} + \Delta\mathbf{x}, \lambda + \Delta\lambda, \mathbf{y} + \Delta\mathbf{y}, \mu + \Delta\mu) - g(\mathbf{x}, \lambda, \mathbf{y}, \mu). \end{aligned}$$

Then Δw can be written in the form:

$$\Delta w = \Delta\mathbf{x}(a + \varepsilon_1) + \Delta\lambda(b + \varepsilon_2) + \Delta\mathbf{y}(c + \varepsilon_3) + \Delta\mu(d + \varepsilon_4),$$

where a, b, c, d are independent of $\Delta\mathbf{x}, \Delta\lambda, \Delta\mathbf{y}, \Delta\mu$ and $\varepsilon_1, \varepsilon_2, \varepsilon_3, \varepsilon_4$ are functions of $\Delta\mathbf{x}, \Delta\lambda, \Delta\mathbf{y}, \Delta\mu$ such that

$$\lim_{\substack{\Delta\mathbf{x} \rightarrow 0, \Delta\lambda \rightarrow 0, \\ \Delta\mathbf{y} \rightarrow 0, \Delta\mu \rightarrow 0}} \varepsilon_1 = 0, \quad \lim_{\substack{\Delta\mathbf{x} \rightarrow 0, \Delta\lambda \rightarrow 0, \\ \Delta\mathbf{y} \rightarrow 0, \Delta\mu \rightarrow 0}} \varepsilon_2 = 0, \quad \lim_{\substack{\Delta\mathbf{x} \rightarrow 0, \Delta\lambda \rightarrow 0, \\ \Delta\mathbf{y} \rightarrow 0, \Delta\mu \rightarrow 0}} \varepsilon_3 = 0, \quad \lim_{\substack{\Delta\mathbf{x} \rightarrow 0, \Delta\lambda \rightarrow 0, \\ \Delta\mathbf{y} \rightarrow 0, \Delta\mu \rightarrow 0}} \varepsilon_4 = 0.$$

The linear function of $\Delta\mathbf{x}, \Delta\lambda, \Delta\mathbf{y}$ and $\Delta\mu$ is then termed the total differential of w at the point \mathbf{v} and is denoted by dw :

$$dw = a\Delta\mathbf{x} + b\Delta\lambda + c\Delta\mathbf{y} + d\Delta\mu.$$

If $\Delta\mathbf{x}, \Delta\lambda, \Delta\mathbf{y}$ and $\Delta\mu$ are sufficiently small, dw gives a close approximation to Δw . Therefore, the function $w = g(\mathbf{v})$ is said to be differentiable at the point \mathbf{v} .

When $g(\mathbf{v})$ is differentiable, the direction of greatest decrease in the value of g at \mathbf{v} is the direction given by $-\nabla g(\mathbf{v})$ where

$$\nabla g(\mathbf{v}) = \left(\frac{\partial g}{\partial \mathbf{x}}(\mathbf{v}), \frac{\partial g}{\partial \lambda}(\mathbf{v}), \frac{\partial g}{\partial \mathbf{y}}(\mathbf{v}), \frac{\partial g}{\partial \mu}(\mathbf{v}) \right)^T.$$

L-BFGS algorithm for TPEPs

L-BFGS method (as one of the quasi-Newton (QN) methods) is a useful method for solving large-scale systems. Although this method offers modest storage requirements, the convergence rate of this method is linear. More details and the algorithm can be found in [40]. Below, we just give the L-BFGS algorithm that we use in Section 4 for TPEPs.

$$\mathbf{v}_{k+1} = \mathbf{v}_k - \alpha_k H_k \nabla g_k,$$

where α_k is the step size and $-H_k \nabla g_k$ is the search direction. In a QN method, a positive definite matrix H_k is used to approximate the inverse of Hessian matrix $\nabla^2 g_k^{-1}$, instead of computing exact second derivatives. H_k is updated at every iteration by the formula

$$H_{k+1} = (I - \rho_k y_k s_k^T) H_k (I - \rho_k y_k s_k^T) + \rho_k s_k s_k^T,$$

where

$$\rho_k = \frac{1}{y_k^T s_k}, \quad s_k = \mathbf{v}_{k+1} - \mathbf{v}_k, \quad y_k = \nabla g_{k+1} - \nabla g_k.$$

Algorithm 1 L-BFGS algorithm for TPEPs

Input: Initial matrices: $A_i, B_i, C_i, i = 1, 2$

Initial vector: $\mathbf{v} = [x_0, \lambda, y_0, \mu]^T$

Integer m value: $m > 0$

Output: Computed vector: $\mathbf{v} = [x_k, \lambda, y_k, \mu]^T$

for $k = 0, 1, 2, 3, \dots$ **do**

Choose H_k^0 (for example; identity matrix or by using equation (7.20) from [40])

$q \leftarrow \nabla g_k$

for $i = k - 1, k - 2, \dots, k - m$ **do**

$\alpha_i \leftarrow \rho_i s_i^T q$

$q \leftarrow q - \alpha_i y_i$

end for

$r \leftarrow H_k^0 q$

for $i = k - m, k - m + 1, \dots, k - 1$ **do**

$\beta \leftarrow \rho_i y_i^T r$

$r \leftarrow r + s_i (\alpha_i - \beta)$

end for

$p_k = -r \quad (H_k \nabla g_k = r)$

Compute step size α_k (for example, by Backtracking line search method) then

$\mathbf{v}_{k+1} \leftarrow \mathbf{v}_k + \alpha_k p_k$

if $k > m$ **then**

Remove the vector pair $\{s_{k-m}, y_{k-m}\}$ from storage

end if

Compute and save: $s_k \leftarrow \mathbf{v}_{k+1} - \mathbf{v}_k$ and $y_k \leftarrow \nabla g_{k+1} - \nabla g_k$

end for (Or continue iteration until test condition is satisfied)

The steepest descent technique

The following definition and theorems are from the references [41] and [42].

Definition 2 (from [41]) Suppose that $g : \mathbb{R}^n \rightarrow \mathbb{R}$ be continuously differentiable on \mathbb{R}^n and $\mathbf{x}^{(0)} \in \mathbb{R}^n$. Let t_k is the value of $t \geq 0$ that be the minimizer of the function

$$\varphi_k(t) = g(\mathbf{x}^{(k)} - t \nabla g(\mathbf{x}^{(k)})), \quad t \geq 0.$$

Then, given an initial point $\mathbf{x}^{(0)}$, the SD sequence $\mathbf{x}^{(k)}$ defined by the formula

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - t_k \nabla g(\mathbf{x}^{(k)}).$$

Theorem 1 Let $\mathbf{x}^{(k)}$ be the SD sequence for $g(x)$. If $\nabla g(\mathbf{x}^{(k)}) \neq \mathbf{0}$ for some k , then $g(\mathbf{x}^{(k+1)}) < g(\mathbf{x}^{(k)})$.

Proof It can be found in [41].

Theorem 2 Suppose that $g : \mathbb{R}^n \rightarrow \mathbb{R}$ be a coercive function with continuous first partial derivatives on \mathbb{R}^n . Then, given any initial point $\mathbf{x}^{(0)}$ in \mathbb{R}^n , the SD sequence $\mathbf{x}^{(k)}$ contains a subsequence that converges to a critical point of $g(\mathbf{x})$.

Proof It can be found in [41].

Theorem 3 Let $g : \mathbb{R}^n \rightarrow \mathbb{R}$ be a strictly convex, coercive function with continuous first partial derivatives on \mathbb{R}^n . Then, given any initial point $\mathbf{x}^{(0)}$ in \mathbb{R}^n , the SD sequence $\mathbf{x}^{(k)}$ with initial guess $\mathbf{x}^{(0)}$ converges to the unique global minimizer of $g(\mathbf{x})$.

Proof It can be found in [41].

