Origin-shifted algorithm for matrix eigenvalues

Y. Y. Nie\textsuperscript{a,}\textsuperscript{*}, Z. Li\textsuperscript{b} and J.D. Han\textsuperscript{a}

\textsuperscript{a}Robotics Laboratory, Shenyang Institute of Automation, Academia Sinica No.114, Nanta Avenue, Shenyang 110016, P.R. China; \textsuperscript{b}School of Sciences, Northeastern University, Shenyang 110004, P.R. China

(Received 16 August 2006; revised version received 23 October 2006; accepted 10 June 2007)

In this paper an origin-shifted algorithm for matrix eigenvalues based on Frobenius-like form of matrix and the quasi-Routh array for polynomial stability is given. First, using Householder's transformations, a general matrix $A$ is reduced to upper Hessenberg form. Secondly, with scaling strategy, the origin-shifted Hessenberg matrices are reduced to the Frobenius-like forms. Thirdly, using quasi-Routh array, the Frobenius-like matrices are determined whether they are stable. Finally, we get the approximate eigenvalues of $A$ with the largest real-part. All the eigenvalues of $A$ are obtained with matrix deflation. The algorithm is numerically stable. In the algorithm, we describe the errors of eigenvalues using two quantities, shifted-accuracy and satisfactory-threshold. The results of numerical tests compared with QR algorithm show that the origin-shifted algorithm is fiducial and efficient for all the eigenvalues of general matrix or for all the roots of polynomial.

Keywords: matrix eigenvalues; origin shifts; Hessenberg matrix; Frobenius-like form; quasi-Routh array

2000 AMS Subject Classification: 15A21; 93D05; 65H17

1. Introduction

Between 1958 and 1965 the QR algorithm was developed by Rutishauser and Francis [1–3,9,10]. Since then there is no significant advance on algorithm for all the eigenvalues of a general real matrix $A$. The QR algorithm based on power method and unitary transformations is a numerically stable method with 2-degree speed of convergence. Suppose $H$ is an upper Hessenberg matrix, then $O(n^2)$ flops are required for each iterative step, from $H_k$ to $H_{k+1}$. In order to get one or two eigenvalues with very high accuracy, the QR algorithm needs a great many and random iterative steps. Moreover, the backward error of QR algorithm is proportional to number of iterative steps, see [10]. If there are many multiple-eigenvalues or many eigenvalues with a small module, then all the power methods, inverse power method and QR algorithm may bring about difficulty of computation and give a non-fiducial solution, see, for example, [4].

On the other hand, finding roots of the eigenpolynomial in root-distributed theory, see, for example, [8], all the iterations are only origin-shifted transformations in an interval with the length

\*Corresponding author. Emails: ny@sia.cn; jdh@sia.cn

ISSN 0020-7160 print/ISSN 1029-0265 online
© 2008 Taylor & Francis
DOI: 10.1080/00207160701504105
http://www.informaworld.com
and the speed of convergence is exponential. However, the synthetic division of origin-shifted transformation and deflation of polynomial is at the risk of numerical instability.

Using scaling strategy we have developed an algorithm, where, with numerical stability, the origin-shifted Hessenberg matrices $H - sI$ are reduced to the Frobenius-like forms, see [5]. Then, using the Frobenius-like matrix and its quasi-Routh array for polynomial stability, we can develop a numerically stable algorithm which finds the approximate eigenvalues of $H$ with the largest real-part, see [6–8]. After the eigenvalues with the largest real-part are found, the order of origin-shifted matrix $H - sI$ may be deflated into, generally, $(n - 1)$ or $(n - 2)$. The process of matrix deflation is also numerically stable.

Continuing in this way, we may find all the eigenvalues one by one, working with matrices of progressively decreasing order.

Here, the accuracy of complex-eigenvalues is measured by exponential decreasing series of an interval length. This is different from the QR algorithm.

In fact, it is not satisfactory to measure the accuracy of eigenvalues by some elements of origin-shifted matrix $H - sI$, because, first, these elements are possibly contracted or enlarged randomly, secondly, there is no quantitative analysis of the relation between them and accuracy of eigenvalues. So it is better to measure the accuracy of the eigenvalue, specially of the complex-eigenvalue, by a small interval including the real-part of the eigenvalue, see [8].

