基于加权矩阵的多维广义特征值并行分解算法

高迎彬 [1, 2]　徐中英 [1]

摘　要　针对串行广义特征值分解算法实时性差的缺点, 提出基于加权矩阵的多维广义特征值分解算法. 与串行算法不同, 所提算法能够在一次迭代过程中并行地估计出多维广义特征向量. 平稳点分析表明: 当且仅当算法中状态矩阵等于所需的广义特征向量时, 算法达到收敛状态. 通过对比相邻时刻的状态矩阵模值证明了所提算法的自稳定特性. 所提算法参数选取简单, 实际实施较为容易. 数值仿真和实例应用进一步验证了算法的并行性、自稳定性和实用性.

关键词　广义特征值分解, 加权矩阵, 并行分解, 多维估计

引用格式　高迎彬, 徐中英. 基于加权矩阵的多维广义特征值并行分解算法. 自动化学报, 2023, **49**(12): 2639−2644

DOI　10.16383/j.aas.c200399

Multiple Generalized Eigenvalue Decomposition Algorithm in Parallel Based on Weighted Matrix

GAO Ying-Bin[1, 2]　XU Zhong-Ying[1]

Abstract　In order to overcome the disadvantages of sequential algorithms, such as poor real time, a multiple generalized eigenvalue decomposition algorithm is proposed based on weighted matrix method. Unlike sequential algorithms, the proposed algorithm is able to estimate multiple generalized eigenvectors in parallel only through one iteration procedure. The stationary point analysis shows that the algorithm reaches convergence state if and only if the state matrix is equal to the desired generalized eigenvectors. The self-stabilization characteristics of the proposed algorithm is proved by comparing the state matrix module values of adjacent moments. The proposed algorithm parameters are simple to select and easy to implement in practice. Numerical simulation and example application further verify the parallelism, self-stability and practicality of the algorithm.

Key words　Generalized eigenvalue decomposition, weighted matrix, parallel decomposition, multiple estimation

Citation　Gao Ying-Bin, Xu Zhong-Ying. Multiple generalized eigenvalue decomposition algorithm in parallel based on weighted matrix. *Acta Automatica Sinica*, 2023, **49**(12): 2639−2644

广义特征值分解是现代信号处理领域内重要的分析工具, 已经广泛应用于多个领域[1−6]. 在工程实际中, 不同的信号处理问题需要广义特征值分解的维数是不一样的. 根据提取广义特征向量的维数, 广义特征值分解可以分为 3 类[7]: 1) 仅能估计出矩阵束最大 (小) 广义特征值对应的广义特征向量的单维分解算法[8]; 2) 能够估计出若干个广义特征向量张成空间的子空间跟踪算法[9]; 3) 能够准确估计任意个广义特征向量的多维分解算法[10].

假定存在矩阵束 $(\boldsymbol{R}_y, \boldsymbol{R}_x)$, 其中, $\boldsymbol{R}_y$ 和 $\boldsymbol{R}_x$ 分别是两个 $n \times n$ 维对称正定矩阵, 则广义特征值分解就是要计算出一个 $n \times 1$ 维向量 $\boldsymbol{w}$ 和一个标量 λ, 使得

$$\boldsymbol{R}_y \boldsymbol{w} = \lambda \boldsymbol{R}_x \boldsymbol{w} \tag{1}$$

满足上述方程的向量 $\boldsymbol{w}$ 和标量 λ 分别记为矩阵束 $(\boldsymbol{R}_y, \boldsymbol{R}_x)$ 的广义特征向量和广义特征值. 基于特征值分解的算法是进行广义特征值分解的传统方法, 而基于神经网络的算法是近些年来涌现出的一类新型算法. 相比传统算法, 神经网络类算法具有计算量低、实时性强、能够处理非平稳信号等优点[11], 已经成为广义特征值分解领域内一个主流的研究方向.

基于神经网络, 学者们提出了很多优秀的广义特征值分解算法. 例如 RNNM (Recurrent neural network model) 算法[12]、R-GEVE (Reduced-rank generalized eigen-decomposition extraction) 算法[13]、基于 NOCS (Nested orthogonal complement structure) 架构的算法[9]、ANQ (Adaptive normalized quasi-Newton) 算法[14]、GChen (Generalized Chen algorithm) 算法[8]、GDM (Generalized Douglas minor component analysis) 算法[10] 等. 在上述算法中, RNNM 算法、R-GEVE 算法、ANQ 算法和 GCHen 算法是单维广义特征值分解算法; NOCS 算法属于子空间跟踪算法; 只有 GDM 算法是多维分解算法. 由于广义特征向量可以看作子空间的一组特殊基, 因此多维分解算法也适用于子空间算法的领域; 同时当多维分解算法提取维数为 1 时, 多维分解算法退化为单维分解算法. 综上可知, 多维分解算法具有最广泛的应用范围[10], 即研究多维分解算法更具有普适性.

在多维广义特征值分解算法领域主要存在两类算法: 串行算法和并行算法. 串行算法中多维广义特征向量是依次串行获取的, 即先用单维分解算法计算出最小广义特征值对应的广义特征向量, 然后利用膨胀技术再计算出次小广义特征值对应的广义特征向量, 依次类推[10]. 文献 [8] 提出了一种压缩技术, 文献 [10] 详细分析了该压缩技术并对其进行了改进, 实现了多维广义特征向量的串行提取. 相比并行算法, 串行算法有如下两点不足: 1) 由于串行算法中广义特征向量是串行依次提取, 整个提取过程需要时间较长, 因此串行算法不适合实时性要求较高的场合; 2) 串行算法在广义特征向量的估计过程中会用到上一次的估计结果, 而前一次的估计误差会累积到本次估计过程中, 因此会造成估计误差的累积传播[11]. 然而, 目前多维广义特征值分解算法还不多见, 因此发展并行多维算法仍是非常有意义的工作.

基于加权矩阵的思想, 本文提出一个多维广义特征值并行分解算法, 并详细分析了算法的收敛性和稳定性. 相比串行算法, 所提算法可以实现多维广义特征向量的并行计算, 实时性较强. 本文结构安排如下: 第 1 节提出新型广义特征值分解算法并对其进行收敛性和稳定性分析; 第 2 节是仿真实验和实例验证; 第 3 节是全文的总结.