Theorem 4 Suppose that $g : \mathbb{R}^n \rightarrow \mathbb{R}$ is continuously differentiable on the set $S = \{\mathbf{x} \in \mathbb{R}^n \mid g(\mathbf{x}) \leq g(\mathbf{x}^{(0)})\}$ and that S is closed and bounded set. Then every point $\bar{\mathbf{x}}$ that is a cluster point of the sequence $\{\mathbf{x}^{(k)}\}$ satisfies $\nabla g(\bar{\mathbf{x}}) = \mathbf{0}$.

Proof It can be found in [42].

Also, the following feature

$$\lim_{k \rightarrow \infty} \|\nabla g_k\| = 0,$$

is satisfied [40]. By this aspect of the technique, the problems faced by Newton's method in TPEPs including large-scale matrices can be diminished. To see this aspect of the technique, we benefit from [Theorem 1 - Theorem 4](#).

The purpose is to reduce $g(\mathbf{v})$ to its minimal value of zero, so a convenient choice for $\mathbf{v}^{(k+1)}$ is to move away from $\mathbf{v}^{(k)}$ in the direction that gives the greatest decrease in the value of $g(\mathbf{v})$. Hence we let

$$\mathbf{v}^{(k+1)} = \mathbf{v}^{(k)} - \alpha \nabla \mathbf{z}, \quad \text{where } \mathbf{z} = \frac{\nabla g(\mathbf{v}^{(k)})}{\|\nabla g(\mathbf{v}^{(k)})\|_2}. \quad (3)$$

Here, the scalar $\alpha \geq 0$ is called step size or step length. An appropriate value of α should be chosen so that $g(\mathbf{v}^{(k+1)})$ will be significantly less than $g(\mathbf{v}^{(k)})$. Therefore, three possible methods are referred below to determine α .

Quadratic polynomial interpolation. One of the alternative options to assign step size alpha is the quadratic polynomial interpolation(qpi) method [43]. So, we could consider the single-variable function as

$$h(\alpha) = g(\mathbf{v}^{(0)} - \alpha \mathbf{z}),$$

where the value of α minimizes h .

In order to find a minimal value for h , the quadratic polynomial $P(\alpha)$ that interpolates h at α_1, α_2 and α_3 is constructed. Here, $\alpha_1 < \alpha_2 < \alpha_3$ should be chosen that is close to where the minimum value of $h(\alpha)$ occurs. So, α is defined in $[\alpha_1, \alpha_3]$.

We now find the quadratic polynomial $P(\alpha)$ which has a minimum in $[\alpha_1, \alpha_3]$ to approximate the minimal value of $h(\alpha)$. For this reason, we use Newton's forward divided-difference to find quadratic polynomial $P(\alpha)$ that interpolates the data $(\alpha_1, g_1), (\alpha_2, g_2)$ and (α_3, g_3) . The quadratic polynomial $P(\alpha)$ has the form

$$P(\alpha) = g_1 + h_1(\alpha - \alpha_1) + h_3(\alpha - \alpha_1)(\alpha - \alpha_2),$$

where $h_1 = \frac{g_2 - g_1}{\alpha_2 - \alpha_1}, h_2 = \frac{g_3 - g_2}{\alpha_3 - \alpha_2}$ and $h_3 = \frac{h_2 - h_1}{\alpha_3 - \alpha_1}$.

In our algorithm, first we define $\alpha_1 = 0$ and calculate $g_1 = g(\mathbf{v}^{(0)} - \alpha_1 \mathbf{z}) = g(\mathbf{v}^{(0)})$. Also, we let $\alpha_3 = 1$ and calculate $g_3 = g(\mathbf{v}^{(0)} - \alpha_3 \mathbf{z})$. Finally, we check for condition $g_3 < g_1$. If the condition does not provide, it should be updated α_3 with $\alpha_3 = \alpha_3/2$ until the condition is valid. When the condition is provided, we set $\alpha_2 = \alpha_3/2$ and calculate $g_2 = g(\mathbf{v}^{(0)} - \alpha_2 \mathbf{z})$.

The minimum value of $P(\alpha)$ on $[\alpha_1, \alpha_3]$ occurs at the critical point of $P(\alpha)$. So, we have $P'(\alpha) = 0$ and from this obtain $\alpha = 0.5(\alpha_2 - h_1/h_3)$. If $\hat{g} = g(\mathbf{v}^{(0)} - \alpha \mathbf{z})$ is smaller than g_1 and g_3 , we assign

$$\mathbf{v}^{(1)} = \mathbf{v}^{(0)} - \alpha \mathbf{z}.$$

Backtracking line search. Backtracking line search (bls) is a kind of inexact line search method that works by choosing a step size α which approximately minimizes a given function [44]. This method reduce g along $\{\mathbf{v} - \alpha \nabla g(\mathbf{v}) \mid \alpha \geq 0\}$ given a descent direction $\nabla g(\mathbf{v})$. It is based on two constants ρ, τ with $0 < \rho < 0.5, 0 < \tau < 1$. We start with choosing constants $\rho = 0.01$ and $\tau = 0.1$. The method initializes with step size $\alpha = 1$ and then reduces it by a factor τ until stopping condition is satisfied

$$g(\mathbf{v} - \alpha \nabla g(\mathbf{v})) \leq g(\mathbf{v}) - \rho \alpha \|\nabla g(\mathbf{v})\|_2^2.$$

Therefore, we obtain an appropriate α value for the SD method. Finally, we assign

$$\mathbf{v}^{(1)} = \mathbf{v}^{(0)} - \alpha \nabla g(\mathbf{v}^0).$$

Algorithm 2 Backtracking line search

```

 $\alpha = 1$ 
while  $g(\mathbf{v} - \alpha \nabla g(\mathbf{v})) > g(\mathbf{v}) - \rho \alpha \|\nabla g(\mathbf{v})\|_2^2$  do
     $\alpha = \tau \alpha$ 
end while

```

Two-point step size. Two-point step size (tpss) was proposed by Barzilai and Borwein [45] for the SD method. This method works on the iteration $\mathbf{v}^{(k+1)} = \mathbf{v}^{(k)} - \alpha^{(k)} \mathbf{h}^{(k)}$ where $\mathbf{h}^{(k)} = \mathbf{h}(\mathbf{v}^{(k)}) =$

$\nabla g(\mathbf{v}^{(k)})$ and $g : \mathbb{R}^{n_1+n_2+2} \rightarrow \mathbb{R}$. Therefore, formula of $\alpha^{(k)}$ is given as below

$$\alpha^{(k)} = \frac{\langle \Delta \mathbf{v}, \Delta \mathbf{h} \rangle}{\langle \Delta \mathbf{h}, \Delta \mathbf{h} \rangle},$$

and $\langle \mathbf{a}, \mathbf{b} \rangle$ denotes the scalar product of the vectors \mathbf{a} and \mathbf{b} . Here, $\alpha^{(k)}$ minimizes $\|\Delta \mathbf{v} - \alpha \Delta \mathbf{h}\|^2$, with $\Delta \mathbf{v} = \mathbf{v}^{(k)} - \mathbf{v}^{(k-1)}$ and $\Delta \mathbf{h} = \mathbf{h}^{(k)} - \mathbf{h}^{(k-1)} = \nabla g(\mathbf{v}^{(k)}) - \nabla g(\mathbf{v}^{(k-1)})$. Then we obtain the iteration in the following form

$$\mathbf{v}^{(k+1)} = \mathbf{v}^{(k)} - \alpha^{(k)} \nabla g(\mathbf{v}^{(k)}).$$

Convergence of the steepest descent technique for TPEPs

We propose the following theorem about the convergence of the SD technique for TPEPs.

