A STRUCTURE-PRESERVED OPTIMIZATION ALGORITHM FOR SOLVING MULTIPARAMETER EIGENVALUE PROBLEMS*

ZHENYU XU† AND JINLONG ZHENG‡

Abstract. We survey about the eigenvalue problems with eignevector nonlinearties, especially its transformation such as multiparameter eigenvalue problems and generalized eigenvalue problems, and a variety of numerical optimization techniques. Emphasis is given to developing a structure-perserved algorithm to solve problems. As a motivation, we prove that our quasi-Newton algorithm can solve the question at a lower computation cost than selected benchmark residual iteration method. We also conduct detailed analysis and experiment on the selection of inverse matrix of quasi-Newton method and using Rayleigh quotient as one of the acceleration approaches.

Key words. nonlinear eigenvalue problems, multiparameter eigenvalue problems, quasi-Newton method, structure-preserved, residual iteration method, inverse iteration method, Rayleigh qoutient

AMS subject classifications. 65F15, 49M15, 65Y20, 15A18

1. Introduction and Preliminary.

Nonlinear eigenvalue problems (NEP) raise a high demand for efficient algorithms and computer implementation. Most of NEP, including issues that we are concerned about in this article, have the following form:

$$T(\lambda, x)x = 0, \quad x \neq 0, \tag{1.1}$$

where $T : \{\mathbb{F}, \mathbb{F}^n\} \mapsto \mathbb{F}^m$ is defined as a nonlinear function of eigenvalue λ and eigenvector x both in field $\mathbb{F}$, real or complex in most cases.

The nonlinearity of target eigenvalue often greatly increases the difficulty of algorithmic convergence, which makes linearization a practical technique when dealing with most of linearizable eigenvalue problems. However, a special class of NEP with eigenvector nonlinearities (NEPv) focuses on the nonlinearity of the eigenvectors and dealing with linear eigenvalues, and tends to make linearization quite expensive when its system contains many nonlinear eigenvector terms. In the following research, we care about the following NEPv

$$\big(A + \lambda B + f_1(x)C_1 + \cdots + f_m(x)C_m\big)x = 0, \tag{1.2}$$

$$f_i(x) = \frac{r_i^{\mathrm{T}} x}{s_i^{\mathrm{T}} x}, \quad i = 1, 2, \ldots, m \tag{1.3}$$

where A, B, $C_i \in \mathbb{C}^{n\times n}$ and r_i, $s_i \in \mathbb{C}^n$ for $i = 1, 2, \ldots, m$. A basic property of scaling invariant of f_i, i.e. $f_i(\alpha x) = f_i(x)$ for complex number $\alpha \neq 0$, guarantees that the eigenvectors can be stretched in its direction and thus normalization is required.

NEPv arise both in the field of natural science and data science. A common application of natural science is in the field of quantum chemistry for electronic structure calculations [8], where the discretization of the Kohn-Sham and Hartree-Fock equations rely on solving NEPv. Spectral clustering can be applied in large-scale graph problem, and nonlinear generalization of its relative standard graph Laplacian leads to NEPv [2]. In [9], the equivalence is established between the least-squares problem and solving the natural discrete optimization problem about Rayleigh quotient,

*Submitted to Prof. Jiang on January 15, 2023. Codes are available here

†FDU School of Data Science (22210980117@m.fudan.edu.cn).

‡FDU School of Data Science (22210980123@m.fudan.edu.cn).

which can show that a new model for core-periphery detection in network science is also equivalent to NEPv.

Linearization helps us explore the nature of NEPv, and a common approach to deal with equation 1.3 is to multiply the denominator over and introduce new accompanion eigenvalues μ_i's to eliminate the nonlinearity with respect to x. We can rewrite NEPv as following multiparameter eigenvalue problem (MEP)

$$(A + \lambda B + \mu_1 C_1 + \mu_2 C_2 + \cdots + \mu_m C_m)x = 0, \tag{1.4}$$

$$\mu_i s_i^{\mathrm{T}} x = r_i^{\mathrm{T}} x, \quad i = 1, 2, \ldots, m, \tag{1.5}$$

where $r_i^{\mathrm{T}} x$ is not necessary to be nonzero, because strong linearization holds to establish equivalence between $\mu_i = \infty$ and zero eigenvalue of inv T [4]. A mild modification guarantees the matrix form of the eigenvalue coefficients as we multiply nonzero row vector to the left side of equation 1.5. In any statement that follows about MEP, we will replace equation 1.5 with the generalizd eigenvalue problems (GEP)

$$\mu_i g_i s_i^{\mathrm{T}} x = g_i r_i^{\mathrm{T}} x, \quad i = 1, 2, \ldots, m, \tag{1.6}$$

where $g_i \in \mathbb{C}^n \setminus \{0\}$, or

$$\left((A + g_i r_i^{\mathrm{T}}) + \lambda B + \mu_1 C_1 + \cdots + \mu_i (C_i + g_i s_i^{\mathrm{T}}) + \cdots + \mu_m C_m\right) x = 0, \quad i = 1, \ldots, m, \tag{1.7}$$

that is added by equation 1.4 to improve the numerical stability of the whole eigenvalue system.