收稿日期 2020-06-10　录用日期 2020-11-18

Manuscript received June 10, 2020; accepted November 18, 2020

国家自然科学基金 (62106242, 62273354) 资助

Supported by National Natural Science Foundation of China (6210 6242, 62273354)

本文责任编委 曾志刚

Recommended by Associate Editor ZENG Zhi-Gang

1. 火箭军工程大学导弹工程学院 西安 710025　2. 中国电子科技集团公司第五十四研究所 石家庄 050081

1. College of Missile Engineering, the Rocket Force University of Engineering, Xi′an 710025　2. The 54th Research Institute, China Electronics Technology Group Corporation, Shijiazhuang 050081

1 新型广义特征值分解算法

1.1 算法的提出

基于神经网络模型, Oja 算法[15] 首次提出了下述特征值分解算法:

$$\boldsymbol{w}(k+1)=\boldsymbol{w}(k)+\eta\left[\boldsymbol{R}\boldsymbol{w}(k)-\boldsymbol{w}^{\mathrm{T}}(k)\boldsymbol{R}\boldsymbol{w}(k)\boldsymbol{w}(k)\right] \tag{2}$$

其中, $\boldsymbol{R}\in\mathbf{R}^{n\times n}$ 是信号的自相关矩阵, $\boldsymbol{w}\in\mathbf{R}^{n\times 1}$ 是神经网络的状态向量, η 是算法的学习因子, n 为输入信号的维数. 然而该算法只能估计单个矩阵 $\boldsymbol{R}$ 的特征向量, 并非矩阵束的广义特征向量; 此外该算法只能实现单维分解, 不能进行多维分解. Oja 算法是特征值分解算法领域最为经典的算法, 很多现有算法在基础理论方面和 Oja 算法是相同的, 因此如果以 Oja 算法为代表进行研究并成功将其扩展为多维广义特征向量估计算法, 该研究结果对其他算法势必具有很强的借鉴意义.

假定 $\boldsymbol{R}_y$, $\boldsymbol{R}_x\in\mathbf{R}^{n\times n}$ 是两个信号的自相关矩阵, 其广义特征值和广义特征向量分别为 λ_i 和 $\boldsymbol{v}_i$, 其中, $i=1, 2, \cdots, n$. 为后续方便使用, 这里将矩阵束 $(\boldsymbol{R}_y, \boldsymbol{R}_x)$ 的 n 个正广义特征值进行降序排列, 即

$$\lambda_1>\lambda_2>\cdots>\lambda_n>0 \tag{3}$$

其对应的 n 个关于 $\boldsymbol{R}_x$ 正交的广义特征向量 $\boldsymbol{v}_i$ $(i=1, 2, \cdots, n)$ 也相应进行排列. 根据矩阵理论[16] 有

$$\boldsymbol{R}_y\boldsymbol{v}_i=\lambda\boldsymbol{R}_x\boldsymbol{v}_i \tag{4}$$

$$\boldsymbol{v}_j^{\mathrm{T}}\boldsymbol{R}_y\boldsymbol{v}_i=\delta_{ij},\quad i, j\in\{1, 2, \cdots, n\} \tag{5}$$

其中, δ_{ij} 是 Kronecker δ 函数.

为了计算矩阵束 $(\boldsymbol{R}_y, \boldsymbol{R}_x)$ 的多维广义特征向量, 通过对 Oja 算法添加加权矩阵, 提出如下的多维分解算法:

$$\boldsymbol{W}(k+1)=\boldsymbol{W}(k)+\eta[{\boldsymbol{R}_x}^{-1}\boldsymbol{R}_y\boldsymbol{W}(k)\boldsymbol{D}-\boldsymbol{W}(k)\boldsymbol{W}^{\mathrm{T}}(k)\boldsymbol{R}_y\boldsymbol{W}(k)] \tag{6}$$

其中, $\boldsymbol{W}\in\mathbf{R}^{n\times r}$ 是神经网络的状态矩阵; $\boldsymbol{D}=\mathrm{diag}\{d_1, d_2, \cdots, d_r\}$ 是一个对角矩阵, 其对角线元素满足 $d_1>d_2>\cdots>d_r>0$, 这里将其称为加权矩阵; r 是需要计算的广义特征向量的维数. 通过式 (6) 给定状态矩阵更新规则, 状态矩阵 $\boldsymbol{W}$ 进行更新迭代, 最终将收敛到矩阵束 $(\boldsymbol{R}_y, \boldsymbol{R}_x)$ 最大的 r 个广义特征值对应的广义特征向量.

通过对式 (6) 进行分析可以发现, 如果令式 (6) 中 $\boldsymbol{R}_x=\boldsymbol{I}_n$ 和 $\boldsymbol{D}=\boldsymbol{I}_r$, 且进行单维分解 (即 $r=1$), 式 (6) 就会退化为 Oja 算法. 因此可以说式 (6) 是 Oja 算法的扩展. 式 (2) 是单维特征向量估计, 如果利用文献 [10] 中的压缩技术将其扩展为串行算法, 则其计算复杂度为 $\mathrm{O}(n^3r/3+4n^2r+2nr)$, 而式 (6) 是多维广义特征向量估计, 其计算复杂度为 $\mathrm{O}(n^3/3+2n^2r+2nr^2)$, 显然要低于串行算法的计算复杂度. 由于加权矩阵的加入, 使得状态矩阵在收敛过程中对广义特征向量与其构成子空间的夹角更为敏感[17], 进而通过在迭代过程中对神经网络状态矩阵施加的隐形 Gram-Schmidt 正交化 (GS orthogonalization, GSO) 操作, 使得状态矩阵能够收敛到广义特征向量, 而非其构成的子空间. 加权矩阵仅需满足约束条件 $d_1>d_2>\cdots>d_r>0$, 在实际使用中该约束条件很容易达到 (如可以取一等差数列), 因此, 式 (6) 是符合实际使用需要的.

1.2 算法的收敛性分析

算法是否能够准确地计算出矩阵束 $(\boldsymbol{R}_y, \boldsymbol{R}_x)$ 的广义特征向量, 是算法发展过程中首先要解决的问题, 本节将通过定理 1 证明算法的收敛性.

定理 1. 当且仅当神经网络的状态矩阵 $\boldsymbol{W}=\boldsymbol{U}\boldsymbol{D}^{\frac{1}{2}}$ 时, 所提算法达到稳定的收敛状态, 其中, $\boldsymbol{U}=[\boldsymbol{v}_1, \boldsymbol{v}_2, \cdots, \boldsymbol{v}_r]$ 是由矩阵束 $(\boldsymbol{R}_y, \boldsymbol{R}_x)$ 最大的 r 个广义特征值对应的广义特征向量构成的矩阵.