Theorem 5 *Suppose that $A_1, B_1, C_1 \in \mathbb{R}^{n_1 \times n_1}$, $A_2, B_2, C_2 \in \mathbb{R}^{n_2 \times n_2}$ are coefficient matrices. Suppose the matrices of size $n_1 n_2 \times n_1 n_2$ obtained by the related tensor products T_i where $i = 0, 1, 2$. Assume that tensor product $T_0 = B_1 \otimes C_2 - C_1 \otimes B_2$ is nonsingular and diagonally dominant. Let S be the set of feasible solutions for TPEP. Let g be the gradient sequence generated by the SD technique. Suppose that $g : \mathbb{R}^{n_1+n_2+2} \rightarrow \mathbb{R}$ is continuously differentiable on the set $S = \{\mathbf{v} \in \mathbb{R}^{n_1+n_2+2} \mid g(\mathbf{v}) \leq g(\mathbf{v}^{(0)})\}$. Then every point $\bar{\mathbf{v}}$ that is a cluster point of the sequence $\{\mathbf{v}^{(k)}\}$ satisfies $\nabla g(\bar{\mathbf{v}}) = \mathbf{0}$.*

Proof The TPEP can be reduced to a system of 2 one-parameter problems as in [2]. Since the matrix of size $n_1 n_2 \times n_1 n_2$ obtained by the related tensor product T_0 is nonsingular and diagonally dominant, S is closed and bounded set obtained by using the Gershgorin circle theorem [46] with closed circles and the related bounds for spectral radii coming from [47]. By using the [Theorem 4](#), every point $\bar{\mathbf{v}}$ that is a cluster point of the sequence $\{\mathbf{v}^{(k)}\}$ satisfies $\nabla g(\bar{\mathbf{v}}) = \mathbf{0}$.

In [Section 4](#), we have an application on Lamé's system which gives the tensor product T_0 nonsingular and diagonally dominant. Also, we use randomly generated matrices in other examples. So, T_0 coming from them is highly likely nonsingular.

2 Proposed method via fusedparopt algorithm using L-BFGS for TPEPs

In order to improve RQ_N algorithm in [Figure 1](#), we have used the SD technique because of being globally convergent in our paper [4] before. Now, we propose L-BFGS method for an alternative method to SD in this paper. We can rearrange starting approximations or eigenpairs (during iteration) by using L-BFGS technique. Therefore, we can use L-BFGS technique in three ways as seen in [Figure 2](#). So, we call the fused algorithm as *fusedparopt_LBFGS* algorithm.

The first way is using L-BFGS technique between RQ and Newton's methods (using just P_1 in [Figure 2](#)). This means that it can be used before Newton's method to improve the initial eigenvectors generated randomly and the initial eigenvalues generated by the RQ. Thus, the Newton method can start to run in better conditions than before. In the second way, the L-BFGS technique is mounted inside Newton's method to upgrade the eigenpairs when needed (using just P_2 in [Figure 2](#)). For instance, if we encounter conditions such as divergence or stagnation, we should upgrade the eigenpairs. The third way occurs in combination with the first and the second ways (using P_1 and P_2 together in [Figure 2](#)).

When we consider *fusedparopt_LBFGS* algorithm and the step size method such as bls together, we could show our fused algorithm in three ways as in [Figure 2](#). These ways are;

- fused algorithm with the first way P_1 and the bls method is $RQ + LBFGS_{bls} + Newton$

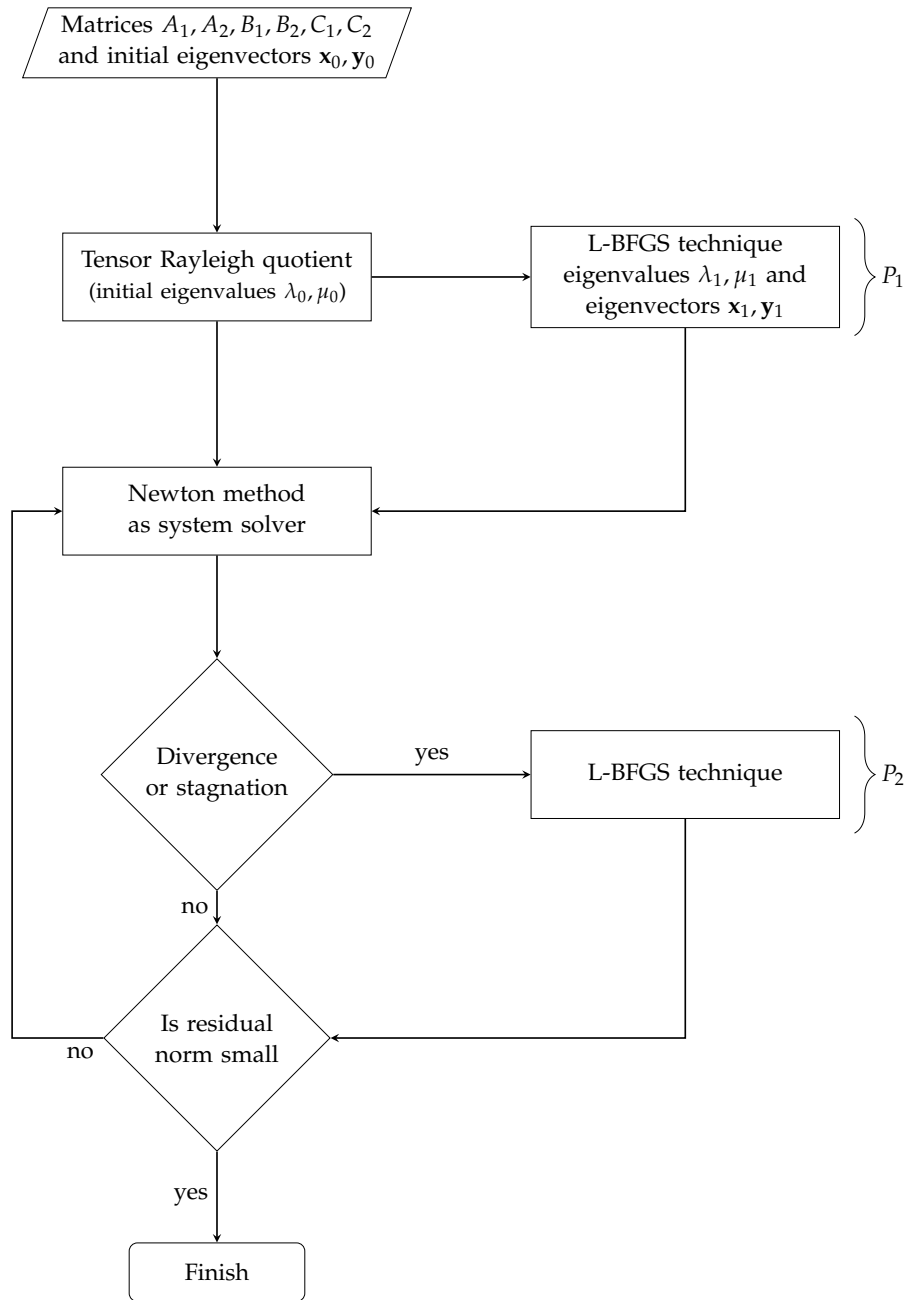


Figure 2. Proposed *fusedparopt_LBFGS* algorithm

- fused algorithm with the second way P_2 and the bls method is $RQ + Newton[LBFGS_{bls}]$
- fused algorithm with the third way (using P_1 and P_2 together) and the bls method is $RQ + LBFGS_{bls} + Newton[LBFGS_{bls}]$

Algorithm 3 Proposed fused RQ_N_I (I: Improver) algorithm with option SD or option L-BFGS for TPEP. There are two options for RQ_N_I including *fusedparopt_SD* using steepest descent and *fusedparopt_LBFGS* using L-BFGS.

