Graded and Hessenberg Form together for Symmetric Matrices in QR Algorithm

Alpesh Virani[1] , Rahul Athale[2]

[*] M.Phil in Computer Science, Persistent Systems Limited
[**] Ph.D. in Mathematics

Abstract- The significance of the present research can be attributed to the systematic study of QR algorithm to solve eigenvalues. The main objective is to study how to develop a new method or strategy to find eigenvalues which improve the convergence of QR algorithm. It is observed that in general the QR algorithm succeeds when the matrix is graded downward with hessenberg form. Our future goal is to analyze theoretical proof for the same and find the well balanced input matrix for QR algorithm. This paper will helpful to all new students who want to work on Matrix decomposition problem.

Index Terms- QR algorithm, Graded and Hessenberg matrix and shifting algorithm

I. INTRODUCTION

Matrix multiplication algorithm is a very important algorithm for eigenvalues problem. Matrix multiplication is a very important part of the eigenvalue related problems. In QR algorithm if we will add more multiplication step then it will be helpful to add more parallelization in QR algorithm [1, 2].

We discuss the details of QR algorithm and important topics which need to understand this algorithm. We will present some results which are needed to understand to improve the performance of QR algorithm. In this section we discuss the results of our implementation of QR algorithm.

When dealing with graded matrix arrange the grading downward by rows, columns or diagonals and balance the matrix. The Wilkinson shift proposed by Wilkinson for symmetric tridiagonal matrices, where it insures global convergence of the QR algorithm [8]. If matrix is graded and change the shifting strategy, it will improve the convergence and reduce the number of iteration. In this section we took some results for graded downward matrix by diagonal with hessenberg form.

See the given reading in section 2 which we have taken on 8 GB RAM Workstation which have dual core processor with 3.16 GHz speed. We applied symmetric matrices as an input without hessenberg reduction. We are using random symmetric square matrices for input, there are inputs for matrices are between 1 to 9. We implement QR algorithm in java. We performed many tests for QR algorithm on given platforms. All this tests are performed for the purpose of finding execution time, suitable approach and the performance of QR algorithm on given machine.

1.1 The QR algorithm

All general-purpose eigenvalue algorithms are necessarily iterative. This statement is a consequence of Abel's famous proof that there is no algebraic formula for the roots of a general polynomial of degree greater than four. Specifically, to any polynomial $p(x) = x^n - a_{n-1}x^{n-1} - .....a_1x - a_0$ there corresponds a companion matrix

$$C_p = \begin{bmatrix} a_{n-1} & a_{n-2} & .... & a_1 & a_0 \\ 1 & 0 & ... & 0 & 0 \\ 0 & 1 & ... & 0 & 0 \\ : & : & & : & : \\ 0 & 0 & ... & 0 & 0 \\ 0 & 0 & ... & 1 & 0 \end{bmatrix}$$

Whose characteristic polynomial is *p*. Thus if we had a non-iterative general-purpose eigenvalue algorithm, we could apply it to the companion matrix of an arbitrary polynomial to find its zeros in a finite number of steps—in effect producing a formula for the roots. Thus the history of eigenvalue computations is the story of iterative methods for approximating eigenvalues and eigenvectors. An important theme in this story concerns methods based on powers of a matrix. The climax of that theme is the QR algorithm—the great success story of modern matrix computations [3]. This algorithm, with proper attention to detail, can compute a Schur form of a general matrix in *0(n3)* operations. It can be adapted to reduce a real matrix in real arithmetic to a real variant of the Schur form. Other forms of the algorithm compute the eigenvalues of and eigenvectors of a Hermitian matrix, the singular value decomposition of a general matrix, and the generalized Schur form of a matrix pencil. And the underlying theory of the method continues to suggest new algorithms. The algorithm is closely related to two algorithms that are important in their own right—the power method and the inverse power method..

The QR algorithm is a complicated device that generates a lot of code. Nonetheless, its various manifestations all have a family resemblance. The QR algorithm is a procedure for determining all eigenvalues of a real matrix A. The algorithm sequentially constructs matrices A_k $(K = 1, 2, ..., n)$ by forming QR decompositions.

The name QR derived from the letter Q to denote orthogonal matrices and R denote upper triangular matrices. It

computes the eigenvalues of real symmetric matrices, eigenvalues of real nonsymmetrical matrices and singular values of general matrices. Many people have contributed to development of various QR algorithms. But, first complete implementation and convergence analysis are due to J. H. Wilkinson. QR algorithm is an iterative process that is not always guaranteed to converge [3]. It is difficult to improve convergence without sacrificing accuracy and applicability.

1.1.1 QR decomposition

In linear algebra, a QR decomposition (also called a QR factorization) of a matrix is a decomposition of the matrix into an orthogonal and a right triangular matrix. QR decomposition is often used to solve the linear least squares problem, and is the basis for a particular eigenvalue algorithm, the QR algorithm. There are several methods for actually computing the QR decomposition, such as by means of the Gram-Schmidt process, Householder transformations, or Givens rotations. Each has a number of advantages and disadvantages. We used the Gram-Schmidt method in QR decomposing.