Combine equation 1.4 and 1.7, and consider the following standard form of MEP (MEPs),

$$\begin{aligned} A_{1,0}x_1 &= (\lambda_1 A_{1,1} + \lambda_2 A_{1,2} + \cdots + \lambda_{m+1} A_{1,m+1})x_1, \\ A_{2,0}x_2 &= (\lambda_1 A_{2,1} + \lambda_2 A_{2,2} + \cdots + \lambda_{m+1} A_{2,m+1})x_2, \\ &\vdots \\ A_{m+1,0}x_{m+1} &= (\lambda_1 A_{m+1,1} + \lambda_2 A_{m+1,2} + \cdots + \lambda_{m+1} A_{m+1,m+1})x_{m+1}, \end{aligned} \tag{1.8}$$

where $A_{i,j} \in \mathbb{C}^{n\times n}$ and we aim to find some $x_i \in \mathbb{C}^n$ and $\lambda_i \in \mathbb{C}$ for $1 \le i, j \le m+1$. MEPs is very close to the linear equations system in form, and it can be proved that Crammer's law applies as well. We can see [1] for more details. The operator determinant of a $k \times k$ array of matrices $\{A_{i,j}, 1 \le i, j \le k\}$ is defined as

$$\begin{aligned} \begin{vmatrix} A_{1,1} & A_{1,2} & \cdots & A_{1,k} \\ A_{2,1} & A_{2,2} & \cdots & A_{2,k} \\ \vdots & \vdots & \ddots & \vdots \\ A_{k,1} & A_{k,2} & \cdots & A_{k,k} \end{vmatrix}_{\otimes} &= (-1)^0 A_{1,1} \otimes \begin{vmatrix} A_{2,2} & A_{2,3} & \cdots & A_{2,k} \\ A_{3,2} & A_{3,3} & \cdots & A_{3,k} \\ \vdots & \vdots & \ddots & \vdots \\ A_{k,2} & A_{k,3} & \cdots & A_{k,k} \end{vmatrix}_{\otimes} \\ &\quad + \cdots + (-1)^{k-1} A \otimes \begin{vmatrix} A_{2,1} & A_{2,2} & \cdots & A_{2,k-1} \\ A_{3,1} & A_{3,2} & \cdots & A_{3,k-1} \\ \vdots & \vdots & \ddots & \vdots \\ A_{k,1} & A_{k,2} & \cdots & A_{k,k-1} \end{vmatrix}_{\otimes}, \end{aligned} \tag{1.9}$$

$$\left|A_{i,j}\right|_{\otimes} = A_{i,j}, \quad 1 \le i, j \le k,$$

where $\otimes$ denotes Kronecker product between matrices and vectors. An equivalent form of MEPv derived using Crammer's law (MEPc) is denoted by

$$\Delta_i z = \lambda_i \Delta_0 z, \quad i = 1, 2, ..., m+1, \tag{1.10}$$

where the target eigenvector $z = x_1 \otimes x_2 \otimes \cdots \otimes x_{m+1}$, and Δ_0 is the operator determinant of $\{A_{i,j}, 1 \le i, j \le m+1\}$, Δ_i is the result of Δ_0 replacing the i-th column with $\{A_{i,0}, 1 \le i \le n\}$, i.e.,

$$\begin{aligned}
\Delta_0 &= \begin{vmatrix} A_{1,1} & A_{1,2} & \cdots & A_{1,k} \\ A_{2,1} & A_{2,2} & \cdots & A_{2,k} \\ \vdots & \vdots & \ddots & \vdots \\ A_{m+1,1} & A_{m+1,2} & \cdots & A_{m+1,m+1} \end{vmatrix}_{\otimes}, \\
\Delta_i &= \begin{vmatrix} A_{1,1} & \cdots & A_{1,i-1} & A_{1,0} & A_{1,i+1} & \cdots & A_{1,m+1} \\ A_{2,1} & \cdots & A_{2,i-1} & A_{2,0} & A_{2,i+1} & \cdots & A_{2,m+1} \\ \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ A_{m+1,1} & \cdots & A_{m+1,i-1} & A_{m+1,0} & A_{m+1,i+1} & \cdots & A_{m+1,m+1} \end{vmatrix}_{\otimes},
\end{aligned} \tag{1.11}$$

and thus $\Delta_i \in \mathbb{C}^{n^{m+1}}$, resulting in a matrix of extreme large scale when m and n are slightly large. For example, when $n = 5$ and $m = 4$, storage of 3125×3125 elements is required to hold each Δ_i and computational cost is much larger for most GEP algorithms such as QZ iteration if no simplification is applied.