Input: Initial matrices: $A_i, B_i, C_i, i = 1, 2$
Initial eigenvectors: x_0, y_0

Output: Computed eigenvalues: λ, μ
Final eigenvectors: x, y

- Calculate the tensor Rayleigh quotient

$$(\lambda_0, \mu_0) = rayleighQuotient(x_0, y_0, A_1, B_1, C_1, A_2, B_2, C_2)$$

- **Improve initial eigenpairs by**

- **Option-1: the steepest descent technique**

$$(\lambda_1, \mu_1, x_1, y_1) = \text{steepestDescent}(\lambda_0, \mu_0, x_0, y_0, A_1, B_1, C_1, A_2, B_2, C_2)$$

- **Option-2: the L-BFGS technique**

$$(\lambda_1, \mu_1, x_1, y_1) = \text{LBFGS}(\lambda_0, \mu_0, x_0, y_0, A_1, B_1, C_1, A_2, B_2, C_2)$$

- **Approximate the eigenvalues by the Newton's method**

Use the values $(\lambda_1, \mu_1, x_1, y_1, A_1, B_1, C_1, A_2, B_2, C_2)$

for $k = 1, 2, 3, \dots$ **do**

- Calculate the vectors $p_{ik}, q_{ik}, i = 1, 2$

$$p_{1k} \leftarrow v_1^{-1} B_1 x_k, p_{2k} \leftarrow v_1^{-1} C_1 x_k \text{ where } v_1 = A_1 - \lambda_k B_1 - \mu_k C_1$$

$$q_{1k} \leftarrow v_2^{-1} B_2 x_k, q_{2k} \leftarrow v_2^{-1} C_2 x_k \text{ where } v_2 = A_2 - \lambda_k B_2 - \mu_k C_2$$

- Generate the following linear system by the vectors above and solve it

$$\begin{bmatrix} x_k^T p_{1k} & x_k^T p_{2k} \\ y_k^T q_{1k} & y_k^T q_{2k} \end{bmatrix} \begin{bmatrix} \Delta \lambda_k \\ \Delta \mu_k \end{bmatrix} = \begin{bmatrix} \frac{1}{2}(x_k^T x_k + 1) \\ \frac{1}{2}(y_k^T y_k + 1) \end{bmatrix}$$

- Update the eigenvectors x_{k+1}, y_{k+1}

$$x_{k+1} \leftarrow \Delta \lambda_k p_{1k} + \Delta \mu_k p_{2k}$$

$$y_{k+1} \leftarrow \Delta \lambda_k q_{1k} + \Delta \mu_k q_{2k}$$

- Update the eigenvalues λ_{k+1}, μ_{k+1}

$$\lambda_{k+1} \leftarrow \lambda_k + \Delta \lambda_k$$

$$\mu_{k+1} \leftarrow \mu_k + \Delta \mu_k$$

- Improve eigenpairs for once when the following condition is available

if divergence or stagnation occurs after a certain iteration **then**

- **Option-1: the steepest descent technique**

$$(\lambda_{k+1}, \mu_{k+1}, x_{k+1}, y_{k+1}) = \text{steepestDescent}(\lambda_{k+1}, \mu_{k+1}, x_{k+1}, y_{k+1}, A_i, B_i, C_i, i = 1, 2)$$

- **Option-2: the L-BFGS technique**

$$(\lambda_{k+1}, \mu_{k+1}, x_{k+1}, y_{k+1}) = \text{LBFGS}(\lambda_{k+1}, \mu_{k+1}, x_{k+1}, y_{k+1}, A_i, B_i, C_i, i = 1, 2)$$

end if

- Continue iteration until the test condition is satisfied

end for

3 Computational complexity for *fusedparopt_LBFGS*

In this section, we analyze the computational costs of *fusedparopt_LBFGS* and the alternatives including *RQ_N* and *fusedparopt_SD*. First, we assume that the coefficient matrices A_i, B_i and C_i ($i = 1, 2$) have $n \times n$ dimensions and are dense. Also, the initial vectors x_0 and y_0 have n dimension. In *RQ_N* algorithm, the RQ requires approximately $O(n^2)$ operations and Newton's method using Gaussian elimination has the cost of approximately $O(n^3)$ operations [48].

The L-BFGS method has $O(n)$ computational complexity alone [40]. There is an additional cost due to L-BFGS technique in *fusedparopt_LBFGS* algorithm when it is compared with *RQ_N* algorithm.

Let us discuss the effect of SD technique in *fusedparopt_SD* algorithm. SD technique generally costs approximately $O(n^2)$ operations [48]. In order to compare the costs of step size methods in SD technique, we need to see costs in a little more detail. We can calculate costs approximately $3n^2$ flops for tpss, $(2+k)n^2$ flops for bls and $(6+l)n^2$ flops for qpi where k and l are given as unknown operation numbers because of searching appropriate step size value in the algorithms. Theoretically, *fusedparopt_SD* algorithm seems more costly than *RQ_N* algorithm. However, we find a different situation in practice according to our experiments. The computational cost can

be saved by reducing the number of iterations for Newton’s method when the SD technique is utilized in the fused *fusedparopt_SD* algorithm. In other words, an^2 units of workload by SD technique can be realized instead of potential bn^3 units of workload of Newton’s method, where a and b are unknown numbers of iterations. Therefore, the SD technique can insert negligible or amortized overhead because the SD technique may eliminate the stagnation of Newton’s method for nearly singular matrices. Thus, the cost of Gaussian elimination is dominant in *RQ_N*, *fusedparopt_LBFGS* and *fusedparopt_SD* algorithms.

4 Numerical experiments

In this section, examples are grouped into two different themes. Firstly, we examine L-BFGS method which is an alternative method for SD. We make simulations to compare SD and L-BFGS located in our fused algorithms. In the second theme, there is an application on a real problem. We implement our fused algorithms on Lamé’s system. We use MATLAB R2022a in the machine having Intel(R) Core(TM) i7-7700 CPU @ 3.60GHz and 16GB RAM memory for testing our algorithms throughout this section.

Comparing L-BFGS with an alternative method SD

In this part, we use LBFGS method with bls algorithm as one of the quasi-Newton methods, instead of SD technique, in our proposed fused algorithm. So, we aim to compare the performance of SD technique against an alternative method in the fused algorithm. We use the iteration of L-BFGS method as shown below instead of SD iteration (3)

$$\mathbf{v}^{(k+1)} = \mathbf{v}^{(k)} - \alpha^{(k)} H^{(k)} \nabla g(\mathbf{v}^{(k)}).$$

Unlike the SD technique, we also use $H^{(k)}$ which is the inverse Hessian approximation here. In this method, $H^{(k)}$ is stored by a certain number of vector pairs to reduce storage cost. For example, the m most recent vector pairs are kept. More details can be found in [40].

L-BFGS method alone in the solution of TPEPs

Example 1 We make a simulation to see the status of L-BFGS method alone in the solution of TPEPs. By generating all matrices and initial vectors randomly with the help of *rand()* function in MATLAB, we create a system of equations in each different size. We solve the systems of equations corresponding to different values of m (such as $m = 3, 17, 21$) to a certain number of iterations ($Iter = 1015$). We give the convergence results by residual norm $(\|A_1\mathbf{x}_i - \lambda_i B_1\mathbf{x}_i - \mu_i C_1\mathbf{x}_i\|^2 + \|A_2\mathbf{y}_i - \lambda_i B_2\mathbf{y}_i - \mu_i C_2\mathbf{y}_i\|^2)^{\frac{1}{2}}$ and the computation time is given in seconds in Table 1. According to the results in Table 1, we observe that L-BFGS alone does not converge and is not effective in TPEPs to a certain number of iterations ($Iter = 1015$).