证明. 定义矩阵函数

$$J(\boldsymbol{W})=\boldsymbol{R}_x^{-1}\boldsymbol{R}_y\boldsymbol{W}\boldsymbol{D}-\boldsymbol{W}\boldsymbol{W}^{\mathrm{T}}\boldsymbol{R}_y\boldsymbol{W} \tag{7}$$

则 $J(\boldsymbol{W})$ 代表着算法的学习步长. 当矩阵取值为 $\boldsymbol{W}=\boldsymbol{U}\boldsymbol{D}^{\frac{1}{2}}$ 时

$$J(\boldsymbol{W})=\boldsymbol{R}_x^{-1}\boldsymbol{R}_y\boldsymbol{U}\boldsymbol{D}^{\frac{1}{2}}\boldsymbol{D}-\boldsymbol{U}\boldsymbol{D}^{\frac{1}{2}}\boldsymbol{D}^{\frac{1}{2}}\boldsymbol{U}^{\mathrm{T}}\boldsymbol{R}_y\boldsymbol{U}\boldsymbol{D}^{\frac{1}{2}}=\boldsymbol{U}\boldsymbol{\Lambda}_r\boldsymbol{D}^{\frac{3}{2}}-\boldsymbol{U}\boldsymbol{\Lambda}_r\boldsymbol{D}^{\frac{3}{2}}=\boldsymbol{0} \tag{8}$$

其中, $\boldsymbol{\Lambda}_r=\mathrm{diag}\{\lambda_1, \lambda_2, \cdots, \lambda_r\}$ 是一个对角矩阵, 其对角线元素由矩阵束 $(\boldsymbol{R}_y, \boldsymbol{R}_x)$ 最大的 r 个广义特征值构成.

当且仅当学习步长等于零时算法达到收敛状态, 即 $J(\boldsymbol{W})=\boldsymbol{0}$, 进而有

$${\boldsymbol{R}_x}^{-1}\boldsymbol{R}_y\boldsymbol{W}\boldsymbol{D}=\boldsymbol{W}\boldsymbol{W}^{\mathrm{T}}\boldsymbol{R}_y\boldsymbol{W} \tag{9}$$

对式 (9) 同时左乘 $\boldsymbol{W}^{\mathrm{T}}\boldsymbol{R}_x$, 并经过适当简化, 有

$$\boldsymbol{W}^{\mathrm{T}}\boldsymbol{R}_y\boldsymbol{W}\left(\boldsymbol{D}-\boldsymbol{W}^{\mathrm{T}}\boldsymbol{R}_x\boldsymbol{W}\right)=\boldsymbol{0} \tag{10}$$

由状态矩阵 $\boldsymbol{W}$ 的任意性, 可得

$$\boldsymbol{W}^{\mathrm{T}}\boldsymbol{R}_x\boldsymbol{W}=\boldsymbol{D} \tag{11}$$

令 $\boldsymbol{P}=\boldsymbol{W}\boldsymbol{D}^{\frac{1}{2}}$ 并将其代入式 (11), 有: $\boldsymbol{P}^{\mathrm{T}}\boldsymbol{R}_x\boldsymbol{P}=\boldsymbol{I}$, 即矩阵 $\boldsymbol{P}$ 的列向量是关于矩阵 $\boldsymbol{R}_x$ 正交的. 矩阵 $\boldsymbol{W}^{\mathrm{T}}\boldsymbol{R}_y\boldsymbol{W}$ 的特征值分解为

$$\boldsymbol{W}^{\mathrm{T}}\boldsymbol{R}_y\boldsymbol{W}=\boldsymbol{Q}^{\mathrm{T}}\boldsymbol{\Sigma}\boldsymbol{Q} \tag{12}$$

其中, $\boldsymbol{Q}\in\mathbf{R}^{r\times r}$ 是一个正交矩阵, $\boldsymbol{\Sigma}\in\mathbf{R}^{r\times r}$ 是一个对角矩阵. 将式 (12) 代入式 (9) 并进行适当化简, 有

$$\boldsymbol{R}_y\boldsymbol{U}_r=\boldsymbol{R}_x\boldsymbol{U}_r\boldsymbol{\Sigma} \tag{13}$$

其中, $\boldsymbol{U}_r=\boldsymbol{P}\boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{Q}^{\mathrm{T}}=\boldsymbol{W}\boldsymbol{Q}^{\mathrm{T}}$. 由于 $\boldsymbol{R}_x$ 是对称正定矩阵, 则必然可以分解为 $\boldsymbol{R}_x=(\boldsymbol{C}^{-1})^{\mathrm{T}}\boldsymbol{C}^{-1}$, 其中 $\boldsymbol{C}$ 是一个可逆矩阵. 根据 $\left(\boldsymbol{C}^{\mathrm{T}}\boldsymbol{R}_y\boldsymbol{C}\right)\boldsymbol{C}^{-1}\boldsymbol{U}_r=\boldsymbol{C}^{-1}\boldsymbol{U}_r\boldsymbol{\Sigma}$ 和矩阵 $\boldsymbol{C}^{-1}\boldsymbol{U}_r$ 列满秩有: $\boldsymbol{C}^{-1}\boldsymbol{U}_r$ 必然是由 $\boldsymbol{C}^{\mathrm{T}}\boldsymbol{R}_y\boldsymbol{C}$ 的特征向量构成, 即 $\boldsymbol{U}_r$ 是由矩阵束 $(\boldsymbol{R}_y, \boldsymbol{R}_x)$ 最大的 r 个广义特征向量构成的矩阵. □

1.3 自稳定特性分析

自稳定特性是指不论初始化状态矩阵如何选择, 状态矩阵模值最终能够收敛到一个固定的常值[18]. 由于自稳定特性可以保证状态矩阵模值的稳定性, 因此已经成为算法发展中一个重要的研究内容. 接下来将对所提算法进行自稳定特性分析, 为后续方便使用, 首先定义如下矩阵范数.

定义 1. 假定 $\boldsymbol{R}\in\mathbf{R}^{n\times n}$ 是一个对称正定矩阵, 则令 $\sqrt{\mathrm{tr}(\boldsymbol{W}^{\mathrm{T}}\boldsymbol{R}\boldsymbol{W})}$ 为矩阵 $\boldsymbol{W}\in\mathbf{R}^{n\times r}$ 关于矩阵 $\boldsymbol{R}$ 的范数, 记为 $\|\boldsymbol{W}\|_{\boldsymbol{R}}=\sqrt{\mathrm{tr}(\boldsymbol{W}^{\mathrm{T}}\boldsymbol{R}\boldsymbol{W})}$.

容易证明, 上述定义符合矩阵范数的规定. 基于定义

1, 定理 2 给出所提算法的自稳定特性分析.

定理 2. 当输入信号是有界的且学习因子 η 足够小时, 所提算法 (6) 中状态矩阵 $\boldsymbol{W}$ 模值是自稳定的, 且状态矩阵模值 $\|\boldsymbol{W}\|_{\boldsymbol{R}_x}$ 的收敛值等于 $\mathrm{tr}(\boldsymbol{D})$.