Many NEPv are complex beacuse of its multivariates and thus its generated MEPc combine a large number of GEP. It would be much easier if you don't have to deal with all these equations simultaneously, and it is achievable. The main simplification of equation 1.4 and 1.7 compared to equation 1.8 is that we just need to make sure that $x_1 = x_2 = \cdots = x_{m+1}$ which produce symmetric z and asymmetric otherwise. Other generated eigenvalues λ_i for $2 \le i \le m+1$ is corresponding to a rank-one matrix and are not considered during calculation.

It is feasible to find all possible solutions of

$$\Delta_1 z = \lambda_i \Delta_0 z \tag{1.12}$$

including each symmetric or asymmetric $z \in \mathbb{C}^{n^{m+1}}$, but only those symmetric are our target solution. A basic idea to simplify the solving procedure is to develop a symmetric-structure-preserved optimization algorithm to find eigenvalue around some fixed points and its corresponding eigenvector. A fast iterative method is more preferred than QZ iteration or Jacobi-Davision algorithm.

Residual iteration is proposed in [6] and well studied in [3], and shows stable convergence property and nice structure-preserved property. However, this algorithm is also computationally heavy because of its lots of work dealing with huge Jacobi systems. In the following chapter we will try to develop a structure-preserved quasi-Newton-based method to reduce its computational cost. Main difficulty is that the approximation of the Jacobi matrix needs well designing to maintain the symmetric structure of eigenvector. Residual iteration and QZ iteration will be our benchmark.

One remark is on the choice of g_i in equation 1.6. It is suggested that almost all choice of g_i is effective, but we do not care about eigenvalue at infinity despite of the fact indicated by strong linearization theorem that we can still work the corresponding eigenvector out by solving inverse MEPv. In this sense, g_i which satisfies $g_i^{\mathrm{T}} x_i \neq 0$

is preferred. We can find some interesting deduction in [3] with respect to residual iteration method, and we have done some experiments on the choice of g_i. A general view of how to reduce the backward error of the algorithm by choosing appropriate g_i still remains to be further studied.

In this section we have given a general view about NEPv and MEPc. In the following context, section 2 is about the basic idea of our qausi-Newton based algorithm, section 3 is about the convexity analysis of NEPv and MEPc, and relative improvement of our quasi-Newton algorithm, along with the result of numerical experiment of our algorithm.

2. Quasi-Newton Based Structure-Preserved Algorithm. A common idea to apply Newton's method is to use the Jacobi structure of NEP to perform inverse iteration [7]. For NEP equation 1.1, a stepwise iterative method is defined as

(2.1)
$$\begin{bmatrix} \frac{\partial T(\lambda^{(k)},x^{(k)})}{\partial x^{(k)}}^{\mathrm{H}} x^{(k)} + T(\lambda^{(k)},x^{(k)}) & \frac{\partial T(\lambda^{(k)},x^{(k)})}{\partial \lambda^{(k)}} \\ v^{\mathrm{H}} & 0 \end{bmatrix} \begin{bmatrix} x^{(k+1)}-x^{(k)} \\ \lambda^{(k+1)}-\lambda^{(k)} \end{bmatrix} = -\begin{bmatrix} T(\lambda^{(k)},x^{(k)})x^{(k)} \\ v^{\mathrm{H}}x-1 \end{bmatrix},$$

where $v \in \mathbb{C}^n$ and an effective analytic function $v^{\mathrm{H}}x-1$ will prevent x from converging to 0. This equation shows how to update variable from k-th step to $(k+1)$-th step. For our convenience we can formulate equation 2.1 as follows,

$$\begin{aligned} T(\lambda^{(k)},x^{(k)})u^{(k+1)} &= \left(\frac{\partial T(\lambda^{(k)},x^{(k)})}{\partial x^{(k)}}^{\mathrm{H}} x^{(k)} + T(\lambda^{(k)},x^{(k)})\right) x^{(k)}, \\ \lambda^{(k+1)} &= \lambda^{(k)} - \frac{v^{\mathrm{H}}x^{(k)}}{v^{\mathrm{H}}u^{(k+1)}}, \\ x^{(k+1)} &= \frac{u^{(k+1)}}{v^{\mathrm{H}}u^{(k+1)}}. \end{aligned} \tag{2.2}$$

.

