International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:4, No:5, 2010

An Algorithm of Ordered Schur Factorization For Real Nonsymmetric Matrix

Lokendra K. Balyan

Abstract—In this paper, we present an algorithm for computing a Schur factorization of a real nonsymmetric matrix with ordered diagonal blocks such that upper left blocks contains the largest magnitude eigenvalues. Especially in case of multiple eigenvalues, when matrix is non diagonalizable, we construct an invariant subspaces with few additional tricks which are heuristic and numerical results shows the stability and accuracy of the algorithm.

Keywords—Schur Factorization, Eigenvalues of nonsymmetric matrix, Orthoganal matrix.

I. Introduction

A Real square matrix A yields the decomposition of the form [6]

$$A = QTQ^T, \quad (1)$$

where Q is an orthogonal matrix and T is a block upper triangular matrix (also called 'quasi triangular' matrix).

$$T = \begin{bmatrix} T_{11} & T_{12} & \dots & T_{1m} \\ 0 & T_{22} & \dots & T_{2m} \\ \vdots & \vdots & \ddots & \\ 0 & 0 & & T_{mm} \end{bmatrix} = D + N, \quad (2)$$

where D is a diagonal matrix, which has all the diagonal elements are 1×1 and 2×2 blocks and N is strictly upper triangular. Scalar blocks contain the real eigenvalues and the complex conjugate eigenvalues correspond to the 2×2 blocks. In general, these 2×2 and 1×1 diagonal blocks can appear in any arbitrary order and reordered these blocks by swapping as our requirement, but this algorithm gives an ordered real Schur form.

There are some subroutines in LAPACK Library [8] and in NAG Library, which computes the Schur factorization of a nonsymmetric matrix for the user. In contrast of these work, the real Schur form has much less predictable structure. The new developed algorithm takes care of this irregularity and attains higher efficiency for a wide range of setting. No matter matrix is diagonalizable or not. The problem of reordered eigenvalues of a matrix in Schur factorization arises in the computation of the invariant subspaces. Swapping two 1×1 blocks or swapping 1×1 and 2×2 blocks proposed in [2] and later Bai and Demmel [1] proposed an algorithm for swapping 2×2 blocks.

The algorithm is based on the original idea proposed by Schur for real matrix [4], [6]. If we factorize a matrix with simple eigenvalues we can apply power method to find out the dominant eigenpair but if matrix has multiple eigenvalues, the dominant eigenpair can be chosen by any method suggested in *EISPACK* [7]. Our main concentration is to calculate the orthogonal invariant subspaces for real Schur form with ordered diagonal blocks without using unitary matrix, even if it has complex eigenvalues. Which plays an important role in many engineering problems.

The rest of the article is organized as follows: In section 2, we describe how to handle the real and complex eigenvalues, separately. Section 3 contains our main algorithm of real Schur form, applications and advantages. Finally, in section 4, we present numerical experiments. Henceforth the notation used in this paper are: matrices and vectors will be boldface capital letters and lower case letters respectively, if nothing else is stated.

A. Construction of orthogonal matrix:

Let **A** be a nonsymmetric real matrix with eigenvalues and corresponding eigenvectors are real, complex or both. Let λ be an eigenvalue and u an associated eigenvector, choose so that $\|u\|_2 = 1$. Now we define **U** be an orthogonal matrix that has u as its first column. In this section, we discuss both cases, for real and complex eigenvalues, separately. Theoretically this may not be required but numerically it is meaningful.

1) Case I:: First we discuss the case of real eigenvalues. For simplicity, we choose the columns of orthogonal matrix **U** as standard basis vectors of R^n with u as its first column, i.e. $\mathbf{U} = [u, \mathbf{V}] = (u, e_2, ..., e_n)$. Since u is a calculated eigenvector, it may not be orthonormal to other columns. To reduce these vectors in orthonormal vectors we use Gram Schmidt processor [6]. For this processor the vectors should be linearly independent but it is not always true. Most likely this type of situation occurs when u is an eigenvector corresponding to multiple eigenvalue. To face such problems we make a simple arrangement. Since e_i's are elements of standard basis, first component of u must be zero for linearly dependent. To deal with this we make a slight change by taking a constant *c* instead of zero as first component. For this modification, without loss of generality, we choose that column which has same number as first nonzero element of first column. Inside algorithm matrices denoted by capital letters only.