证明. 假定当前时刻状态矩阵模值为 $\|\boldsymbol{W}(k)\|_{\boldsymbol{R}_x} = \sqrt{\mathrm{tr}\left[\boldsymbol{W}^{\mathrm{T}}(k)\boldsymbol{R}_x\boldsymbol{W}(k)\right]}$, 则下一时刻的状态矩阵模值为

$$\begin{aligned}
&\|\boldsymbol{W}(k+1)\|^2_{\boldsymbol{R}_x} = \mathrm{tr}\left[\boldsymbol{W}^{\mathrm{T}}(k+1)\boldsymbol{R}_x\boldsymbol{W}(k+1)\right] = \\
&\mathrm{tr}\{[\boldsymbol{W}(k)+\eta\boldsymbol{R}_x{}^{-1}\boldsymbol{R}_y\boldsymbol{W}(k)\boldsymbol{D} - \\
&\eta\boldsymbol{W}(k)\boldsymbol{W}^{\mathrm{T}}(k)\boldsymbol{R}_y\boldsymbol{W}(k)]^{\mathrm{T}}\boldsymbol{R}_x[\boldsymbol{W}(k)+ \\
&\eta\boldsymbol{R}_x{}^{-1}\boldsymbol{R}_y\boldsymbol{W}(k)\boldsymbol{D} - \eta\boldsymbol{W}(k)\boldsymbol{W}^{\mathrm{T}}(k)\boldsymbol{R}_y\boldsymbol{W}(k)]\} = \\
&\mathrm{tr}\{\boldsymbol{W}^{\mathrm{T}}(k)\boldsymbol{R}_x\boldsymbol{W}(k) - 2\eta\boldsymbol{W}^{\mathrm{T}}(k)\boldsymbol{R}_y\boldsymbol{W}(k)\times \\
&\boldsymbol{W}(k)\boldsymbol{R}_x\boldsymbol{W}(k) + 2\eta\boldsymbol{D}\boldsymbol{W}^{\mathrm{T}}(k)\boldsymbol{R}_y\boldsymbol{W}(k) + \\
&\eta^2\boldsymbol{D}\boldsymbol{W}^{\mathrm{T}}(k)\boldsymbol{R}_y\boldsymbol{R}_x{}^{-1}\boldsymbol{R}_y\boldsymbol{W}(k)\boldsymbol{D} + \eta^2\boldsymbol{W}^{\mathrm{T}}(k)\times \\
&\boldsymbol{R}_y\boldsymbol{W}(k)\boldsymbol{W}(k)\boldsymbol{R}_x\boldsymbol{W}(k)\boldsymbol{W}^{\mathrm{T}}(k)\boldsymbol{R}_y\boldsymbol{W}(k)\} \approx \\
&\|\boldsymbol{W}(k)\|^2_{\boldsymbol{R}_x} - 2\eta\|\boldsymbol{W}(k)\|^2_{\boldsymbol{R}_y}\left(\|\boldsymbol{W}(k)\|^2_{\boldsymbol{R}_x} - \mathrm{tr}(\boldsymbol{D})\right)
\end{aligned} \tag{14}$$

由于 η 足够小, 因此省去式 (14) 中 η 的二阶项. 对比相邻时刻状态矩阵模值, 有

$$\begin{aligned}
&\frac{\|\boldsymbol{W}(k+1)\|^2_{\boldsymbol{R}_x}}{\|\boldsymbol{W}(k)\|^2_{\boldsymbol{R}_x}} = \\
&\frac{\|\boldsymbol{W}(k)\|^2_{\boldsymbol{R}_x} - 2\eta\|\boldsymbol{W}(k)\|^2_{\boldsymbol{R}_y}\left(\|\boldsymbol{W}(k)\|^2_{\boldsymbol{R}_x} - \mathrm{tr}(\boldsymbol{D})\right)}{\|\boldsymbol{W}(k)\|^2_{\boldsymbol{R}_x}} = \\
&1 - 2\eta\frac{\|\boldsymbol{W}(k)\|^2_{\boldsymbol{R}_y}}{\|\boldsymbol{W}(k)\|^2_{\boldsymbol{R}_x}}\left(\|\boldsymbol{W}(k)\|^2_{\boldsymbol{R}_x} - \mathrm{tr}(\boldsymbol{D})\right)
\end{aligned} \tag{15}$$

由式 (15) 可得

$$\frac{\|\boldsymbol{W}(k+1)\|^2_{\boldsymbol{R}_x}}{\|\boldsymbol{W}(k)\|^2_{\boldsymbol{R}_x}}\begin{cases} >1, & \|\boldsymbol{W}(k)\|^2_{\boldsymbol{R}_x} < \mathrm{tr}(\boldsymbol{D}) \\ =1, & \|\boldsymbol{W}(k)\|^2_{\boldsymbol{R}_x} = \mathrm{tr}(\boldsymbol{D}) \\ <1, & \|\boldsymbol{W}(k)\|^2_{\boldsymbol{R}_x} > \mathrm{tr}(\boldsymbol{D}) \end{cases}$$

从式 (15) 可知, 如果当前时刻 $\|\boldsymbol{W}(k)\|^2_{\boldsymbol{R}_x} < \mathrm{tr}(\boldsymbol{D})$, 则在下一时刻状态矩阵模值将增加; 反之, 如果 $\|\boldsymbol{W}(k)\|^2_{\boldsymbol{R}_x} > \mathrm{tr}(\boldsymbol{D})$, 则下一时刻状态矩阵模值减小, 即状态矩阵模值最终的收敛结果为 $\|\boldsymbol{W}(k)\|^2_{\boldsymbol{R}_x} = \mathrm{tr}(\boldsymbol{D})$. □

2　仿真实验和实例应用

本节将通过仿真实验和实例应用来验证所提算法的性能. 实验 1 考察算法多维广义特征向量的估计能力; 实验 2 考察算法的自稳定性; 实验 3 考察算法在信号处理中的实用性. 在仿真实验开始前, 首先随机生成两个对称正定矩阵, 该矩阵束 ($\boldsymbol{R}_y$, $\boldsymbol{R}_x$) 最大的 3 个广义特征值对应的广义特征向量分别如式 (16) ~ (18) 所示 (见本页下方).