1.1.2 Gram-Schmidt process

The Gram-Schmidt orthogonalization process ([4] as presented in Chapter 11) may yield grossly inaccurate results due to round off error under finite-digit arithmetic (see Problems 20.10 and 20.11). A modification of that algorithm exists which is more stable and which generates the same vectors in the absence of rounding (see Problem 20.12). This modification also transforms a set of linearly independent vectors $\{X_1, X_2, ..., X_n\}$ into a set of orthonormal vectors $\{Q_1, Q_2, ..., Q_n\}$ such that each vector $Q_k \ (K = 1, 2, ..., n)$ is a linear combination of X_1 through. X_{k-1} The modified algorithm is iterative, with the kth iteration given by the following steps:

STEP 1: Set $r_{kk} = \| X_k \|_2$ and $Q_k = (1/r_{kk}) X_k$.

STEP 2: For j = κ + 1, κ + 2, n, set $r_{kj} = \langle X_j, Q_k \rangle$

STEP 3: For j = κ + 1, κ + 2, n, replace X_j by $X_j - r_{kj} Q_k$.

1.1.3 Householder transformation

A Householder reflection (or Householder transformation) is a transformation that takes a vector and reflects it about some plane [8]. We can use this operation to calculate the QR factorization of an m-by-n matrix *A* with m ≥ n. *Q* can be used to reflect a vector in such a way that all coordinates but one disappear.

1.1.4 Givens rotation

QR decompositions can also be computed with a series of Givens rotation [8]. Each rotation zeros an element in the sub diagonal of the matrix, forming the R matrix. The concatenation of all the Givens rotations forms the orthogonal Q matrix. In practice, Givens rotations are not actually performed by building a whole matrix and doing a matrix multiplication. A Givens rotation procedure is used instead which does the equivalent of the sparse Givens matrix multiplication, without the extra work of handling the sparse elements. The Givens rotation procedure is useful in situations where only a relatively few off diagonal elements need to be zeroed, and is more easily parallelized than Householder transformations.

1.1.5 The Hessenberg form

It is now to focus on the cost of the QR algorithm. Notice that the algorithm requires a decomposition A = QR which takes O (n3) operations. Because this needs to be repeated for each eigenvalue we obtain an overall cost of O (n4) which is prohibitively expensive. The cure to this problem is the transformation of the matrix to upper-Hessenberg form H. When this is done the total cost of the algorithm is O (n3) [6, 7].

QR algorithm is not always simple. It always preceded by reduction in hessenberg form. In which all the elements below sub diagonal elements are zero. Due to this factorization can be done much more quickly.

I found from one paper where real nonsymmetrical QR algorithm fail to converge for 4 by 4 matrixes. The double roots slow down convergence.

$$A = \begin{bmatrix} 0 & 2 & 0 & -1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

So, in some cases the shifted algorithm will fail. If we use double shift then we will get immediate satisfaction.

$$A = \begin{bmatrix} 2 & 3 & 3 & 3 \\ 0.002 & 3 & 3 & 6 \\ 0 & 0.001 & 2 & 1 \\ 0 & 0 & 0.01 & 7 \end{bmatrix}$$

Some times we will get very small values in subdiagnol elements that are not particularly small and also not negligible. That time iterative step will increase and convergences also slow down. Convergence is chaotic and, in some cases, the iteration limit is reached. We should plan to add a test that will avoid this behavior.

1.1.6 Deflation for QR

In the QR algorithm, convergence is detected by checking the sub diagonal entries of the Hesenberg matrix for given iteration. If the (i+1,i) sub diagonal entry satisfies

$$| h_{i+1,i} | \le u \max\{ | h_{i,i} |, | h_{i+1,i+1} | \},$$

---------- (A)

Where u denotes the round off (double precision u= 1.1 x 10E-16) then $h_{i+1,i}$ is set to zero and H becomes block upper

triangular. The eigenvalue problem deflates into two smaller problems associated with the two diagonal blocks of H, which can be treated separately by the QR algorithm. Typically, the convergence takes place at the bottom right corner and the size of the lower diagonal block is roughly the number of shifts used in the iteration.

Recently, the award winning aggressive early deflation (AED) strategy is used in addition to (A). We will see it why we should study AED in detail and why more efficient method is required for QR algorithm [6, 7].