We have also made tests with m values different from those in Table 1, but we still could not obtain convergence. This may be related to the fact that the eigenvalue spectrum is too dispersed in problems involving randomly generated matrices. It may need more iterations for convergence. In this case, the cost will be very high.

fusedparopt_LBFGS method versus *fusedparopt_SD* method for the solution of TPEPs

Example 2 We design a new simulation by generating all A_i, B_i, C_i ($i = 1, 2$) matrices and initial vectors randomly with the help of *rand()* function in MATLAB, so that the tensor product T_0 of this system is nonsingular. We create 500 different equation systems having matrices of size 2000×2000 . We solve each system of equations with the methods *RQ_N*, *fusedparopt_SD* and *fusedparopt_LBFGS* respectively. We use

Table 1. Results of the L-BFGS method in different sizes of matrices which are generated fully random (Time is in seconds)

SIZE	L-BFGS (m=3)			L-BFGS (m=17)			L-BFGS (m=21)		
	Iter.	Residual	Time	Iter.	Residual	Time	Iter.	Residual	Time
250	1014	2.04E+02	3.04	1014	2.09E+02	3.22	1014	2.10E+02	3.35
500	1014	4.34E+03	35	1014	4.41E+03	36	1014	4.43E+03	36
1000	1014	1.44E+04	126	1014	1.44E+04	127	1014	1.44E+04	127
2000	1014	4.61E+04	457	1014	4.61E+04	458	1014	4.60E+04	459
4000	1014	1.11E+05	1740	1014	1.11E+05	1754	1014	1.11E+05	1758
8000	1014	4.49E+05	7308	1014	4.49E+05	7432	1014	4.49E+05	7464

the bls algorithm for determining the step lengths in *fusedparopt_SD* and *fusedparopt_LBFGS* methods where $m = 3$ for the L-BFGS method. Therefore, we apply 3 different variations of each *fusedparopt* method in the solutions. The convergences are calculated by residual norm $(\|A_1x_i - \lambda_i B_1x_i - \mu_i C_1x_i\|^2 + \|A_2y_i - \lambda_i B_2y_i - \mu_i C_2y_i\|^2)^{\frac{1}{2}}$ in this simulation. We run our algorithm until the residual norm $< 1e-10$ is satisfied. If not satisfied, we stop the algorithm when the iteration number is 1015.

Table 2. Basic statistical results in different methods for 500 cases. (c.time is in seconds)

	RQ_N		<i>fusedparopt_SD</i>		<i>fusedparopt_LBFGS</i>	
	iter.	c.time	iter.	c.time	iter.	c.time
Mean	177	56	38	14	38	15
Std. Dev.	331	105	17	6	15	5
Max. Val.	1014	348	154	53	113	42
Min. Val.	11	3	13	6	12	6
Median	46	15	32	13	34	14
IQR	45	15	13	4	12	5

Table 2 contains the basic statistical results including mean, standard deviation(Std. Dev.), maximum value(Max. Val.), minimum value(Min. Val.), median and interquartile range(IQR) of the iteration numbers and the computation times for the solutions. According to the **Table 2**, the results obtained via *fusedparopt_SD* and *fusedparopt_LBFGS* methods are better than RQ_N method. Moreover, *fusedparopt_SD* and *fusedparopt_LBFGS* methods have values close to each other.

Table 3. The number of winning methods and their proportions among 500 cases

(a) The number of winning methods in *fusedparopt_SD*

<i>fusedparopt_SD</i> METHODS	QUANT	PERCENT
RQ + SD _{bls} + Newton	210	42%
RQ + SD _{bls} + Newton[SD _{bls}]	143	28.6%
RQ + Newton[SD _{bls}]	147	29.4%
TOTAL	500	100%

(b) The number of winning methods in *fusedparopt_LBFGS*

<i>fusedparopt_LBFGS</i> METHODS	QUANT	PERCENT
RQ + LBFGS _{bls} + Newton	213	42.6%
RQ + LBFGS _{bls} + Newton[LBFGS _{bls}]	127	25.4%
RQ + Newton[LBFGS _{bls}]	160	32%
TOTAL	500	100%

In Table 3, we see the numbers and proportions of the methods giving the best results in the simulation. Table 3a gives the number of winning methods in *fusedparopt_SD* and Table 3b gives the number of winning methods in *fusedparopt_LBFGS*. In both Tables, we can say that the first methods ($RQ + SD_{bls} + Newton$ and $RQ + LBFGS_{bls} + Newton$) are more effective than the others.

Table 4. Statistics of the iteration numbers for 100 cases in different sizes of matrices used in the simulation

SIZE	RQ_N				<i>fusedparopt_SD</i>				<i>fusedparopt_LBFGS</i>			
	Mean	StdDev	Mdn	IQR	Mean	StdDev	Mdn	IQR	Mean	StdDev	Mdn	IQR
250	162	330	34	34	32	20	26	17	34	16	31	14
500	166	329	37	26	33	16	28	15	34	14	33	13
1000	174	339	38	25	33	15	30	13	36	17	32	13
2000	184	336	47	38	37	16	33	12	37	14	34	11
4000	187	335	49	46	38	13	36	16	40	13	38	13
8000	197	331	58	48	45	15	43	19	50	18	46	18

We show the statistical results of the iteration numbers and the computation times in second that occur in solving systems of equations having different matrix sizes in Table 4 and Table 5, respectively. We generate 100 different equation systems in each size in the simulation and then we solve them. While high-cost solutions are available with the RQ_N method, we can obtain more effective solutions with the *fusedparopt_SD* and *fusedparopt_LBFGS* methods. In addition, it is seen that the results obtained by these two methods are close to each other. However, we can say that the *fusedparopt_SD* method is a little more effective than *fusedparopt_LBFGS* method in terms of the average number of iterations and the average computation times.

Table 5. Statistics of the computation time in second for 100 cases in different sizes of matrices used in the simulation

SIZE	RQ_N				<i>fusedparopt_SD</i>				<i>fusedparopt_LBFGS</i>			
	Mean	StdDev	Mdn	IQR	Mean	StdDev	Mdn	IQR	Mean	StdDev	Mdn	IQR
250	0.43	0.87	0.09	0.09	0.10	0.05	0.08	0.05	0.11	0.04	0.11	0.04
500	2.1	4.0	0.47	0.35	0.56	0.22	0.54	0.22	0.65	0.22	0.63	0.25
1000	10	19	2.2	1.5	2.5	0.92	2.3	0.91	2.9	1.1	2.8	1.2
2000	57	104	14	12	14	5	12	5	14	5	13	5
4000	382	688	100	97	86	28	81	35	93	27	90	33
8000	2371	4003	699	540	589	197	568	254	652	235	605	233

Using different m values in *fusedparopt_LBFGS* method for the solution of TPEPs

Example 3 We design a new simulation by generating all A_i, B_i, C_i ($i = 1, 2$) matrices and initial vectors randomly with the help of *rand()* function in MATLAB, so that the tensor product T_0 of this system is nonsingular. 100 different equation systems are generated in each distinct size from 250×250 to 8000×8000 . We solve each system of equations with the *fusedparopt_LBFGS* method. Thus, we display the means of the iteration number and computation times(second), and their standard deviations in Table 6 for $m = 3$ and 17. When we use m values larger than 17 such as 21, 25 and 29, the results are similar to those of 17. Table 6 shows that using larger m values in *fusedparopt_LBFGS* method for these kinds of problems having large-scale matrices gives a small effect in computational cost at the solution.

As a result, it is known that the L-BFGS method alone has a lower cost than the SD method. However, we could not see a big difference between solution costs of the fused algorithms

Table 6. Statistics (means of the iteration number and computation times (second) and their standard deviations) of the *fusedparopt_LBFGS* method with different m values for 100 cases in different sizes of matrices

SIZE	<i>fusedparopt_LBFGS</i> (m=3)				<i>fusedparopt_LBFGS</i> (m=17)			
	Iter.	StdDev	Time	StdDev	Iter.	StdDev	Time	StdDev
250	32	18	0.11	0.06	32	16	0.1	0.04
500	34	15	0.67	0.22	35	15	0.66	0.21
1000	35	13	2.99	0.87	35	13	2.97	0.8
2000	36	11	15	4	37	13	16	5
4000	43	17	96	32	41	12	91	24
8000	50	21	656	258	48	17	625	210

fusedparopt_SD and *fusedparopt_LBFGS*, since the weight of Newton's method in the solution costs is more dominant in our fused algorithms as discussed in Section 3 previously.