2.1　算法能力验证

实验 1 分别利用所提算法和 GDM 算法[10] 对矩阵束 ($\boldsymbol{R}_y$, $\boldsymbol{R}_x$) 最大的 3 个广义特征值对应的广义特征向量进行估计, 即计算 $\boldsymbol{v}_1$, $\boldsymbol{v}_2$, $\boldsymbol{v}_3$. 在算法的迭代过程中计算两个指标, 第 1 个指标是状态矩阵列向量与 $\boldsymbol{v}_1$, $\boldsymbol{v}_2$, $\boldsymbol{v}_3$ 之间的方向余弦 (Directional cosine, DC), 即

$$DC_i(k) = \frac{\left|\boldsymbol{v}_i^{\mathrm{T}}\boldsymbol{W}_i(k)\right|}{\|\boldsymbol{v}_i\|\,\|\boldsymbol{W}_i(k)\|} \tag{19}$$

其中, $i = 1, 2, 3$, $\boldsymbol{W}_i(k)$ 表示第 k 次迭代时状态矩阵 $\boldsymbol{W}(k)$ 的第 i 列向量.

第 2 个指标是状态矩阵列向量之间关于矩阵 $\boldsymbol{R}_x$ 的范数, 即

$$Norm_{ij}(k) = \boldsymbol{W}_i^{\mathrm{T}}(k)\boldsymbol{R}_x\boldsymbol{W}_j(k), \quad i, j = 1, 2, 3 \tag{20}$$

两个算法的初始化状态矩阵是随机产生的, 学习因子 $\eta = 0.25$. GDM 算法中, 参数 $\tau = 3$; 所提算法中, 加权矩阵取 $\boldsymbol{D} = \mathrm{diag}\{3, 2, 1\}$, 满足 $d_1 > d_2 > d_3 > 0$ 的要求. 在仿真过程中, 为了更好地衡量算法性能, 这里进行了 100 次独立实验, 然后分别计算每次实验过程中数据的均值和上下界. 表 1 是两种算法完成 3 个广义特征向量所需的平均时间. 图 1 和图 2 分别是所提算法和 GDM 算法的方向余弦曲线, 其中实线是方向余弦的平均值, 虚线是数据上界, 点划线是数据下界. 为了清晰起见, 图 3 和图 4 分别只给出了所提算法的列向量模值曲线 (平均值) 和不同列向量之间内积关系曲线 (平均值).

$$\boldsymbol{R}_y = \begin{bmatrix} 0.7904 & -0.0920 & -0.0370 & 0.0697 & -0.0563 & -0.0082 \\ -0.0920 & 0.7087 & -0.0087 & -0.0555 & -0.1367 & 0.2456 \\ -0.0370 & -0.0087 & 0.5733 & -0.1282 & -0.2532 & -0.1875 \\ 0.0697 & -0.0555 & -0.1282 & 0.5665 & -0.2483 & 0.0344 \\ -0.0563 & -0.1367 & -0.2532 & -0.2483 & 0.5787 & 0.0201 \\ -0.0082 & 0.2456 & -0.1875 & 0.0344 & 0.0201 & 0.2733 \end{bmatrix} \tag{16}$$

$$\boldsymbol{R}_x = \begin{bmatrix} 0.5539 & -0.0273 & 0.0212 & 0.1184 & -0.1045 & 0.1350 \\ -0.0273 & 0.5815 & 0.0254 & -0.0481 & -0.1415 & -0.0006 \\ 0.0212 & 0.0254 & 0.7190 & -0.0575 & -0.0469 & 0.1114 \\ 0.1184 & -0.0481 & -0.0575 & 0.5436 & 0.0689 & 0.0281 \\ -0.1045 & -0.1415 & -0.0469 & 0.0689 & 0.4302 & 0.1308 \\ 0.1350 & -0.0006 & 0.1114 & 0.0281 & 0.1308 & 0.5010 \end{bmatrix} \tag{17}$$

$$\begin{cases} \boldsymbol{v}_1 = [-1.0839,\ -0.1148,\ -0.0482,\ 0.8737,\ -1.4241,\ 0.7618]^{\mathrm{T}} \\ \boldsymbol{v}_2 = [0.4091,\ -0.8696,\ 0.5280,\ 0.2559,\ -0.4955,\ -0.6944]^{\mathrm{T}} \\ \boldsymbol{v}_3 = [-0.9839,\ -0.4039,\ 0.3737,\ -0.3484,\ 0.2866,\ -0.2464]^{\mathrm{T}} \end{cases} \tag{18}$$