1.1.7 Graded matrices

A matrix is graded if it shows a systematic decrease or increase in its elements as one proceeds from one end of the matrix to the other [10]. It is observed that in general the QR algorithm succeeds when the matrix is graded downward but it fails when the matrix is graded up-wards. Therefore balancing can help and solve the problem. For example, the matrix

$$A = \begin{bmatrix} -1.5940e+00 & -3.9990e-09 & 7.1190e-17 \\ -1.4410e-08 & 6.9000e-17 & 1.2900e-24 \\ 5.7110e-17 & 8.1560e-25 & 6.6860e-33 \end{bmatrix}$$

Exhibits symmetric grading from large to small along the diagonal. We therefore say that *A* is graded downward by diagonals. If we set

$$D = diag(1, 10^{-8}, 10^{-16})$$

Then

$$D^{-1}AD = \begin{bmatrix} -1.5940e+00 & -3.9990e-17 & 7.1190e-33 \\ -1.4410e+00 & 6.9000e-17 & 1.2900e-32 \\ 5.7110e-01 & 8.1560e-17 & 6.6860e-33 \end{bmatrix}$$

Is graded downward by rows, while

$$D^{-1}AD = \begin{bmatrix} -1.5940e+00 & -3.9990e-01 & 7.1190e-01 \\ -1.4410e+16 & 6.9000e-17 & 1.2900e-16 \\ 5.7110e-33 & 8.1560e-33 & 6.6860e-33 \end{bmatrix}$$

Is graded downward by columns

It turns out that the QR algorithm usually works well with matrices that are graded downward by diagonals. For example, the computed eigenvalues of *A* are

$$5.1034150572761617e-33$$
$$1.0515156210790460e-16$$
$$-1.5940000000000001e+00$$

Which are almost fully accurate? On the other hand, the algorithm does not do as well with other forms of grading. For example, reverse the rows and columns of *A* to get the matrix

$$B = \begin{bmatrix} 6.6860e-33 & 8.1560e-25 & 5.7110e-17 \\ 1.2900e-24 & 6.9000e-17 & 1.4410e-08 \\ 7.1190e-17 & -3.9990e-09 & -1.5940e+00 \end{bmatrix}$$

Which is graded upward by diagonals? If we compute the eigenvalues of *B,* we get the following results.

$$6.6859999999999994e-33$$
$$1.0515156534681920e-16$$
$$-1.5940000000000005e+00$$

The relative errors in these eigenvalues are

$$3.1e-01,\ 3.1e-08,\ 2.8e-16,$$

From which it is seen that the smallest eigenvalue has no accuracy at all, the second smallest is accurate to only eight decimal digits, and the largest is correct to working precision.

Row and column grading can also cause the algorithm to fail. The computed eigenvalues of $D^{-1}AD$ have relative errors of

$$1.0e+16,\ 2.3e+00,\ 1.4e-16,$$

While the computed eigenvalues of DAD^{-1} have relative errors of

$$1.3e+00,\ 3.4e-01,\ 0.0e+00.$$

The two smaller eigenvalues of both matrices have no accuracy. There is no formal analysis of why the algorithm succeeds or fails on a graded matrix. However, inspection of special cases will often show that the algorithm causes a loss of important information by combining large and small elements. For example, consider the first column of *B* and the corresponding first column of the Hessenberg form of *B:*

$$\begin{bmatrix} 6.6859999999999994e-33 & 6.6859999999999994e-33 \\ 1.2900000000000000e-24 & -7.1190000000000004e-17 \\ 7.1189999999999992e-17 & 0.0000000000000000e+00 \end{bmatrix}$$

This display shows that the first column of the Hessenberg form is the same as the one obtained by first setting the element 1.290e-24 to zero. This is a small error compared to the norm of *B,* but the smallest eigenvalue of *B* is quite sensitive to it. In particular the relative errors in the eigenvalues of the altered matrix *B* are

$$1.6e+00,\ 2.3e-16,\ O.Oe+00$$

Balancing can help in this case. For example, balancing DAD^{-1} and $D^{-1}AD$ restores the lost accuracy. On the other hand, balancing *B* has no effect, since it cannot change the direction of the grading. This suggests the following rule of thumb. We do not claim that this strategy is foolproof, and therefore anyone employing it should take a hard look at the results to see if they make sense.

1.2 QR Algorithm without Shifts

$A_0 = A$
For k=1, 2…
$Q_kR_k = A_k$
$A_{k+1} = R_kQ_k$
End

A_{k+1} tend to an upper triangular matrix with the same eigenvalues as A. These eigenvalues lie along the main diagonal of A_{k+1}. QR algorithm is using QR factorization. It is similar to gram-Schmidt process [4, 12].

1.3 QR Algorithm with Shifts

The step of QR iteration
$A - S * I = QR$
s = A (n, n) element of matrix A or also called shift (It accelerate convergence)
$A = R * Q + S * I$

The QR factorization makes matrix triangular. Each iteration is effectively transferring mass from lower to upper triangular [4, 9]. Finally, the matrix convert in upper triangular form and the diagonal elements represent the eigenvalues of that matrix.

1.4 The small bulge multishift QR algorithm

$(A_l - s_1 I).....(A_l - s_m I) \rightarrow Q_l R_l$

$A_{l+m} \leftarrow Q_l^H A_l Q_l$

Matrices:
A Hessenberg
Q Unitary
R Upper triangular

Shifts $s_1 ... s_m$

Perform m steps of the QR method at once. Computational procedure for single iteration is different here. It will take m/2 bulges and transform the matrix by chasing m/2 bulges [9, 11].