An application: Lamé's system

We consider Lamé's system in trigonometric form with angle $0 < \chi < 2\pi$:

$$[1 - k^2 \cos^2(\phi)]L''(\phi) + k^2 \sin(\phi)\cos(\phi)L'(\phi) + [k^2\rho(\rho + 1)\sin^2(\phi) + \delta]L(\phi) = 0, \quad 0 < \phi < \pi, \quad (4)$$

$$[1 - k'^2 \cos^2(\theta)]N''(\theta) + k'^2 \sin(\theta)\cos(\theta)N'(\theta) + [k'^2\rho(\rho + 1)\sin^2(\theta) - \delta]N(\theta) = 0, \quad 0 < \theta < \pi/2, \quad (5)$$

where $k = \sin(|\pi - \chi|/2)$ and $k^2 + k'^2 = 1$. The boundary conditions are $L(0) = L'(\pi) = 0$ and $N'(0) = N'(\pi/2) = 0$ if $0 < \chi < \pi$ or $N(0) = N'(\pi/2) = 0$ if $\pi < \chi < 2\pi$. This Lamé's system of differential Eqs. (4) and (5) can be transformed into a TPEP using Chebyshev collocation. Therefore, we obtain nonsingular A_1 and singular A_2 matrices. Here, the fact that A_2 is singular could make the problem solution difficult from time to time. See [7] for more details and other references.

Comparing methods using given initial values

Example 4 We choose $\chi = 0.125\pi$ and transform differential equations (4) and (5) into TPEPs with sizes 1000 and 2000 using the function *lamé_mep* in toolbox [10] from Matlab. The initial vectors are found in files named *Lamé_initvec_N1000.mat* and *Lamé_initvec_N2000.mat* with sizes 1000 and 2000 respectively in supplementary. The convergences are calculated by residual norm $(\|A_1\mathbf{x}_i - \lambda_i B_1\mathbf{x}_i - \mu_i C_1\mathbf{x}_i\|^2 + \|A_2\mathbf{y}_i - \lambda_i B_2\mathbf{y}_i - \mu_i C_2\mathbf{y}_i\|^2)^{\frac{1}{2}}$ and the computation time is given in seconds.

We give examples showing the limitation of RQ_N method in Table 7. It is seen that RQ_N method has a convergence problem up to 1015 iterations in Table 7a. We can eliminate this problem by using our proposed fused algorithms. *fusedparopt_SD* and *fusedparopt_LBFGS* methods give solutions around $1e-8$ convergence at similar costs in Table 7b and Table 7c.

In addition, we solve these problems using the function *twopareigs* in toolbox [10]. The *twopareigs* method computes solution in 3.30 seconds for size 1000 and in 21.98 seconds for size 2000. These costs seem to be higher than the costs of our fused algorithms.

Simulation and convergence diagram for Lamé's system

Example 5 We make a simulation using Lamé's system that has transformed to a TPEP having size 1000 as in previous Example 4. We generate 500 different initial vectors randomly with the help of *rand()* function in MATLAB. We run our algorithms until the residual norm $< 1e-7$ is satisfied. If not satisfied, we stop the algorithm when the iteration number is 1015.

Our aim is to have an idea about behaviors for numbers of iterations and computational times on the obtained solutions in general. We can use convergence diagrams in Figure 3 and Figure 4 to

Table 7. Some Lamé’s system examples solving in different methods where the same initial vectors are used (C.TIME is in seconds)

(a) Results obtained with RQ_N method

INITIAL VECTOR	METHOD	ITER	CONV	C.TIME
Lame_initvec_N1000	RQ + Newton	1014	3.79E+01	53
Lame_initvec_N2000	RQ + Newton	1014	1.02E+05	282

(b) Results obtained with fusedparopt_SD method

INITIAL VECTOR	METHOD	ITER	CONV	C.TIME
Lame_initvec_N1000	RQ + Newton[SD _{bls}]	17	2.70E-08	1.50
Lame_initvec_N2000	RQ + SD _{qpi} + Newton	33	4.54E-08	10.66

(c) Results obtained with fusedparopt_LBFGS method

INITIAL VECTOR	METHOD	ITER	CONV	C.TIME
Lame_initvec_N1000	RQ + Newton[LBFGS _{bls}]	19	5.51E-08	1.81
Lame_initvec_N2000	RQ + LBFGS _{bls} + Newton	22	7.58E-08	10.22

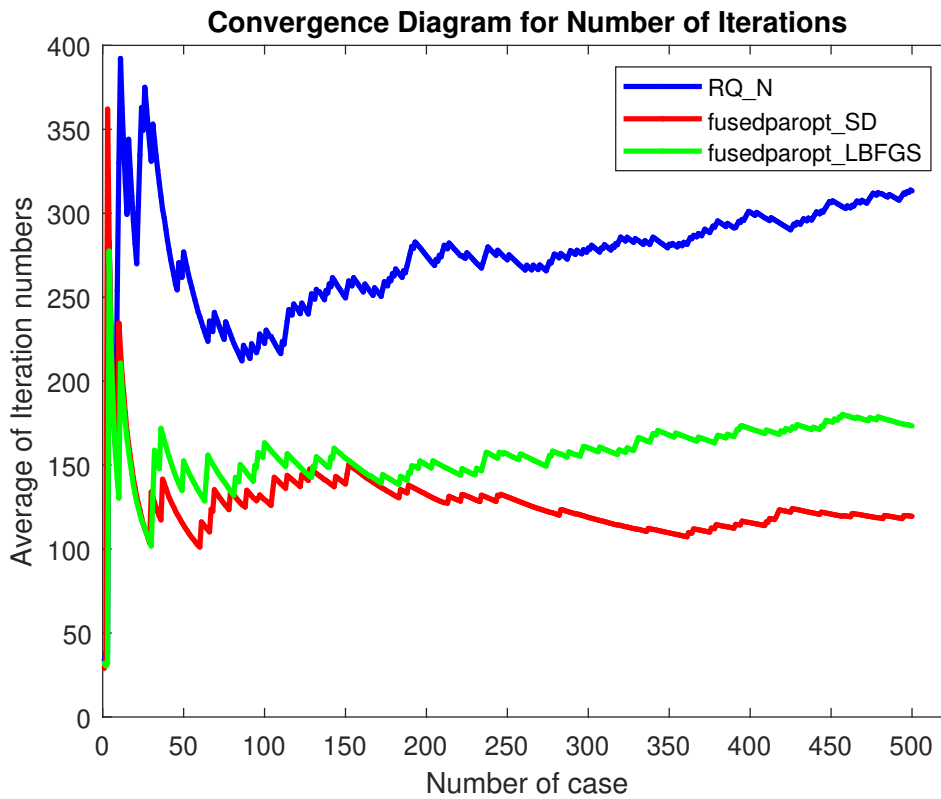


Figure 3. Convergence diagrams according to numbers of iteration for Lamé’s system having size 1000 matrices

examine these behaviors. These diagrams were created based on the average numbers of iteration or computational times¹. We can observe whether numbers of iterations or computational times converge to an average number or not.