Basically, we take a similar approach, but with some improvements in the implementation details. We focus on equation 1.4, and 1.7, and replace x with the corresponding x_i. Structure that we need to preserve is that from $x_1^{(0)} = x_2^{(0)} = \cdots = x_{m+1}^{(0)}$, we can keep the fact that $x_1^{(k)} = x_2^{(k)} = \cdots = x_{m+1}^{(k)}$ during iteration. Combine these two equations and rewrite them as below,

$$T_i(\lambda,\mu_1,\mu_2,\ldots,\mu_m)x_i = 0, \quad i = 1,2,\ldots,m+1, \tag{2.3}$$

and we define the constraint function $c_i(x_i) = 0$ for $i = 1,2,\ldots,m+1$. Newton's method applied to this system can be expressed as

(2.4)
$$J(x_1^{(k)},\ldots,x_{m+1}^{(k)},\lambda,\mu_1^{(k)},\ldots,\mu_{m+1}^{(k)}) \begin{bmatrix} x_1^{(k+1)}-x_1^{(k)} \\ \vdots \\ x_{m+1}^{(k+1)}-x_{m+1}^{(k)} \\ \lambda^{(k+1)}-\lambda^{(k)} \\ \mu_1^{(k+1)}-\mu_1^{(k)} \\ \vdots \\ \mu_{m+1}^{(k+1)}-\mu_{m+1}^{(k)} \end{bmatrix} = \begin{bmatrix} T_1(\lambda^{(k)},\mu_1^{(k)},\ldots,\mu_{m+1}^{(k)})x_1^{(k+1)} \\ \vdots \\ T_{m+1}(\lambda^{(k)},\mu_1^{(k)},\ldots,\mu_{m+1}^{(k)})x_{m+1}^{(k+1)} \\ c_1(x_1^{(k)}) \\ \vdots \\ c_{m+1}(x_{m+1}^{(k)}) \end{bmatrix}.$$

Generally we will consider analytic function $c_i(x) = v_i^{\mathrm{H}}x - 1$ where $v \in \mathbb{C}^n$ is chosen freely. There is a fair degree of confidence that random selection of v_i's will not

lead to an unconvergent route. Projection constraint is suitable for solving complex problem, while there are better choices in $\mathbb{R}$. Nonanalytic function $c_i(x) = x^{\mathrm{H}}x - 1$ ensures that all directions of x_i are even, without putting excessive weight in a certain direction.

Jacobi matrix in equation 2.4 can be expanded as follows,

$$J(x_1^{(k)}, \ldots, x_{m+1}^{(k)}, \lambda, \mu_1^{(k)}, \ldots, \mu_{m+1}^{(k)}) = \begin{bmatrix} J_{1,1} & J_{1,2} & J_{1,3} \\ J_{2,1} & & \\ J_{3,1} & & \end{bmatrix}, \tag{2.5}$$

and

$$J_{1,1} = \begin{bmatrix} T_1(\lambda^{(k)}, \mu_1^{(k)}, \ldots, \mu_m^{(k)}) & & \\ & \ddots & \\ & & T_{m+1}(\lambda^{(k)}, \mu_1^{(k)}, \ldots, \mu_m^{(k)}) \end{bmatrix}^{\mathrm{H}},$$

$$J_{1,3} = \begin{bmatrix} C_1 + g_1 r_1^{\mathrm{H}} & C_2 & \cdots & C_m \\ C_1 & C_2 + g_2 r_2^{\mathrm{H}} & \cdots & C_m \\ \vdots & \vdots & \ddots & \vdots \\ C_1 & C_2 & \cdots & C_m + g_m r_m^{\mathrm{H}} \end{bmatrix}^{\mathrm{H}}, \tag{2.6}$$

$$J_{3,1} = \begin{bmatrix} \frac{\partial c_1}{\partial x}(x_1) & & \\ & \ddots & \\ & & \frac{\partial c_{m+1}}{\partial x}(x_{m+1}) \end{bmatrix}^{\mathrm{H}},$$

and $J_{2,1}$, $J_{1,2}$ are about λ_i and do not differ from μ_i.

The main difficulty lies in the heavy work about the inverse of $J_{1,3}$ and storing this large-scaled equation in memory. Considering the approximation $P_i(t)$ of each component $T_i\left(\lambda^{(k)}, \mu_1^{(k)}, \ldots, \mu_{m+1}\right)$ of $J_{1,1}$, it will be very cheap to solve systems of linear equations or to find the inverse matrix about $T_i\left(\lambda^{(k)}, \mu_1^{(k)}, \ldots, \mu_{m+1}^{(k)}\right)$. We use hyperparameter t to determine P_i, and $\tilde{J}(t)$ generated from $P_i(t)$'s is the approximation of J.