II. RESULTS AND ANALYSIS

We have implemented QR algorithm in java language. We have developed the package for matrix and vector related operations. We have taken the result for QR algorithm using input matrix as a Hessenberg form, graded downward and graded upward form. We convert the input matrix graded downward by diagonal and graded upward by diagonal. Our goal was to find the difference between these techniques. We can arrange the matrix graded downward by rows, columns and diagonal. It is already conclude that QR algorithm usually work well with matrices that are graded downward by diagonal. But, there is no formal analysis of why the algorithm succeeds or fails on graded matrix. So, this reading will help us to do this analysis or help in my research work.

Table 2.1 Table showing execution time for symmetric matrices on QR algorithm with graded matrix
CPU: Intel® Core™ 2 Duo CPU E8500@3.16 GHz, 8GB RAM

Matrix Size	Double shift (Time in millisecond)	Double shift (graded downward) (Time in millisecond)	Double shift (graded upward) (Time in millisecond)
4	3	3	3
4	49	46	6
5	17	59	72
6	82	75	41
8	241	73	57
10	159	152	231
12	3523	195	146
15	439	460	510
18	2699	3439	5528
20	4293	2534	1796
22	1603	2482	4424
25	NC	79778	3264
28	3813	8401	11811
30	20846	20738	53808
35	498401	42750	696350
40	26640	51758	177434
42	523374	255320	255607

The Table 2.1 is displaying the execution time for the QR algorithm with graded matrix and simple matrix. In these results we have not converge the matrix in hessenberg form. This result is looking very strange. We can't say that matrices which are

graded downward by diagonal are working well. In reading we can see that execution time is good for graded matrix. But, sometimes the matrices which are graded upward by diagonal are working better than graded downward as well as simple matrix. So, there is a need to analyze the graded matrix for improvement in eigenvalues method.

Table 2.2 Table showing execution time for symmetric matrices on QR algorithm with graded matrix + Hessenberg form
CPU: Intel® Core™ 2 Duo CPU E8500@3.16 GHz, 8GB RAM

Matrix Size	Double shift	Double shift (graded downward)	Double shift (graded upward)
4	7	7	6
4	71	33	4
5	31	9	82
6	76	54	71
8	83	81	90
10	111	137	190
12	3837	477	3755
15	537	567	583
18	2352	3449	3440
20	3489	3574	3535
22	1177	1118	1088
25	79565	4928	5034
28	3993	4050	7713
30	17346	386925	374848
35	10278	7774	7688
40	14332	10258	9629
42	588975	563333	537273

To check the conformability for reading, we check same matrices on different platform and we got same strange results. The most important thing we can conclude from the Table 2.1 is that the time difference is not very large when input simple matrices are working well. But, when the graded matrices working well that time, we can say time difference is very large. So, we can say that, graded matrices are really improving the performance of QR algorithm.

Table 2.3 Table showing execution time for symmetric matrices on QR algorithm with graded matrix
CPU: Intel® Core™ 2 Duo CPU E8500@2.93 GHz, 4GB RAM

Matrix Size	Double shift	Double shift (graded downward)	Double shift (graded upward)
4	43	50	11
4	3	4	4
5	19	73	79
6	69	55	48
8	251	123	80
10	199	166	264
12	3826	218	214
15	484	455	565
18	2960	3781	6119
20	4716	2766	1982
22	1748	2742	4870
25	NC	89110	3607
28	4248	9404	13257
30	23940	23522	60345
35	573290	49261	794631
40	31115	59345	203730
42	606363	293465	288396

The Table 2.2 is showing the execution time for QR algorithm. But, here we used hessenberg form after converting matrix in graded form. This result is looking very strange because, it is performing well but not for all size of matrices. But from the result, we can say hessenberg form will help when it will apply to graded matrix.

Table 2.4 Table showing execution time for symmetric matrices on QR algorithm with graded matrix + Hessenberg form
CPU: Intel® Core™ 2 Duo CPU E8500@2.93 GHz, 4GB RAM

Matrix Size	Double shift	Double shift (graded downward)	Double shift (graded upward)
4	8	11	7
4	32	39	5
5	23	11	92
6	71	88	68
8	88	107	181
10	124	156	184
12	4113	476	4101
15	534	605	543
18	2552	3616	3751
20	3938	3897	3964
22	1205	1256	1310
25	89012	5583	5649
28	4691	4692	8935
30	19587	440193	427867
35	11894	8999	8606
40	16645	11839	11156
42	678574	643774	615074

So, this combination of graded and Hessenberg form is helpful, when we want to improve the performance of QR algorithm. We will see the comparison of graded matrix with and without hessenberg matrix.

Table 2.3 shows the result of QR algorithm without Hessenberg matrix on 4GB RAMS with Processor speed of 2.93 GHz. These results will show the same graph as previous results for 8GB RAM. And Table 2.4 shows the results for QR algorithm with Hessenberg matrix. We can go for conclusion, after analyzing the graph for the given table. We compare the graded matrix output with and without Hessenberg form. These results are really useful to conclude something and do the research in that direction.