表 1 两种算法的计算时间

Table 1 The time cost of the two algorithms

算法	时间 (ms)
所提算法	2.16
GDM 算法	14.61

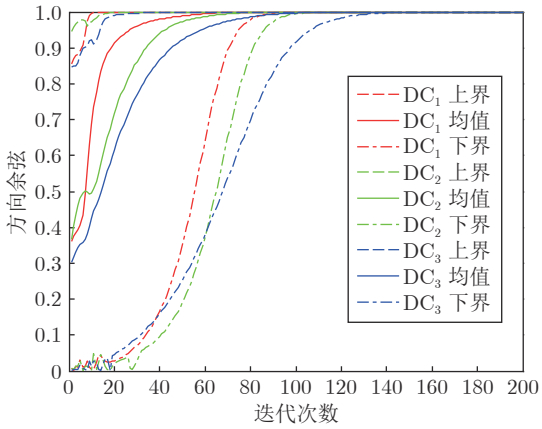

图 1 所提算法方向余弦曲线

Fig. 1 The DC curves of the proposed algorithm

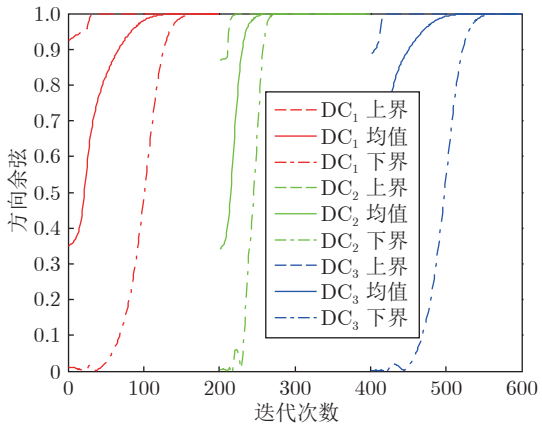

图 2 GDM 算法方向余弦曲线

Fig. 2 The DC curves of the GDM algorithm

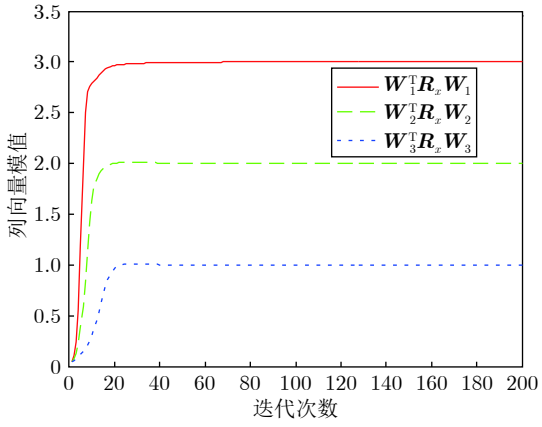

图 3 列向量模值曲线

Fig. 3 The norm curves of the column vectors

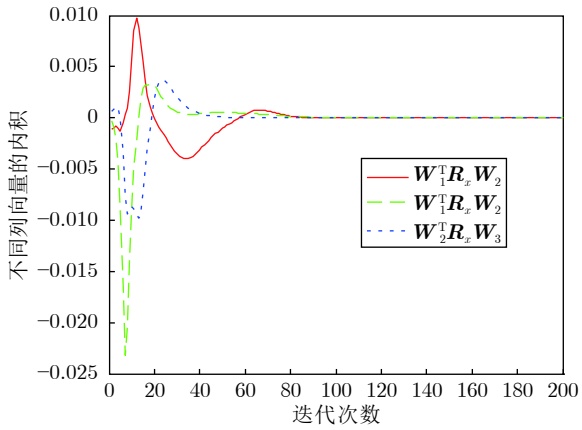

图 4 不同列向量内积关系曲线

Fig. 4 The inner product curves of different column vectors

从图 1 中可以看出, 所提算法在经历了最快 10 次, 最慢 140 次, 平均 90 次左右的迭代运算后, 3 条方向余弦曲线均能收敛到单位 1. 表明不论收敛快慢, 神经网络的 3 个列向量均能准确地收敛到 3 个广义特征向量 $\boldsymbol{v}_1$, $\boldsymbol{v}_2$, $\boldsymbol{v}_3$ 的方向, 且其对应的广义特征值也是按照降序排列的. 从图 3 中可以看出, 在方向余弦收敛时, 状态矩阵的各个列向量模值也都收敛到一个常值, 即算法具有很好的收敛性. 综合图 1 和图 3 可以得知, 所提算法能够有效地估计矩阵束的广义特征向量.

从图 2 中可以看出, GDM 算法是串行算法, 所有广义特征向量是一个接一个顺序估计的. 由于下次估计过程只有在上一次广义特征向量估计完成后才可以开始, 因此后两次迭代开始时刻分别是 200 和 400, 而并非 0. 显然这样会产生较长的等待时间, 且当估计广义特征向量维数增加时, 该等待时间会更长.

表 1 中, 本文所提算法的平均运行时间是 2.16 ms, 远小于 GDM 算法的 14.61 ms. 通过对比图 1 和图 2 以及综合表 1 可得, 由于所提算法仅需要一次迭代运算, 进而避免了等待时间, 因此计算时间较短. 需要注意的是, 由于串行算法需要信号采样完成后才可以进行运算, 而并行算法可以边采样边计算, 因此, 实际运行时并行算法在实时性方面会更有优势.

此外, 通过研究图 3 和图 4 中的收敛结果可以发现: 各个列向量模值的收敛值分别为 3, 2, 1, 刚好等于加权矩阵对角线上的元素; 而两个不同的列向量关于 $\boldsymbol{R}_x$ 的内积最终收敛到了零, 即状态矩阵中不同列向量是关于矩阵 $\boldsymbol{R}_x$ 正交的. 综合图 2 和图 3 可知, $\boldsymbol{W}^{\mathrm{T}}\boldsymbol{R}_x\boldsymbol{W} = \boldsymbol{D} = \mathrm{diag}\{3, 2, 1\}$, 显然该结果与定理 1 中的理论分析结果是一致的.

2.2 自稳定特性实验

实验 2 仍利用所提算法对矩阵束 $(\boldsymbol{R}_y, \boldsymbol{R}_x)$ 的广义特征向量进行计算. 为了验证算法的自稳定特性, 这里进行四次独立实验. 第 1 次取加权矩阵 $\boldsymbol{D}_1 = \mathrm{diag}\{10, 7, 5\}$, 则 $\mathrm{tr}(\boldsymbol{D}_1) = \sqrt{22}$; 第 2 次取加权矩阵 $\boldsymbol{D}_2 = \mathrm{diag}\{3, 2, 1\}$, 则 $\mathrm{tr}(\boldsymbol{D}_2) = \sqrt{6}$; 第 3 次取加权矩阵 $\boldsymbol{D}_3 = \mathrm{diag}\{1, 0.8, 0.2\}$, 则 $\mathrm{tr}(\boldsymbol{D}_3) = \sqrt{2}$; 第 4 次取加权矩阵 $\boldsymbol{D}_4 = \mathrm{diag}\{0.5, 0.3, 0.2\}$, 则 $\mathrm{tr}(\boldsymbol{D}_4) = 1$. 算法的初始化状态矩阵是随机产生的, 其模值 $\|\boldsymbol{W}(0)\|_{\boldsymbol{R}_x}$ 分别设置为大于、等于和小于 $\mathrm{tr}(\boldsymbol{D}_i)$ $(i = 1, 2, 3, 4)$ 等三种情况. 四次实验中学习因子均设置为 $\eta = 0.001$. 图 5 是四次实验中的状态矩阵模值曲线.