There is a constant tendency to increase in the RQ_N graph. Since too many solutions do not

1 In general, the reason why Monte Carlo simulation works is the Law of Large Numbers (see [34, 49]).

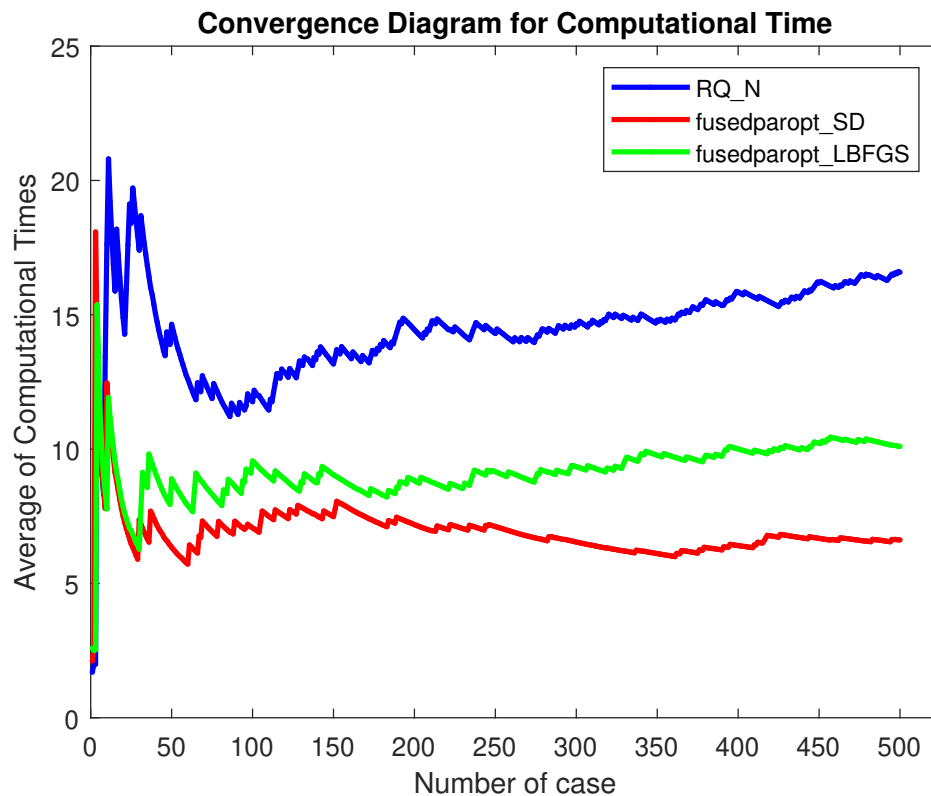


Figure 4. Convergence diagrams according to computational times for Lamé’s system having size 1000 matrices

satisfy the condition for the residual norm $< 1e-7$. So, they reach 1015 iterations. However, this situation is less common in our fusedparopt algorithms. So, our fusedparopt graphs are more stable than RQ_N graph.

5 Conclusions

In this paper, we present a new method *fusedparopt_LBFGS* via fusedparopt algorithm having several variations for TPEPs. Moreover, we show the convergence diagrams of *fusedparopt_LBFGS*, RQ_N and *fusedparopt_SD* algorithms in terms of the number of iterations.

The L-BFGS method alone is less costly than the SD method and this is attractive (see [37], [38], [50] and [51]). However, we observe that L-BFGS method alone is not robust for small values of m (such as $m = 3, 17, 21$) where the m most recent vector pairs are kept, in the solution of TPEPs. This result is consistent with [40] for different applications. Moreover, we test our fused algorithm *fusedparopt_LBFGS*. While high-cost solutions are obtained with the RQ_N method, we have more effective solutions with the *fusedparopt_LBFGS* and *fusedparopt_SD* methods. In addition, it is seen that the results obtained by the *fusedparopt_LBFGS* and *fusedparopt_SD* methods are close to each other and comparable, depending on the matrix type. Thus, we see that *fusedparopt_LBFGS* and *fusedparopt_SD* methods achieve robustness, reasonable cost and good performance for TPEPs based on our data set.

According to the application on Lamé’s system, we find clearly that our fused algorithm is more successful than the RQ_N method. While the RQ_N method has a convergence problem, our proposed *fusedparopt_LBFGS* method converges for the same initial conditions. Besides, our algorithm can converge earlier than the state-of-art twopareigs method ([7, 10]) for the Lamé’s system.

We plan to compare the proposed method with alternative methods such as subspace methods in

another future work.

Declarations

Use of AI tools

The authors declare that they have not used Artificial Intelligence (AI) tools in the creation of this article.

Data availability statement

Dataset files `Lame_initvec_N1000.mat` and `Lame_initvec_N2000.mat` are available at https://github.com/aduran2005/fusedparopt_LBFGS.

Ethical approval (optional)

The authors state that this research complies with ethical standards. This research does not involve either human participants or animals.

Consent for publication

Not applicable

Conflicts of interest

The authors declare that they have no conflict of interest.

Funding

No funding was received for this research.

Author's contributions

H.Ü.Ö.: Conceptualization, Methodology, Data Curation, Writing-Original Draft Preparation-Review & Editing, Formal Analysis, Investigation, Software, Validation, Visualization. A.D.: Supervision, Conceptualization, Methodology, Investigation, Software, Formal analysis, Writing-Original Draft Preparation-Review & Editing, Validation. All authors have read and agreed to the published version of the manuscript

Acknowledgements

Not applicable

References

- [1] Atkinson F.V. Multiparameter spectral theory. *Bulletin of the American Mathematical Society*, 74, 1-27, (1968). [[CrossRef](#)]
- [2] Atkinson F.V. *Multiparameter Eigenvalue Problems: Matrices and Compact Operations*. Academic Press: New York, (1972).
- [3] Volkmer, H. *Multiparameter Problems and Expansion Theorems*. Springer-Verlag: New York, (1988).
- [4] Özer, H.Ü., Duran, A. and Duran, F.S. Fused parameter optimization to find eigenvalue and eigenvectors for three-parameter eigenvalue problems. *Submitted*.
- [5] Teodorescu, P.P. *Treatise on Classical Elasticity*. Springer: Dordrecht, (2013). [[CrossRef](#)]

- [6] Abapolova, E.A. and Soldatov, A.P. Lamé system of elasticity theory in a plane orthotropic medium. *Journal of Mathematical Sciences*, 157, 387-394, (2009). [[CrossRef](#)]
- [7] Plestenjak, B., Gheorghiu, C.I. and Hochstenbach, M.E. Spectral collocation for multiparameter eigenvalue problems arising from separable boundary value problems. *Journal of Computational Physics*, 298, 585-601, (2015). [[CrossRef](#)]
- [8] Boersma, J. and Jansen, J.K.M. *Electromagnetic field singularities at the tip of an elliptic cone*. Ph.D. Thesis, Department of Mathematics and Computing Science, Technische Universiteit Eindhoven, (1990).
- [9] Morrison, J.A. and Lewis, J.A. Charge singularity at the corner of a flat plate. *SIAM Journal on Applied Mathematics*, 31(2), 233-250, (1976). [[CrossRef](#)]
- [10] Plestenjak, B. MultiParEig MATLAB central file exchange, (Retrieved October 27, 2023). <https://www.mathworks.com/matlabcentral/fileexchange/47844-multipareig>
- [11] Bailey, P.B. The automatic solution of two-parameter Sturm-Liouville eigenvalue problems in ordinary differential equations. *Applied Mathematics and Computation*, 8(4), 251-259, (1981). [[CrossRef](#)]
- [12] Ji, X. On 2D bisection method for double eigenvalue problems. *Journal of Computational Physics*, 126(1), 91-98, (1996). [[CrossRef](#)]
- [13] Jarlebring, E. and Hochstenbach, M.E. Polynomial two-parameter eigenvalue problems and matrix pencil methods for stability of delay-differential equations. *Linear Algebra and its Applications*, 431(3-4), 369-380, (2009). [[CrossRef](#)]
- [14] Molzahn, D. *Power System Models Formulated as Eigenvalue Problems and Properties of Their Solutions*. Ph.D. Thesis, Department of Electrical Engineering, University of Wisconsin-Madison, (2010). [<https://minds.wisconsin.edu/handle/1793/46255>]
- [15] Plestenjak, B. A continuation method for a right definite two-parameter eigenvalue problem. *SIAM Journal on Matrix Analysis and Applications*, 21(4), 1163-1184, (2000). [[CrossRef](#)]
- [16] Hochstenbach, M.E. and Plestenjak B. A Jacobi-Davidson type method for a right definite two-parameter eigenvalue problem. *SIAM Journal on Matrix Analysis and Applications*, 24(2), 392-410, (2002). [[CrossRef](#)]
- [17] Hochstenbach, M.E., Kosir, T. and Plestenjak, B. A Jacobi-Davidson type method for the two-parameter eigenvalue problem. *SIAM Journal on Matrix Analysis and Applications*, 26(2), 477-497, (2004). [[CrossRef](#)]
- [18] Meerbergen, K. and Plestenjak, B. A Sylvester-Arnoldi type method for the generalized eigenvalue problem with two-by-two operator determinants. *Numerical Linear Algebra with Applications*, 22(6), 1131-1146, (2015). [[CrossRef](#)]
- [19] Ringh, E. and Jarlebring, E. Nonlinearizing two-parameter eigenvalue problems. *SIAM Journal on Matrix Analysis and Applications*, 42(2), 775-799, (2021). [[CrossRef](#)]
- [20] Blum, E.K. and Geltner, P.B. Numerical solution of eigentuple-eigenvector problems in Hilbert spaces by a gradient method. *Numerische Mathematik*, 31, 231-246, (1978). [[CrossRef](#)]
- [21] Blum, E.K. and Curtis, A.R. A convergent gradient method for matrix eigenvector-eigentuple problems. *Numerische Mathematik*, 31(3), 247-263, (1978). [[CrossRef](#)]
- [22] Browne, P.J. and Sleeman, B.D. A numerical technique for multiparameter eigenvalue problems. *IMA Journal of Numerical Analysis*, 2(4), 451-457, (1982). [[CrossRef](#)]
- [23] Dong, B. The homotopy method for the complete solution of quadratic two-parameter