For better understand of results, we draw the comparison graph for execution time. We can see from Figure 2.1, that the graded downward matrix by diagonal is taking less execution time than the other two approaches. It is remarkable that graded upward matrix by diagonal also have mix response with respect to standard input. From this graph we can say that we should develop a method which can be helpful to improve the performance of QR algorithm. People are not using graded matrix for input due to this kind of mix approach. That's why we took reading of graded downward with hessenberg matrix. And we got some good results for this mixed approach.

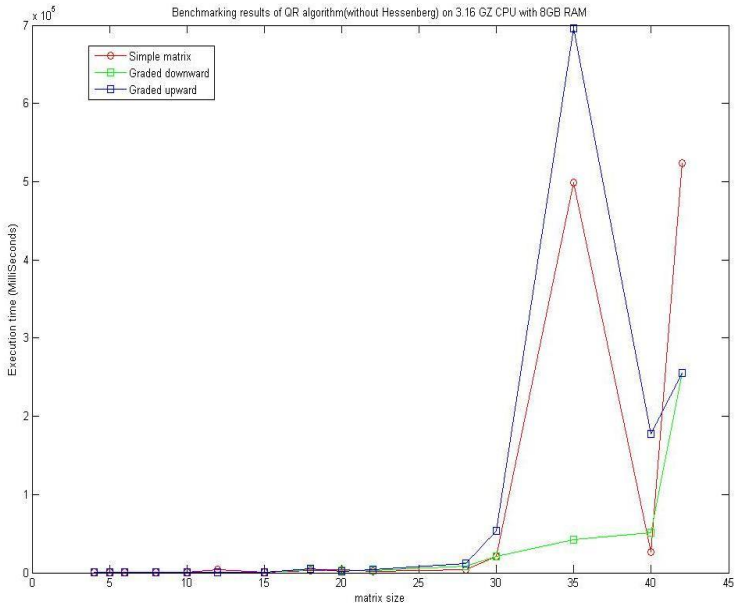

Figure 2.1 Benchmarking results of QR algorithm (without Hessenberg) on 3.16 GHZ CPU with 8GB RAM

Figure 2.2 is showing graph for graded upward and graded downward with the hessenberg approach. From this graph we can say graded upward by diagonal matrix also performing well. Most of the times all three approaches have equal performance but graded upward by diagonal has some good results compare to other. Figure 2.3 shows the same comparison but platform with 4 GB and CPU speed with 2.93 GHZ. Results of this machine

also have same response like 8GB RAM and 3.16 GHz CPU speed. So, it is confirm we may get good results on all kind of platform.

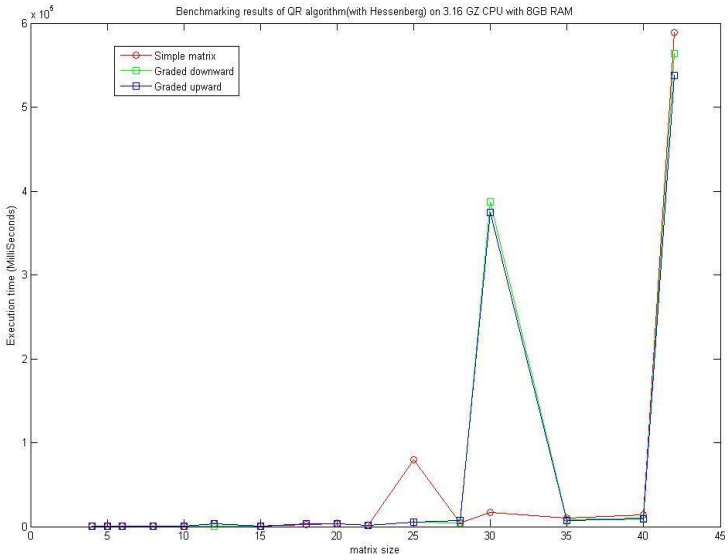

Figure 2.2 Benchmarking results of QR algorithm (with Hessenberg) on 3.16 GHZ CPU with 8GB RAM

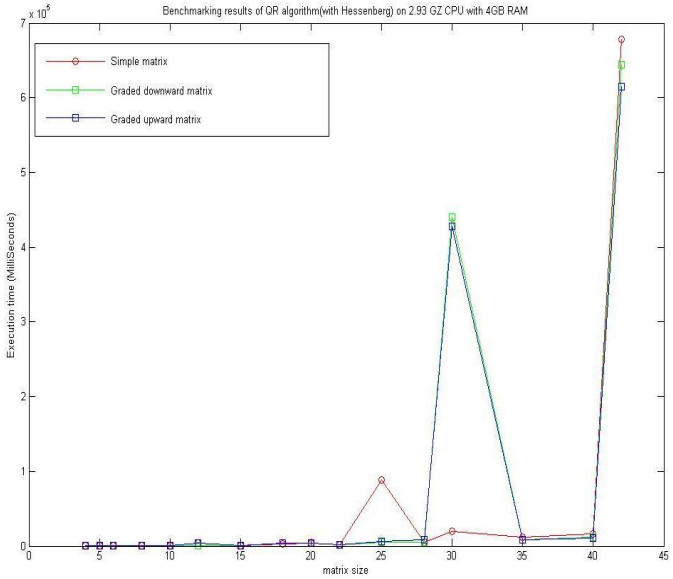

Figure 2.3 Benchmarking results of QR algorithm (with Hessenberg) on 2.93 GHZ CPU with 4GB RAM

Why we are speaking about different platform? There is a very good question. We also took some results for matrix multiplication on different platform. But, we got the very strange results. CPU with less speed has given good results than the CPU with more speed [1, 2].