Department of Computer Sciences and Engineering, Indian Institute of Information Technology (DM)-Jabalpur, India. E-mail: balyan@iiitdmj.ac.in.

Algorithm 1.

1. *Calculate one eigenpair* $(\lambda, u)\ s.t.\ \parallel u \parallel_2 = 1$ used as input.
2. *Consider an orthogonal matrix s.t.*
   $U = [u, V] = [u, v_1, ..., v_{n-1}] = [u_1, ..., u_n]$
3. *If* U *is linearly independent goto step 4.*
   for $i = 2,\ n$
   if $(u_{i,1} \neq 0)\ then$
   $u_{1,i} = c$ and $u_{i,i} = 1$
   else
   $u_i = e_i$
   end if
   end do
4. *Make it orthonormal by Gram Schmidt process*
5. $A^1 = U^T A U$

2) Case II:: Here we deals with complex eigenvalues. Let $u + iv$, where u and v are real vectors, be the complex eigenvector associated with the complex eigenvalue $\alpha + i\beta$. Then $\alpha - i\beta$ is the remaining conjugate eigenvalue and $u - iv$ the corresponding eigenvector which are linearly independent over complex field because these eigenvectors are associated with distinct eigenvalues.

Let $a = a_1 + ia_2$ and $b = b_1 + ib_2 \in C$

$$a(u + iv) + b(u - iv) = 0 \Rightarrow a = 0, b = 0.$$

$$(a + b)u + i(a - b)v = 0.$$

$$\{(a_1 + b_1)u + (-a_2 + b_2)v\} + i\{(a_2 + b_2)u + (a_1 - b_1)v\} = 0.$$

$$(a_1 + b_1)u + (-a_2 + b_2)v = 0\,,\ (a_2 + b_2)u + (a_1 - b_1)v = 0.$$

$a_1 + b_1 = 0 = -a_2 + b_2 = a_2 + b_2 = a_1 - b_1$. Thus u and v are linearly independent over real field. Now we define an orthogonal matrix as

$$U = [u, v, V] = (u, v, e_3, ..., e_n). \tag{3}$$

In the case of complex multiple eigenvalues, which is rarely occur in practice, we will retrace the same path that we have just traversed for real multiple eigenvalues.

B. Real Schur Decomposition:

When working with nonsymmetric matrices, we must be prepared to deal with complex numbers. Since complex arithmetic is much slower than real arithmetic. To avoid complex arithmetic as much as possible, we arrange **U** is a real orthogonal matrix. It turns out that we can bring a matrix nearly to triangular form, called block triangular matrix, without using complex arithmetic. A quasi triangular matrix $\mathbf{T} \in R^n$, where each main diagonal blocks are either 1×1 or 2×2, and each 2×2 block has complex eigenvalues.

Let λ be any eigenvalue of the matrix $\mathbf{A} \in R^{n \times n}$. Now let us focus on the complex case, suppose $\mathbf{Ax} = \lambda\mathbf{x}$, where $\mathbf{x} = u + iv$, and $\lambda = \alpha + i\beta, \alpha, \beta \in R$ and $\beta \neq 0$. Then its numerical representation $\mathbf{AZ} = \mathbf{ZB}$ implies that the columns of $\mathbf{Z} = [u, v]$ span an invariant subspace of **A** and block **B** has complex eigenvalues of **A**. Since u, v are linearly independent vector, otherwise u and v must satisfy $\mathbf{A}u = \lambda u$ and $\mathbf{A}v = \lambda v$, and λ must be real. Finally, there exist an orthogonal matrix as discussed in §2 such that

$$\mathbf{A}^1 = \begin{bmatrix} C & & & & \\ 0 & 0 & * & \cdots & * \\ 0 & 0 & & \hat{\mathbf{A}} & \end{bmatrix},$$

where $\hat{\mathbf{A}} \in R^{(n-2 \times n-2)}$ and **C** is similar matrix to **B** of order 2.

Remark: For real eigenvalues $\hat{\mathbf{A}} \in R^{(n-1 \times n-1)}$.

Compute the Schur factorization of matrix **A** is given by

AU=UT,

where **U** is orthogonal and **T** is block triangular matrix. The main diagonal blocks are ordered according to decreasing magnitude of the corresponding eigenvalues. The columns of the orthogonal matrix **U** are the Schur vectors.