以图 5(a) 为例, 如果初始化状态矩阵 $\|\boldsymbol{W}(0)\|_{\boldsymbol{R}_x} <$

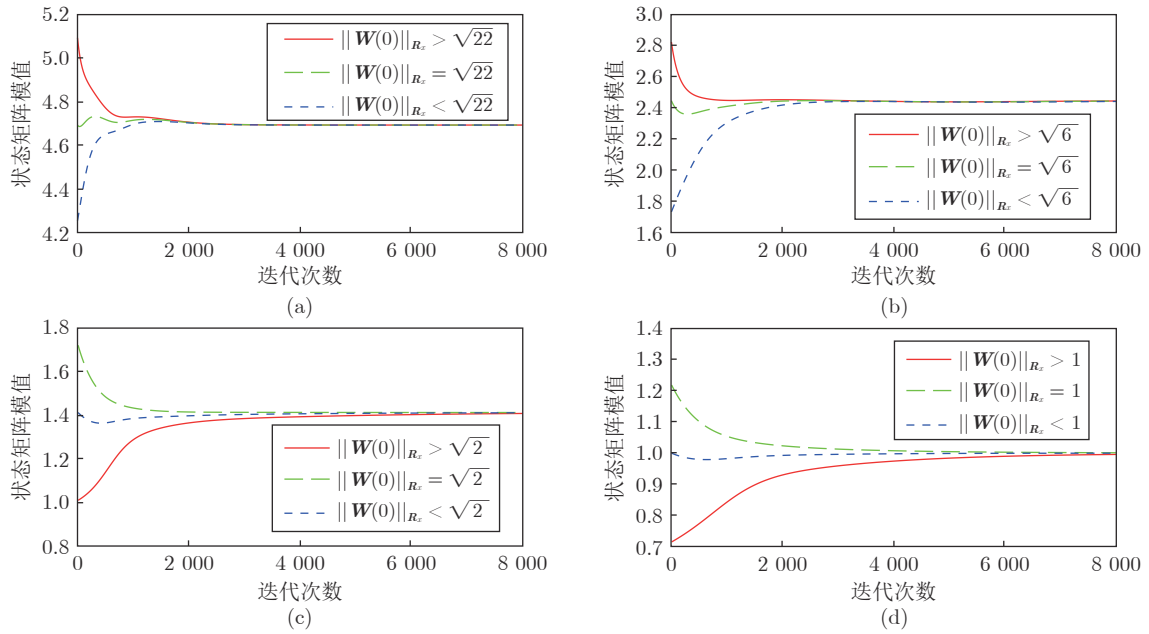

图 5　不同对角矩阵下状态矩阵模值曲线

Fig.5　The norm curves of the state matrix with different diagonal matrices

$\sqrt{22}$ 时, 状态矩阵模值会不断增加, 最终收敛到 $\sqrt{22}$; 反之, 当 $\|\boldsymbol{W}(0)\|_{\boldsymbol{R}_x} > \sqrt{22}$ 时, 模值就会减小, 最终收敛值也是 $\sqrt{22}$. 从图 5(b) ~ 5(d) 中可以看到相类似的状态矩阵模值变化曲线, 不同之处在于: 状态矩阵模值收敛结果分别为对应的加权矩阵的迹. 所有曲线的变化规律是与定理 2 的分析结果相吻合的, 这也进一步证明了所提算法具有自稳定特性.

2.3　盲分离实验

实验 3 将所提算法应用于解决盲分离问题, 其中四个源信号取自 ICALAB 工具箱中的 ABio7.mat 文件[19]. 图 6 是四个源信号的波形曲线.

源信号经过混合矩阵 $\boldsymbol{M}$, 并添加加性白噪声以后得到观测信号 $\boldsymbol{x}$. 盲分离问题就是找出合适的分离矩阵 $\boldsymbol{W}$, 使得分离信号 $\boldsymbol{s} = \boldsymbol{W}^{\mathrm{T}}\boldsymbol{x}$ 彼此相互独立. 文献 [20] 提出了一个基于广义特征值分解的盲分离算法, 而分离矩阵 $\boldsymbol{W}$ 是由矩阵束 $(\boldsymbol{R}_y, \boldsymbol{R}_x)$ 所有广义特征向量构成的, 其中 $\boldsymbol{R}_x = \mathrm{E}[\boldsymbol{x}\boldsymbol{x}^{\mathrm{T}}]$ 和 $\boldsymbol{R}_y = \mathrm{E}[\boldsymbol{y}\boldsymbol{y}^{\mathrm{T}}]$, $\boldsymbol{y}$ 是一个滤波器的输出. 接下来利用所提算法求解该分离矩阵, 算法的初始化参数设置与实验 1 相同, 加权矩阵取 $\boldsymbol{D} = \mathrm{diag}\{4, 3, 2, 1\}$, 混合矩阵为

$$\boldsymbol{M} = \begin{bmatrix} 0.547 & 0.276 & -1.071 & 0.615 \\ 0.911 & -0.327 & 0.512 & -1.158 \\ -0.273 & -0.418 & -0.571 & -0.595 \\ 0.757 & -1.623 & -2.361 & 0.408 \end{bmatrix} \tag{21}$$

图 7 和图 8 分别是混合后的信号和所提算法获得的分离信号. 从图 7 中可以看出, 经过混合矩阵后, 源信号与观测信号具有非常大的差别. 而对比图 8 和图 6 可以发现, 分离信号与源信号具有相似的波形. 通过计算, 四个分离信号与源信号之间的相关系数分别为 0.9999, 1.0000, 0.9989, 0.9665, 即分离信号与源信号相似程度非常高, 表明所提算法能够有效地解决盲分离问题.

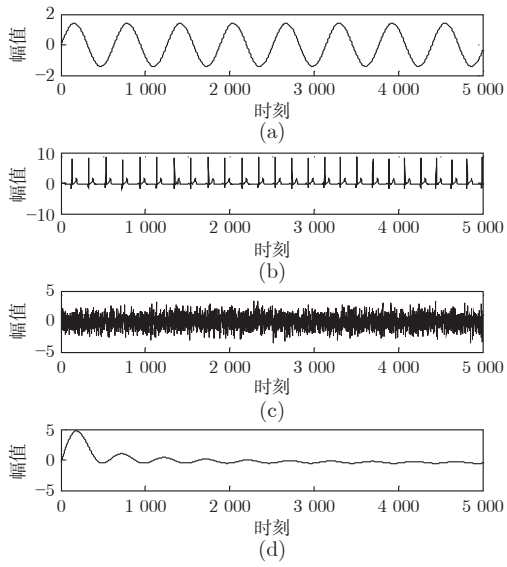

图 6　源信号波形

Fig.6　The waveform of source signals

3　结束语

多维广义特征值分解是应用范围最广的一类广义特征值分解算法. 针对串行算法的缺点, 本文利用加权矩阵法提出了一个并行广义特征值分解算法, 并分析了算法的收敛性和自稳定特性. 与以往算法相比, 所提算法可以在一次迭代过程中完成多维广义特征向量的同时估计, 因此具有很好的实时性和准确性. 仿真实验和应用实验表明所提算法能够有效地估计出所需的广义特征向量, 而且能够应用于解决盲分离问题. 后续研究将会考虑如何能够进一步降低算法的复杂度和加权矩阵法推广应用的问题, 这将有利于多维分解算法的发展.