We see it in previous results that graded downward have good performance comparatively simple input matrix. To get more clear results, we compare the graded downward by diagonal with Hessenberg and without hessenberg. See Figure 2.4, we can say that only in two readings graded downward without hessenberg have less execution time. So, it is confirm that if we add this kind of approach in graded downward by diagonal matrix, we can improve the performance of QR algorithm. We will work more towards this mixed approach. Figure 2.5 presenting the same graph for the results.

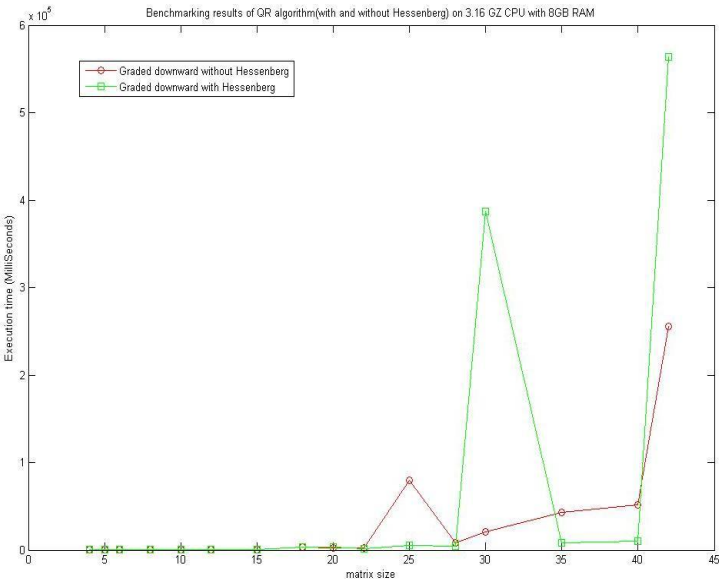

Figure 2.4 Benchmarking results of QR algorithm (with and without Hessenberg) on 3.16 GHZ CPU with 8GB RAM

From Figure 2.4 and 2.5, we can say there is a requirement to analyze the graded matrix with other different methods. For symmetric matrices, the Hessenberg reduction process leaves a symmetric at each step, so zeros are created in symmetric positions. This means we need to work on only half the matrix, reducing the operation count to $4/3n^3 + O(n^2)$ or $8/3n^3 + O(n^2)$ to form $Q_{n-1}....Q_1$ as well. So, it is really helpful to check these approaches and develop a new method which will improve the performance of QR algorithm.

We can also say the reading is not so helpful to predict the new method development for QR algorithm. But, the results of graded downward matrix with the hessenberg form are very satisfactory. So, we will continue our research in the same direction.

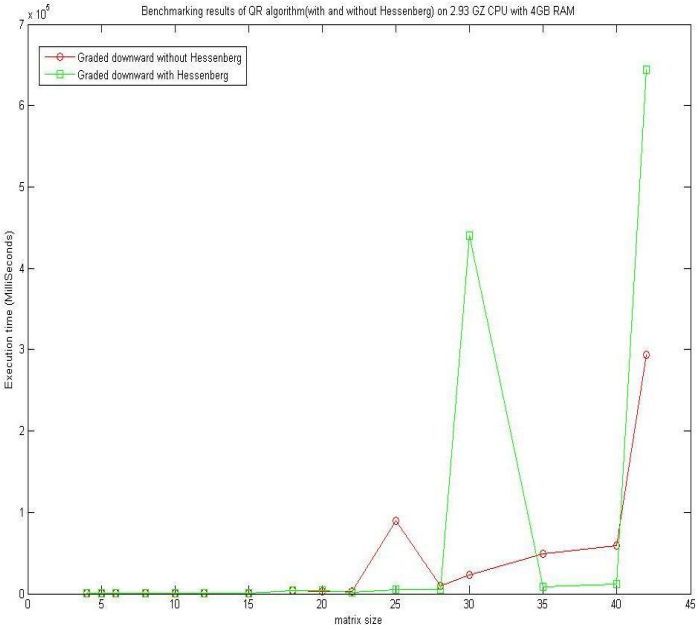

Figure 2.5 Benchmarking results of QR algorithm (with and without Hessenberg) on 2.93 GHZ CPU with 4GB RAM

We also compare the results of the QR algorithm with and without hessenberg matrix. The hessenberg matrix is near to the diagonal matrix, so it will reduce the operation count for the QR algorithm From Figure 2.6 we can say QR algorithm with hessenberg matrix will work better than without hessenberg. Our main goal is to improve the convergence and reduce the number of iteration for QR algorithm. If we are able to do this, we can work towards the development of new method. We also try to make a change in shifting strategy. It was not giving the satisfactory results for the QR algorithm. The conclusion from this reading is that there is a need to analyze the graded matrix why it wills success and fails. In one reading you can see that simple matrix is not able to converge but graded matrix is converging in satisfactory time. So, from this reading we are going to work towards the development of new method for graded matrix. We will try to analyze the graded matrix. Hence, it is final that there is still requirement to improve the convergence and performance of eigenvalues related algorithm.