Algorithm 2:

U: Defined as identity matrix
$m = n$ and $l = 0$
while $(\ m \geq 2\ )\ do$

1. Calculate dominant eigenpair of matrix A
2. **if** $\beta = 0$ then
   2.1 $\lambda = \alpha$ and $\mathbf{x} = u$
   2.2 Apply algorithm 1 to get A^1 and Q^1
   2.3 $A = 0$;
   $A_{i,j} = A^1_{i+1,j+1}, \quad for \quad i, j = 1,\ m-1$

otherwise

   2.4 $\lambda = \alpha + i\beta$ and $\mathbf{x} = u + iv$
   2.5 Apply algorithm 1 for (2.1) to get A^1 and Q^1
   2.6 $A = 0$;
   $A_{i,j} = A^1_{i+2,j+2}, \quad for \quad i, j = 1,\ m-2$

step 2 end

3. $l = n - m$
   Q : Defined as identity matrix
   $Q_{i+l,j+l} = Q^1_{i,j}, \quad for \quad i, j = 1,\ m$
4. **if** $m = n$ then $A_1 = Q^T A\, Q$
   otherwise
   $A_1 = Q^T B\, Q$
   end if
   $B = A_1$
5. $U = U\, Q$
6. $m = m - 1$, for real eigenvalue
   $m = m - 2$, for complex eigenvalue

end while loop

7. $T = U^T A\, U$

C. Applications and Advantages:

An application of this procedure found for solving the algebraic *Riccati equation* [5] which arises in control theory, and other lies in the stable reduction of other problems to a more manageable form. Some important examples are the algorithm for solving the *Sylvester equation*, and algorithm

for finding few dominant cluster/multiple eigenvalues by *Block Arnoldi method* [3]. One of the best advantage of the Schur factorization is that it can be computed using only orthogonal transformations, which ensures good numerical behavior. A further consequence of the fact that **Q** is orthogonal is that this factorization leads to a stable and reliable method for computing invariant subspaces.

D. Numerical Experiments:

In this section, we present numerical results. For numerical experiments we apply this algorithm on random matrices of large as well as small order, which shows the accuracy of the method. We have used this algorithm for finding multiple eigenvalues of elliptic eigenvalue problems, in which its performance are not only satisfactory but highly accurate. To demonstrate the performance it is not possible to take large order of matrix, for convenience choose two examples and try to cover all possible cases.

Ex1. In this example we choose a 8×8 real nonsymmetric matrix which has real as well as complex eigenvalues.

$$\begin{bmatrix} 0.009 & -0.571 & 0.000 & 0.005 & 0.236 & -0.134 & 2.419 & 1.032 \\ 0.668 & -0.578 & 0.000 & 0.009 & 2.341 & 1.206 & -0.676 & 2.604 \\ 0.666 & 0.573 & 1.002 & 0.004 & 0.919 & -0.721 & 1.002 & -0.762 \\ 0.335 & 0.004 & 0.008 & 1.006 & 0.003 & 6.298 & 0.338 & 8.206 \\ 1.008 & -3.230 & 2.003 & 0.002 & 0.693 & 1.207 & -0.894 & 2.087 \\ 0.472 & 7.974 & -0.486 & 3.780 & 1.067 & 0.394 & 0.047 & -0.222 \\ 4.054 & 0.000 & 0.631 & 0.184 & 3.712 & 0.002 & 1.004 & 0.008 \\ 0.220 & -2.292 & 6.073 & 0.543 & -0.124 & 2.227 & 0.001 & 3.000 \end{bmatrix}$$

The eigenvalues of the matrix are: $7.90917451, 4.49188024, 1.25746458, -0.04245692,$ $-1.04853973 \pm 3.61163808, -2.49449148 \pm 0.53538985.$

$$\begin{bmatrix} 7.909 & 3.161 & -0.109 & -0.238 & 1.474 & -4.627 & 4.491 & -0.630 \\ 0.000 & 4.492 & 1.265 & 2.295 & 0.901 & -2.671 & -2.405 & 3.240 \\ 0.000 & 0.000 & 2.255 & 5.945 & 3.141 & -0.422 & 2.215 & 0.892 \\ 0.000 & 0.000 & -4.029 & -4.352 & -3.439 & 4.233 & 0.420 & -0.944 \\ 0.000 & 0.000 & 0.000 & 0.000 & -2.492 & -0.364 & -0.208 & 2.989 \\ 0.000 & 0.000 & 0.000 & 0.000 & 0.788 & -2.497 & -1.466 & -2.261 \\ 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 1.257 & 2.088 \\ 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & -0.042 \end{bmatrix}$$