Assuming normalization with respect to all $c_i(\cdot)$'s is taken after every Jacobi update and $c_1\left(x_1^{(k)}\right) = \cdots = c_{m+1}\left(x_{m+1}^{(k)}\right) = 0$ in k-th step, we can eliminate $J_{3,1}$ and equation 2.4 results in

$$J_{3,1}\tilde{J}_{1,1}^{-1}J_{1,3}\begin{bmatrix} \mu_1^{(k+1)} - \mu_1^{(k)} \\ \vdots \\ \mu_{m+1}^{(k+1)} - \mu_{m+1}^{(k)} \end{bmatrix} = J_{3,1}\tilde{J}_{1,1}^{-1}\begin{bmatrix} T_1(\lambda^{(k)}, \mu_1^{(k)}, \ldots, \mu_{m+1}^{(k)})x_1^{(k+1)} \\ \vdots \\ T_{m+1}(\lambda^{(k)}, \mu_1^{(k)}, \ldots, \mu_{m+1}^{(k)})x_{m+1}^{(k+1)} \end{bmatrix}. \tag{2.7}$$

And $x_i^{(k+1)}$'s can be updated as follows

$$\begin{aligned} u_i^{(k+1)} &= x_i^{(k)} - P_i(t)^{-1}T_i\left(\lambda^{(k)}, \mu_1^{(k)}, \ldots, \mu_m^{(k)}\right)x_i^{(k)}, \\ c(lu_i^{(k+1)}) &= 0, \\ x_i^{(k+1)} &= lu_i^{(k+1)}. \end{aligned} \tag{2.8}$$

The next thing to be done is proving the structure-preserved property of this quasi-Newton based algorithm. Assume $x_1^{(k)} = \cdots = x_{m+1}^{(k)}$. From

$$
\begin{aligned}
&P_1(t)x_1^{(k+1)} + \Big(\Delta\lambda B + \cdots + \Delta\mu_1 C_1 + \Delta\mu_m C_m\Big)x_1^{(k)} = -T_1(\lambda^{(k)}, \mu_1^{(k)}, \ldots, \mu_m^{(k)})x_1^{(k)}, \\
&P_i(t)x_i^{(k+1)} + \Big(\Delta\lambda B + \Delta\mu_1 C_1 + \cdots + \Delta\mu_i(C_i + g_i s_i^{\mathrm{T}}) + \cdots + \Delta\mu_m C_m\Big)x_i^{(k)} \\
&\qquad\qquad = -T_i(\lambda^{(k)}, \mu_1^{(k)}, \ldots, \mu_m^{(k)})x_i^{(k)}.
\end{aligned} \tag{2.9}
$$

We choose $c_i(x) = v^{\mathrm{H}}x - 1$ just like the residual iteration does, and equation 2.9 turns out to be

$$
P_1(t)\left(x_i^{(k+1)} - x_1^{(k+1)}\right) = g_i V_i, \tag{2.10}
$$

where V_i is given from the combination of s_i's. Finally, we left-multiply with $v^{\mathrm{T}}P_1(t)^{-1}$, and it leads to

$$
v^{\mathrm{H}}\left(x_i^{(k+1)} - x_1^{(k+1)}\right) = v^{\mathrm{H}}P_1(t)^{-1}g_i V_i. \tag{2.11}
$$

The normalization ensures that $v^{\mathrm{H}}\left(x_i^{(k+1)} - x_1^{(k+1)}\right) = 0$, and thus $V_i = 0$. Because the approximation matrix $P_1(t)$ we choose is not singular, it gives the conclusion that $x_1^{(k+1)} = \cdots x_{m+1}^{(k+1)}$.

The above give us some hints. First of all, there is few $c_i(\cdot)$ meet our equation. But in every step, we can still conduct another normalization after solving the Jacobi system. Second, approximation $P_1(t)$'s must have a clear inverse matrix, even though they can be very close to sigular matrices. And finally, Jacobi system provides an effective approach to update x_i's, and there are still other ways to update λ_i and μ_i's.

It is also feasiable to solve NEPv directly. Newton's method or quasi-Newton method is also a general way for computing. But we need to deal with nonconvex item $\big(r_i^{\mathrm{T}}x\big)/\big(s_i^{\mathrm{T}}x\big)x$, and in practice it may need much more steps and failed to converge in some cases.

Another idea is to focus on GEP system. The symmetric structure of z can be kept during inverse iteration [3] so that $z = \alpha x_1 \otimes x_1 \otimes \cdots \otimes x_1$ is the solution to the whole system. The basic steps are dicatated by

$$
\begin{aligned}
(\Delta_1 - \sigma\Delta_0)u_{k+1} &= \Delta_0 z_k, \\
z_{k+1} &= \frac{u_{k+1}}{\|u_{k+1}\|_2}
\end{aligned} \tag{2.12}
$$

with $\sigma \in \mathbb{C}$ controlling z to converge to eigenvalue closest to a wanted point. This algorithm has a factor of convergence rate of $\mathcal{O}(|\frac{\sigma-\lambda_1}{\sigma-\lambda_2}|^k)$, where λ_1 is the closest eigenvalue to σ and λ_2 is the second. We can also use shifted iteration approach such as Rayleigh quotient to accelerate the convergence.