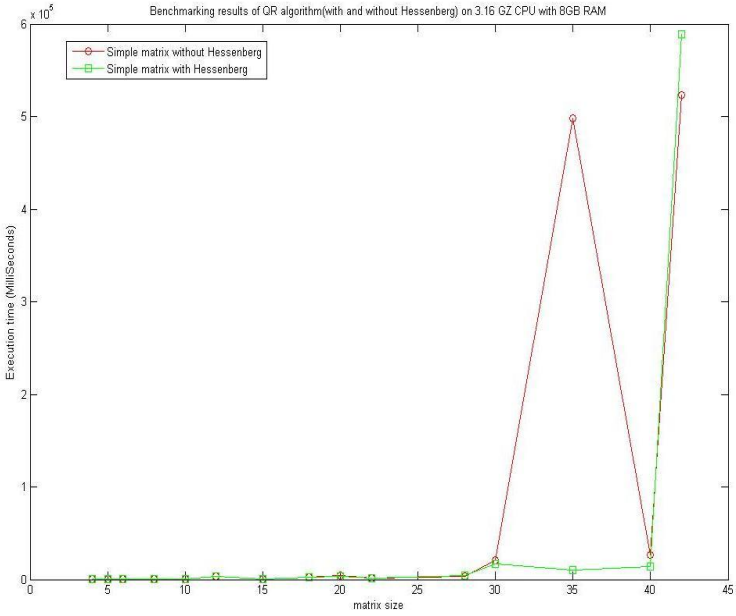

Figure 2.6 Benchmarking results of QR algorithm (with and without Hessenberg) on 3.16 GHZ CPU with 8GB RAM

After the discussion of different input matrices for QR algorithm, we will discuss and analyze the output of input matrices. From the previous results we know that graded downward by diagonal is working well than simple matrix. We also conclude that graded with hessenberg has overall good performance than graded downward by diagonal only. So, we try to analyze the convergence rate of graded downward by diagonal and graded downward by diagonal with hessenberg form. We got some good results, which are help to understand the development of K. Braman, R. Byers and R. Mathias, 2002. This development is very helpful to converged eigenvalue early.

From the table 2.5 we can say eigenvalue for the 5 x 5 symmetric matrix is converged very early at 60^{th} iteration. But we are not able to find converged eigenvalues. If we will see carefully the results of Table 2.5 all the eigenvalues are converged before the 80^{th} iteration. From these results it is final that we need to work on a method which will be helpful to find the converged eigenvalues earlier than the standard QR algorithm do. AED is the method which is used to find eigenvlue earlier than the standard convergence criteria.

Table 2.5 Converged eigenvalue with respect to iteration for QR algorithm using input matrix with graded downward by diagonal (matrix size 5 x 5)

Number of iteration	First eigenvalue	Second eigenvalue	Third eigenvalue	Fourth eigenvalue	Fifth eigenvalue
5	30.342254314 02704	-0.2442687194 0302908	0.7105141732 318789	3.6777781251 28669	1.1478337642 388863
20	30.342254314 02701	-0.24	0.4617984168 0387534	3.9270572969 150654	1.1467879337 033158
40	30.342254314 027	-0.2448321328 4536331	0.4616173098 687457	3.9272384038 501933	1.1467879337 032434
60	30.342254314	-		3.9272384137	1.1467879337

	027	0.2448321328 4536331	0.4616172999 4545997	734786	032434
80	30.342254314 027	- 0.2448321328 4536331	0.4616172999 449115	3.9272384137 740257	1.1467879337 032434
100	30.342254314 026995	- 0.2448321328 4536331	0.4616172999 4491285	3.9272384137 740253	1.8347423174 298956
120	30.342254314 026967	- 0.2448321328 4536243	0.4616172999 4491374	3.9272384137 740235	1.1467879337 03243
130	30.342254314 02696	- 0.2448321328 453611	0.4616172999 449143	3.9272384137 74022	1.1467879337 032412
140	30.342254314 02696	- 0.2448321328 453611	0.4616172999 449143	3.9272384137 74022	1.1467879337 032412
150	30.342254314 02696	- 0.2448321328 453611	0.4616172999 449143	3.9272384137 74022	1.1467879337 032412

See the results of Table 2.6 the eigenvalues are converged early then the Table 2.5. We can see from the table 2.6 the Most of the eigenvalues are converged before 40^{th} iteration. So, after comparing these two approaches we can say the previous results are really helpful to work on graded downward with hessenberg. So, our goal is to find the direction where we can work on to development of new method. This thesis will help to us get good direction for my PhD work.