Clearly the above matrix is in quasi triangular form which diagonal blocks contains decreasing magnitude eigenvalues, where scaler blocks $\mathbf{T}_{11}, \mathbf{T}_{22}, \mathbf{T}_{55}$ and $\mathbf{T}_{66}$ have real eigenvalues and complex conjugate eigenvalues correspond to 2×2 blocks $\mathbf{T}_{33}$ and $\mathbf{T}_{44}$.

Ex2. This example contains a 4×4 matrix in form of 2×2 blocks, one of them is Jordan block form with multiple eigenvalues and other has complex conjugate pair.

$$\mathbf{A} = \begin{bmatrix} 2.5e-5 & 1.0 & 0.0 & 0.0 \\ 0.0 & 2.5e-5 & 0.0 & 0.0 \\ 0.0 & 0.0 & 4.5e-5 & 3.4e-5 \\ 0.0 & 0.0 & -5.6e-5 & 6.0e-5 \end{bmatrix}$$

The eigenvalues of matrix **A** are: $0.00002500, 0.00002500, 0.00005250 \pm 0.00004299$ and largest magnitude is $\mid \lambda \mid = 0.00006786$, which is a complex eigenvalue.

$$\mathbf{T} = \begin{bmatrix} 4.5e-5 & 3.4e-5 & 0.0 & 0.0 \\ -5.6e-5 & 6.0e-5 & 0.0 & 0.0 \\ 0.0 & 0.0 & 2.5e-5 & 1.0 \\ 0.0 & 0.0 & 0.0 & 2.5e-5 \end{bmatrix}$$

II. Conclusion

In this paper I present an algorithm for Schur factorization of a real nonsymmetric matrix which diagonal elements are its eigenvalues in nondecreasing order. I have shown by numerical experiments the algorithm works for simple as well as multiple eigenvalues.

References

[1] Z. Bai, J.W. Demmel, On swapping diagonal blocks in real Schur form, *Lin. Alg. appli.*, Vol. 186, P. 73-95, 1993.
[2] P.V. Dooren, A generalized eigenvalue approch for solving Riccati equations, *SIAM Sci. and Stat. Comp*, Vol. 2, P. 121-135, 1981.
[3] D.L. Harrer, A block Arnoldi method for large nonsymmetric eigenvalue problem, Centre for Mathematics and its applications, School of Mathematical Science, *Australian National University, Canberra, ACT 0200*.
[4] Z. Bai, J Demmel, J. Dongarra, A. Ruhe, H. Vorst, Templates for the solustion of algebraic eigenvalue problems, A practical guide, *SIAM, Philadelphia*, (2000).
[5] J.J. Du Croz, S.J. Hamnarling, Eigenvalue Problems, in Numerical algorithm, *Oxford Uni. Press, Newyork*, P. 29-66.
[6] G.H. Golub and F. Van Loan, Matrix Computations (3^{rd} Edition) *Jhons Hopkins Uni. Press, Baltimore* 1996.
[7] B.S. Garbow, J.M. Boyle, J.J. Dongarra, C.B. Moler, Matrix Eigensystem Routines - *EISPACK Guide Extension*, Lecture Notes in Computer Science, Springer-Verlag, Vol. 51, 1977.
[8] E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. Mckenney, D. Sorensen, *LAPACK User's Guide. SIAM, Philadelphia*, 3^{rd} Edition, 1999.

Lokendra K. Balyan was born in 1981 in INDIA. He did his Master in Science in applied mathematics from Indian Institute of Technology Roorkee and Ph.D in applied mathematics and parallel computing from Indian Institute of Technology Kanpur in 2009. Since 2009, He is working as an assistant professor at Indian Institute of Information Technology Design and Manufacturing Jabalpur. His current research fields are Numerical algorithms and high performence computing.