Compare quasi-Newton algorithm and inverse iteration. Inverse iteration solving GEP system can converge only in very mild condition, but it costs much more to solve Linear system of scale of $\mathcal{O}(n^{3(m+1)})$ than residual iteration of $\mathcal{O}(n^3m^3)$ and quasi-Newton algorithm of $\mathcal{O}(n^3)$ or $\mathcal{O}(n^2)$ in every step. Generally quasi-Newton algorithm needs more steps to converge to the same precision of inverse iteration and have a lower precision bound.

One remark is the selection of $P_i(t)$. Generally there is no abosolute choice. We can depend on DFP or BFGS generated from Sherman-Morrison-Woodbury formula with respect to F, W norm, and can also decompose $T_i\left(\lambda^{(k)}, \mu_1^{(k)}, \ldots, \mu_m^{(k)}\right)$ just making it easier to do some inverse operations. It is shown that algorithm converges almost for every $P_i(t)$, and thus we consider a fixed $P_i(t)$ and dynamic $P_i(t)$ by decomposing $T_i(\lambda, x)$.

Another remark is the local convergence and global convergence of algorithm. it may differ whether to use dynamic shift. Linear convergence rate is guaranteed in most cases, no matter locally and globally, and quadratic converge can be achieved when using dynamic shift for approximating eigenvalues.

3. Analysis and Numerical Experiment. In this section, we will give a detailed view of the performance of our algorithm. We test our code on two paramter MEP, i.e.,

$$\begin{aligned} -Ax &= \lambda Bx + \mu Cx \\ -A - gr^{\mathrm{H}} &= \lambda Bx + \mu(C - gs^{\mathrm{H}})x \end{aligned} \tag{3.1}$$

with matrix size of 100×100 and in real field. Error with respect to $\lambda^{(k)}$ and $x^{(k)}$ is measured by equation 1.2, the definition of NEPv,

$$\begin{aligned} E &= \left\| \left(A + \lambda^{(k)} B + f_1(x^{(k)})C_1 + \cdots + f_m(x^{(k)})C_m\right)x^{(k)} \right\|_2, \\ f_i(x^{(k)}) &= \frac{r_i^{\mathrm{H}} x^{(k)}}{s_i^{\mathrm{H}} x^{(k)}}, \quad i = 1, 2, \ldots, m, \end{aligned} \tag{3.2}$$

and x is bounded by constraint function $c_i(x)$. We choose $c_i(x) = v^{\mathrm{H}}x - 1$, where v is generated randomly. Inital point $x^{(0)}$ and $\lambda^{(0)}$ is given by

$$\begin{aligned} x^{(0)} &= x^* + \delta x, \\ \lambda^{(0)} &= \lambda^* + \delta\lambda, \end{aligned} \tag{3.3}$$

where x^* and λ^* in complex field are given by another more accurate algorithm such as QZ algorithm. δx and $\delta\lambda$ are perturbations generated by normal distribution with standard variance σ. We control σ as

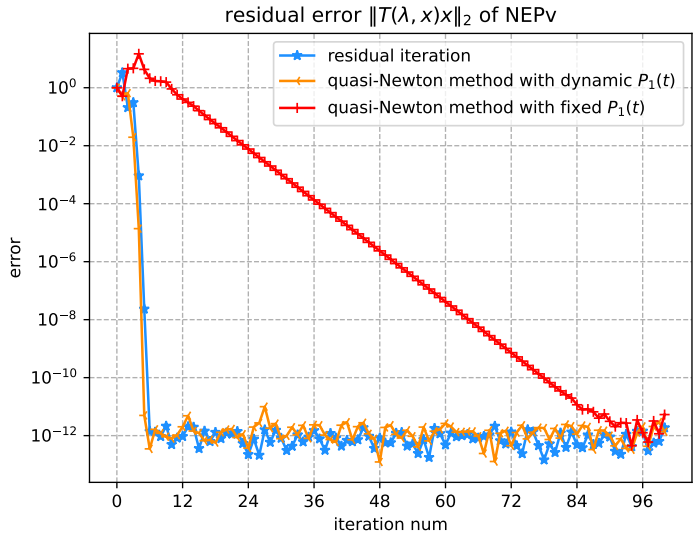
$$\frac{\|\delta x\|_2}{\|x^{(0)}\|_2} = \epsilon, \tag{3.4}$$

and ϵ is a constant to ensure that the random components are of the same scale. We also calculate $z^{(0)} = x^{(0)} \otimes \cdots \otimes x^{(0)}$ and apply it in inverse iteration method. The result shows that inverse iteration converges at a slow rate with a far higher computation cost, both theoretically and practically. In that case we will leave it out in the rest of our pages.