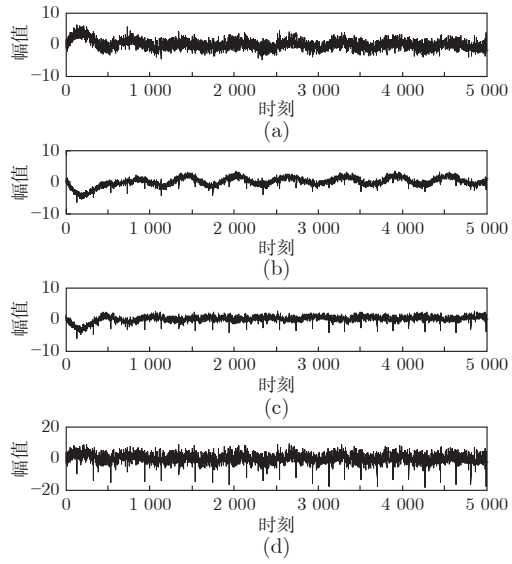

图 7 观测信号曲线

Fig. 7 The waveform of observed signals

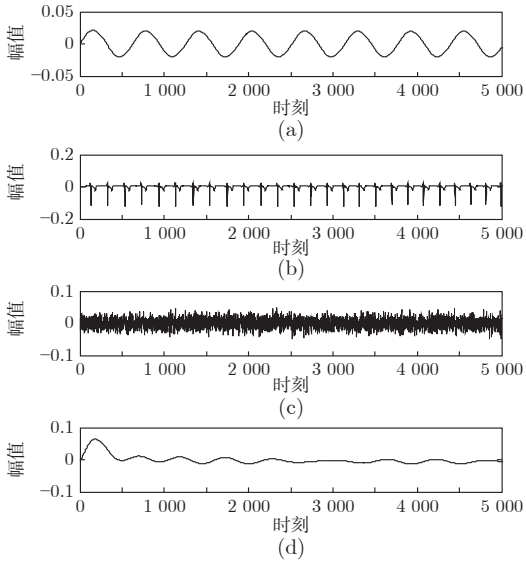

图 8 分离信号曲线

Fig. 8 The waveform of separated signals

References

1 Kong X Y, Du B Y, Feng X W, Luo J Y. Unified and self-stabilized parallel algorithm for multiple generalized eigenpairs extraction. *IEEE Transactions on Signal Processing*, 2020, **68**: 3644−3659

2 Rippl M, Lang B, Huckle T. Parallel eigenvalue computation for banded generalized eigenvalue problems. *Parallel Computing*, 2019, **88**(12): 102542.1−102542.9

3 Sun S L, Xie X J, Dong C. Multiview learning with generalized eigenvalue proximal support vector machines. *IEEE Transactions on Cybernetics*, 2019, **49**(2): 688−697

4 Miyata T. A Riccati-type algorithm for solving generalized Hermitian eigenvalue problems. *The Journal of Supercomputing*, 2021, **77**(2): 2091−2102

5 Guo Ya-Ning, Lin Wei, Pan Quan, Zhao Chun-Hui, Hu Jin-Wen, Ma Juan-Juan. Generalized manifold learning for high resolution remote sensing image object classification. *Acta Automatica Sinica*, 2019, **45**(4): 720−729
(郭亚宁, 林伟, 潘泉, 赵春晖, 胡劲文, 马娟娟. 基于推广流形学习的高分辨遥感影像目标分类. 自动化学报, 2019, **45**(4): 720−729)

6 Chen Xiao-Yun, Liao Meng-Zhen. Dimensionality reduction with extreme learning machine based on sparsity and neighborhood preserving. *Acta Automatica Sinica*, 2019, **45**(2): 325−333
(陈晓云, 廖梦真. 基于稀疏和近邻保持的极限学习机降维. 自动化学报, 2019, **45**(2): 325−333)

7 Kong Xiang-Yu, Feng Xiao-Wei, Hu Chang-Hua. *General Principal Component Analysis and Application*. Beijing: National Defense Industry Press, 2018.
(孔祥玉, 冯晓伟, 胡昌华. 广义主成分分析算法及应用. 北京: 国防工业出版社, 2018.)

8 Gao Y B, Kong X Y, Zhang Z X, Hou L A. An adaptive self-stabilizing algorithm for minor generalized eigenvector extraction and its convergence analysis. *IEEE Transactions on Neural Networks and Learning Systems*, 2018, **29**(10): 4869−4881

9 Nguyen T D, Takahashi N, Yamada I. An adaptive extraction of generalized eigensubspace by using exact nested orthogonal complement structure. *Multidimensional Systems and Signal Processing*, 2013, **24**(3): 457−483

10 Li H Z, Du B Y, Kong X Y, Gao Y B, Hu C H, Bian X H. A generalized minor component extraction algorithm and its analysis. *IEEE Access*, 2018, **6**: 36771−36779

11 Kong X Y, Hu C H, Duan Z S. *Principal Component Analysis Networks and Algorithms*. Singapore: Springer, 2017.

12 Liu L J, Shao H M, Nan D. Recurrent neural network model for computing largest and smallest generalized eigenvalue. *Neurocomputing*, 2008, **71**(16−18): 3589−3594

13 Attallah S, Abed-Meraim K. A fast adaptive algorithm for the generalized symmetric eigenvalue problem. *IEEE Signal Processing Letters*, 2008, **15**: 797−800

14 Nguyen T D, Yamada I. Adaptive normalized quasi-Newton algorithms for extraction of generalized eigen-pairs and their convergence analysis. *IEEE Transactions on Signal Processing*, 2013, **61**(6): 1404−1418

15 Qiu J L, Wang H, Lu J B, Zhang B B, Du K L. Neural network implementations for PCA and its extensions. *International Scholarly Research Notices*, 2012, **2012**(4): Article No. 847305

16 Lewis D W. *Matrix Theory*. Singapore: World Scientific, 1991.

17 Du Bo-Yang, Kong Xiang-Yu, Feng Xiao-Wei. Direction convergence analysis of weighted rule for minor component extraction information criteria. *Journal on Communications*, 2020, **41**(3): 25−32
(杜柏阳, 孔祥玉, 冯晓伟. 次成分提取信息准则的加权规则方向收敛分析. 通信学报, 2020, **41**(3): 25−32)

18 Möller R. Derivation of Coupled PCA and SVD Learning Rules From a Newton Zero-finding Framework, Computer Engineering, Faculty of Technology, Bielefeld University, Berlin, 2017.

19 Gao Y B, Kong X Y, Hu C H, Li H Z, Hou L A. A generalized information criterion for generalized minor component extraction. *IEEE Transactions on Signal Processing*, 2017, **65**(4): 947−959

20 Zhang W T, Lou S T, Feng D Z. Adaptive quasi-Newton algorithm for source extraction via CCA approach. *IEEE Transactions on Neural Networks and Learning Systems*, 2014, **25**(4): 677−689

高迎彬 中国电子科技集团公司第五十四研究所高级工程师. 主要研究方向为自适应信号处理和神经网络.
E-mail: welcome8793@sina.com
(**GAO Ying-Bin** Senior engineer at the 54th Research Institute, China Electronics Technology Group Corporation. His research interest covers adaptive signal processing and neural networks.)

徐中英 火箭军工程大学副教授. 主要研究方向为统计信号处理和系统建模. 本文通信作者.
E-mail: xuzhy1978@163.com
(**XU Zhong-Ying** Associate professor at the Rocket Force University of Engineering. His research interest covers statistical signal processing and system modeling. Corresponding author of this paper.)