Table 2.6 Converged eigenvalue with respect to iteration for QR algorithm using input matrix with graded downward by diagonal with hessenberg

Number of iteration	First eigenvalue	Second eigenvalue	Third eigenvalue	Fourth eigenvalue	Fifth eigenvalue
5	30.342254314 027045	3.9270277154 6233	3.3670942373 445594	- 0.2447815004 820959	0.4616172999 449124
20	30.342254314 027045	3.9272384110 01787	3.3669341741 68375	- 0.2448321328 453651	0.4616172999 449124
40	30.342254314 027045	3.9272384137 7403	3.3669341713 961343	- 0.2448321328 453651	0.4616172999 449124
60	30.342254314 027045	3.9272384137 7403	3.3669341713 961343	- 0.2448321328 453651	0.4616172999 449124
80	30.342254314 027045	3.9272384137 7403	3.3669341713 961343	- 0.2448321328 453651	0.4616172999 449124
100	30.342254314 02703	3.9272384137 740293	3.3669341713 961343	- 0.2448321328 4536487	0.4617680697 8837115
120	30.342254314 02703	3.9272384137 740275	3.3669341713 961334	- 0.2448321328 4536465	0.4616172999 449121
140	30.342254314 02703	3.9272384137 740275	3.3669341713 961334	- 0.2448321328 453642	0.4616172999 4491196
150	30.342254314 02703	3.9272384137 740275	3.3669341713 961334	- 0.2448321328 453642	0.4616172999 449121

184	30.342254314 02703	3.9272384137 740275	3.3669341713 96131	- 0.2448321328 453642	0.4616172999 4491196

III. CONCLUSION

People are not using graded matrix for input due to the problem of convergence. That's why we took reading of graded downward and graded downward with hessenberg matrix. And we got some good results for this mixed approach. So, after comparing the two approaches by convergence, we can say the results are really helpful to work on graded downward with hessenberg. This thesis will help to us get good understanding of why combination of two method will help to improve the performance of the algorithm.

From the previous results we know that graded downward by diagonal is working well than simple matrix. We also conclude that graded with hessenberg has overall good performance than graded downward by diagonal only. So, we try to analyze the convergence rate of graded downward by diagonal and graded downward by diagonal with hessenberg form. We got some good results, which are help to understand the development of K. Braman, R. Byers and R. Mathias, 2002. This development is very helpful to find converged eigenvalue early than the conventional deflation strategy.

IV. FUTURE WORK

This research work has an objective in mind to improve the performance of QR algorithm. It is already conclude that the QR algorithm usually work well with matrices that are graded downward by diagonals. There is no formal analysis of why the algorithm succeeds or fails on a graded matrix. Our next objective is to give the theoretical proof why graded downward

or graded upward approach will help in finding eigenvalues for matrices.

REFERENCES

[1] Alpesh Virani Sandip Tujare and Rahul Athale,Blocked matrix multiplication , International conference on mathematics and computer science, vol 1, pp. 323-326, 2009.

[2] Alpesh Virani and Rahul Athale, Different parallel approach for matrix multiplication (MPI and MPI + Openmp), International conference on Information technology and business Intelligence, Nagpur, India, 6-8 November 2009.

[3] David S. Watkins, The QR algorithm revisited, SIAM, vol 50, No. 1, pp 133-145, 2008

[4] Richard Bronson," Schaum's Outline of Theory and Problems of Matrix Operations", Tata McGraw-Hill Edition 2005, ISBN 0-70-060480

[5] G.W. Stewart," Matrix Algorithms Volume II: Eigen Systems", 2002, SIAM, ISBN 0-89871-414-1

[6] K. Braman, R. Byers and R. Mathias, The multishift QR algorithm Part-I: maintaining well focused shifts and level 3 performance, SIAM J. matrix anal. Appl., 23, pp. 929-947, 2002

[7] K. Braman, R. Byers and R. Mathias, The multishift QR algorithm Part-II: Aggressive early deflation, SIAM J. matrix anal. Appl., 23, pp. 948-973, 2002

[8] G.W. Stewart," Matrix Algorithms Volume I: Basic Decompositions", 1998, SIAM, ISBN 0-89871-414-1

[9] D. S. Watkins, The transmission of shifts and shift blurring in the QR algorithm, Linear Algebra Appl., 241/243; pp 877-896, 1996

[10] H. Golub, Charles F. Van Loan, "Matrix Computation", 3rd edition, 1996, The Johns Hopkins University press, Baltimore and London, ISBN 0-8018-5431-X

[11] D. S. Watkins, Forward stability and transmission of shifts in the QR algorithm, SIAM J> Matrix Ana. Appl., 16(2): pp 469-487, 1995

[12] J. G. F. Francis, The QR transformation, Part I and II, Computer Journal, 4: pp 265-271, pp 332-345, 1961, 1962

[13] Matrix Market – A visual repository of test data for use in comparative studies of algorithms for numerical linear algebra, See http://math.nist.gov/MatrixMarket/

AUTHORS

First Author – Alpesh Virani, M.Phil in Computer Science, Persistent Systems Limited and alpeshviranik@gmail.com.
Second Author – Rahul Athale, Ph.D. in Mathematics and athale.rahul@gmail.com.