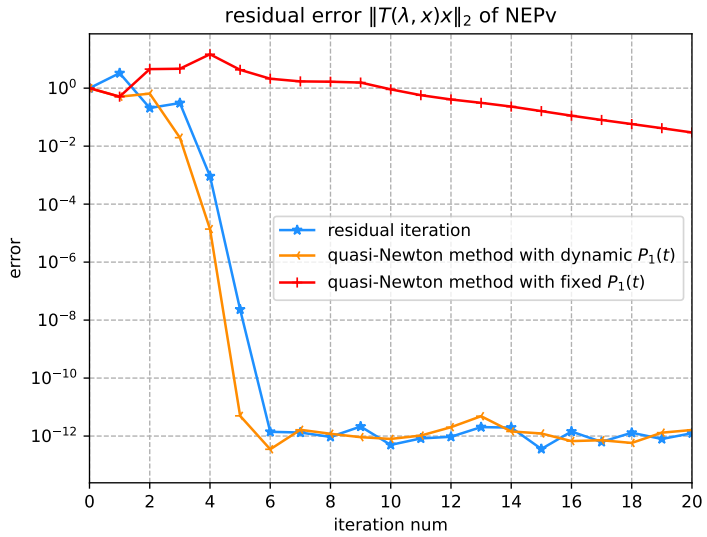
Compare residual iteration, quasi-Newton method with dynamic $P_1(t)$ and fixed $P_1(t)$, and plot the figures to show residuals versus the number of iterations.

Figure 1a shows the residual of the first one hundred iterations of three algorithms, and Figure 1b shows the first twenty steps. All these three algorithms approach the error accuracy about 10^{-12}, while the residual iteration can be slightly better. Residual iteration method and dynamic quasi-Newton method eliminte the first order item of error and approach quadratic convergence, while the fixed quasi-Newton method can only converge at a very low linear rate. However, sometimes we cannot observe

the convergence of quasi-Newton method with fixed $P_1(t)$, perhaps because the choice of initial value is not proper, or the convergence rate is too low and the convergence is not obvious within a given number of iteration steps. The amount of computation is an important factor, and a slower rate of convergence is also regarded as non-convergence.



(a) iteration from 1 to 100



(b) iteration from 1 to 20

Fig. 1: residual error of three algorithms

Convergence of fixed quasi-Newton method depends much more than other two on the size of the matrix n. Figure 2 shows that for MEP of smaller size, when $n = 5$, the convergence performance of fixed quasi-Newton method will be much better, and all these algorithms can converge within 15 steps.

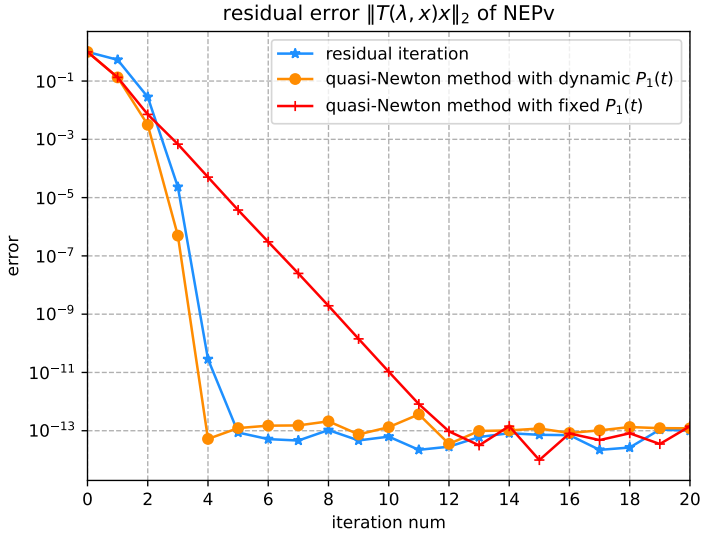


Fig. 2: residual for MEP of small scale

As shown in Table 1, we give a rough comparison about the running time of the algorithms. We use these algorithms to solve NEPv of different scale N, repeat five times except for warm-up, take the average, and get these approximate results.

N	residual iteration	dynamic quasi-Newton	fixed quasi-Newton
$N = 16$	0.0666826	0.1628306	0.152597
$N = 32$	0.0983220	0.2420766	0.2252568
$N = 64$	1.1664036	0.48100670	0.5768486
$N = 128$	8.1515176	6.4828370	7.2820142
$N = 256$	88.0154828	26.9589818	13.7634672

Table 1

Quasi-Newton method with dynamic or fixed $P_i(t)$ uses less time. It remains further research about how to reduce the computation cost of fixed quasi-Newton iteration and make it quicker than dynamic quasi-Newton iteration. It is worth noting that coding implementation is as important as the design of the algorithm for the performance, while the cost of further optimizing the algorithm, such as, improve the BLAS library or compilation optimization, is too high.