- eigenvalue problems. *Journal of Scientific Computing*, 90, 18, (2022). [[CrossRef](#)]
- [24] Eisenmann, H. and Nakatsukasa, Y. Solving two-parameter eigenvalue problems using an alternating method. *Linear Algebra and its Applications*, 643, 137-160, (2022). [[CrossRef](#)]
- [25] Košir, T. and Plestenjak, B. On the singular two-parameter eigenvalue problem II. *Linear Algebra and its Applications*, 649, 433-451, (2022). [[CrossRef](#)]
- [26] Muhič, A. and Plestenjak, B. On the singular two-parameter eigenvalue problem. *Electronic Journal of Linear Algebra*, 18, 420-437, (2009). [[CrossRef](#)]
- [27] Hochstenbach, M.E., Meerbergen, K., Mengi, E. and Plestenjak, B. Subspace methods for three-parameter eigenvalue problems. *Numerical Linear Algebra with Applications*, 26(4), e2240, (2019). [[CrossRef](#)]
- [28] Rodriguez, J.I., Du, J.H., You, Y. and Lim, L.H. Fiber product homotopy method for multiparameter eigenvalue problems. *Numerische Mathematik*, 148, 853-888, (2021). [[CrossRef](#)]
- [29] Dong, B., Yu, B. and Yu, Y. A homotopy method for finding all solutions of a multiparameter eigenvalue problem. *SIAM Journal on Matrix Analysis and Applications*, 37(2), 550-571, (2016). [[CrossRef](#)]
- [30] Trefethen L.N. and Bau III, D. *Numerical Linear Algebra*. SIAM: Philadelphia, USA, (1997). [[CrossRef](#)]
- [31] Duran, A. and Caginalp, G. Parameter optimization for differential equations in asset price forecasting. *Optimization Methods & Software*, 23(4), 551-574, (2008). [[CrossRef](#)]
- [32] Duran, A. and Tuncel, M. Evaluation of a new parallel numerical parameter optimization algorithm for a dynamical system. In Proceedings, *AIP Conference Proceedings*, pp. 090052, Pizzo Calabro, Italy, (2016, October). [[CrossRef](#)]
- [33] Tuncel, M. and Duran, A. Effectiveness of grid and random approaches for a model parameter vector optimization. *Journal of Computational Science*, 67, 101960, (2023). [[CrossRef](#)]
- [34] Jackel, P. *Monte Carlo Methods in Finance*. John Wiley & Sons, England, (2002).
- [35] Kangal, F., Meerbergen, K., Mengi, E. and Michiels, W. A subspace method for large-scale eigenvalue optimization. *SIAM Journal on Matrix Analysis and Applications*, 39(1), 48-82, (2018). [[CrossRef](#)]
- [36] Byrd, R.H., Hansen, S.L., Nocedal, J. and Singer, Y. A stochastic quasi-Newton method for large-scale optimization. *SIAM Journal on Optimization*, 26(2), 1008–1031, (2016). [[CrossRef](#)]
- [37] Nocedal, J. Updating quasi-Newton matrices with limited storage. *Mathematics of Computation*, 35(151), 773-782, (1980). [[CrossRef](#)]
- [38] Saputro, D.R.S. and Widyaningsih, P. Limited memory Broyden-Fletcher-Goldfarb-Shanno (L-BFGS) method for the parameter estimation on geographically weighted ordinal logistic regression model (GWOLR). In Proceedings, *AIP Conference Proceedings*, pp. 040009, Yogyakarta, Indonesia, (2017, August). [[CrossRef](#)]
- [39] Dalvand, Z. and Hajarian, M. Solving generalized inverse eigenvalue problems via L-BFGS-B method. *Inverse Problems in Science and Engineering*, 28(12), 1719-1746, (2020). [[CrossRef](#)]
- [40] Nocedal, J. and Wright, S.J. *Numerical Optimization*. Second Edition, Springer: New York, (2006).
- [41] Peressini, A.L., Sullivan, F.E. and Uhl Jr., J.J. *The Mathematics of Nonlinear Programming*. Springer New York: New York, (1988).

- [42] Freund, R.M. *The Steepest Descent Algorithm for Unconstrained Optimization and a Bisection Line-search Method*. Ph.D. Thesis, Department of Electrical Engineering and Computer Science, The University of Massachusetts, (2004). [https://ocw.mit.edu/courses/15-084j-nonlinear-programming-spring-2004/resources/lec5_steep_desce/]
- [43] Burden, R.L. and Faires, J.D. *Numerical Analysis*. Brooks/Cole Cengage Learning: Boston, (2011).
- [44] Boyd, S. and Vandenberghe, L. *Convex Optimization*. Cambridge University Press: Cambridge, (2004).
- [45] Barzilai, J. and Borwein, J.M. Two-point step size gradient methods. *IMA Journal of Numerical Analysis*, 8(1), 141-148, (1988). [[CrossRef](#)]
- [46] Horn, R.A. and Johnson, C.R. *Topics in Matrix Analysis*. Cambridge University Press: Cambridge, (1991).
- [47] Gil, M. Bounds for the spectrum of a two parameter matrix eigenvalue problem. *Linear Algebra and its Applications*, 498, 201-218 (2016). [[CrossRef](#)]
- [48] Watkins, D.S. *Fundamentals of Matrix Computations*. Wiley-Interscience, A John Wiley & Sons: New York, (2002).
- [49] Jonsson, M. *An Introduction to Monte Carlo Simulations*. Textbook for Numerical Methods with Financial Applications, Department of Mathematics, University of Michigan, (2006).
- [50] June, L.W. and Abu Hassan, M. Modifications of the limited memory bfgs algorithm for large-scale nonlinear optimization. *Mathematical Journal of Okayama University*, 47(1), 175-188, (2005). [[CrossRef](#)]
- [51] Mokhtari, A. and Ribeiro, A. Global convergence of online limited memory BFGS. *Journal of Machine Learning Research*, 16(98), 3151-3181, (2015).

Mathematical Modelling and Numerical Simulation with Applications (MMNSA)
(<https://dergipark.org.tr/en/pub/mmnsa>)



Copyright: © 2024 by the authors. This work is licensed under a Creative Commons Attribution 4.0 (CC BY) International License. The authors retain ownership of the copyright for their article, but they allow anyone to download, reuse, reprint, modify, distribute, and/or copy articles in MMNSA, so long as the original authors and source are credited. To see the complete license contents, please visit (<http://creativecommons.org/licenses/by/4.0/>).

How to cite this article: Özer, H.Ü. and Duran, A. (2024). A new algorithm using L-BFGS for two-parameter eigenvalue problems from Lamé's system and simulation. *Mathematical Modelling and Numerical Simulation with Applications*, 4(4), 448-468. <https://doi.org/10.53391/mmnsa.1523702>