Another improvement about the algorithm is updating λ and μ_i's by solving Rayleigh quotient equation in our quasi-Newton algorithm. It is shown that Rayleigh

quotient update applies well with residual iteration [5], and we will consider updating eigenvalues of MEP equation 1.4 by the following equation,

$$
\begin{aligned}
x^{\mathrm{H}}A_{1,0}x &= \lambda_1 x^{\mathrm{H}}A_{1,1}x + \lambda_2 x^{\mathrm{H}}A_{1,2}x + \cdots + \lambda_{m+1}x^{\mathrm{H}}A_{1,m+1}x,\\
x^{\mathrm{H}}A_{2,0}x &= \lambda_1 x^{\mathrm{H}}A_{2,1}x + \lambda_2 x^{\mathrm{H}}A_{2,2}x + \cdots + \lambda_{m+1}x^{\mathrm{H}}A_{2,m+1}x,\\
&\vdots\\
x^{\mathrm{H}}A_{m+1,0}x &= \lambda_1 x^{\mathrm{H}}A_{m+1,1}x + \lambda_2 x^{\mathrm{H}}A_{m+1,2}x + \cdots + \lambda_{m+1}x^{\mathrm{H}}A_{m+1,m+1}x,
\end{aligned}
\tag{3.5}
$$

and stop when the inverse matrix of linear system is in ill condition.

Figure 3 shows an example of how Rayleigh qoutient reduces the number of steps required for convergence. Under different random conditions, the acceleration effect of the algorithm may vary greatly, even the convergence rate may slow down, both for dynamic and fixed $P_1(t)$.

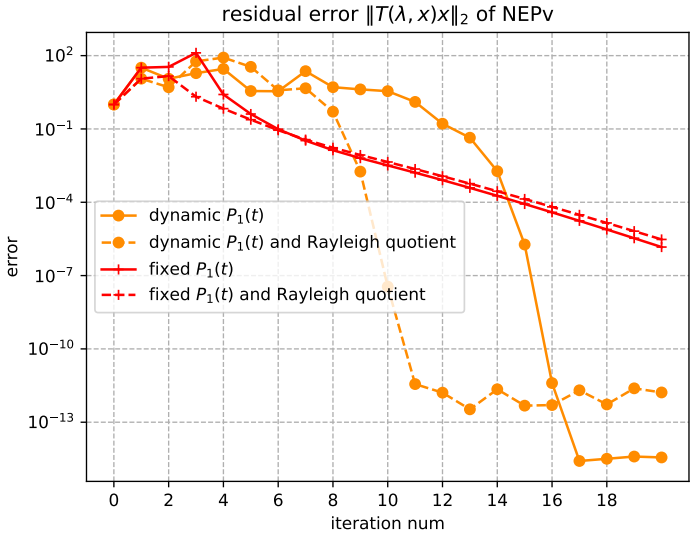


Fig. 3: quasi-Newton iteration acceleration by Rayleigh quotient

4. Conclusion. Our structure-preserved quasi-Newton method can solve NEPv at low computation cost and fast convergence rate when using dynmaic $P_1(t)$, and we can reduce the cost more when applying fixed $P_1(t)$. At the same time we can use Rayleigh qoutient to accelerate the convergence. The further research may depends on other aspects of NEPv, for example, the stepwise selection of g_i, modification of $c_i(x)$, and so on. What's more, a thorough discussion of the implementation of the numerical methods falls outside the scope of this paper, but it is an interesting point for future research.

REFERENCES

[1] F. V. Atkinson and A. Mingarelli, *Multiparameter eigenvalue problems*, vol. 1, Academic Press New York, 1972.

[2] T. Bühler and M. Hein, *Spectral clustering based on the graph p-laplacian*, in Proceedings of the 26th annual international conference on machine learning, 2009, pp. 81–88.
[3] R. Claes, E. Jarlebring, K. Meerbergen, and P. Upadhyaya, *Linearizable eigenvector nonlinearities*, SIAM Journal on Matrix Analysis and Applications, 43 (2022), pp. 764–786.
[4] D. S. Mackey, N. Mackey, C. Mehl, and V. Mehrmann, *Vector spaces of linearizations for matrix polynomials*, SIAM Journal on Matrix Analysis and Applications, 28 (2006), pp. 971–1004.
[5] B. Plestenjak, *A continuation method for a right definite two-parameter eigenvalue problem*, SIAM Journal on Matrix Analysis and Applications, 21 (2000), pp. 1163–1184.
[6] B. Plestenjak, *Numerical methods for nonlinear two-parameter eigenvalue problems*, BIT Numerical Mathematics, 56 (2016), pp. 241–262.
[7] A. Ruhe, *Algorithms for the nonlinear eigenvalue problem*, SIAM Journal on Numerical Analysis, 10 (1973), pp. 674–689.
[8] A. Szabo and N. S. Ostlund, *Modern quantum chemistry: introduction to advanced electronic structure theory*, Courier Corporation, 2012.
[9] F. Tudisco and D. J. Higham, *A nonlinear spectral method for core–periphery detection in networks*, SIAM Journal on Mathematics of Data Science, 1 (2019), pp. 269–292.