This paper gives an origin-shifted algorithm for matrix eigenvalues or for polynomial roots based on the quasi-Routh array of Frobenius-like matrix. The origin-shifted algorithm is numerically stable. It can generally give all the eigenvalues with very high accuracy. If the algorithm cannot give a solution with high accuracy for some matrix, then it can print the information about failure of computation. In the algorithm, we describe the errors of eigenvalues using two quantities, shifted-accuracy and satisfactory-threshold, which are usually dependent on each other. According to ref. [8], shifted-accuracy and satisfactory-threshold can be selected automatically in the algorithm, and they together describe the errors of all the eigenvalues found. The results of numerical tests compared with the QR algorithm show that the algorithm is fiducial and efficient for all the eigenvalues of general matrix or for all the roots of polynomial.

2. Scaling strategy for Frobenius-like matrix

Using stabilized elementary transformations or Householder’s transformations, we can reduce a general real-matrix $A$ to upper Hessenberg form, the reduction is numerically stable for initial perturbation and rounding errors, see [5,10]. As the QR algorithm did, let $A$ be an upper Hessenberg matrix

$$A = \begin{pmatrix}
a_{1,1} & a_{1,2} & \cdots & a_{1,n-1} & a_{1,n} \\
h_2 & a_{2,2} & \cdots & a_{2,n-1} & a_{2,n} \\
\vdots & \vdots & \ddots & \vdots & \vdots \\
 & & & a_{n-1,n-1} & a_{n-1,n} \\
h_n & & & h_n & a_{n,n}
\end{pmatrix}. \tag{1}$$

We denote the off-diagonal elements $a_{r+1,r}$ by $h_{r+1}$ and, without loss of any generality, we may assume that $h_i \neq 0$ $(i = 2, \ldots, n)$.

The matrix having the form

$$F_l = \begin{pmatrix}
0 & 0 & \cdots & 0 & p_0 \\
2^{\beta_2} & 0 & \cdots & 0 & p_1 \\
\vdots & \vdots & \ddots & \vdots & \vdots \\
& & & 2^{\beta_n} & p_{n-1}
\end{pmatrix}. \tag{2}$$
is called Frobenius-like form, where $\beta_i (2 \leq i \leq n)$ is a proper integer. The Frobenius-like form (2) becomes the Frobenius form of a matrix when $\beta_2 = \cdots = \beta_n = 0$. The coefficients of eigen-polynomial $(-1)^{n-1} \det(A - \lambda I)$ satisfy, where $\det(A - \lambda I)$ represents determinant of the matrix $A - \lambda I$,

$$q_0 = 2^{\beta_1 + \cdots + \beta_n} p_0, \quad q_i = 2^{\beta_i+1 + \cdots + \beta_n} p_i, \quad i = 1, \ldots, n - 2; \quad q_{n-1} = p_{n-1}; \quad q_n = -1 = p_n.$$ 

That is, the scaling coefficients $p_i (i = 0, 1, \ldots, n - 1)$ of eigenpolynomial are interested in Frobenius-like form.

Wilkinson’s numerical stability is specific to problems where rounding errors are the dominant form of errors. The term numerical stability has different meanings in other areas. An algorithm for computing $y = f(x)$ is called numerically stable if, for any $x$, it produces a computed $\hat{y}$ with a small absolute error $|\hat{y} - y|$, where $\hat{y} = f(x + \Delta x)$ for some small $\Delta x$. The definition of ‘small’ will be context-dependent. In general, a given problem has several methods for solution, some of which are numerically stable and some not. According to the definition of numerical stability, the known pivoting strategy can ensure that the absolute error estimation of multiplication is not greater than that of addition. That is, an algorithm is numerically stable if the pivoting strategies are adopted in each multiplication, see [5,10]. When the pivoting strategy of multiplication cannot be achieved, we may consider another strategy, which may be called scaling strategy, namely, we may adopt the pivoting strategy for a computed product. If the scaling products are available for some computing process, then the scaling strategy is equivalent to the pivoting strategy, see [5]. In order to adopt scaling strategy for the reduction from the Hessenberg matrix (1) to the Frobenius-like form (2), we consider a similar transformation of the matrix (1).

**Theorem 1** The matrix (1) is similar to the matrix given by

$$\tilde{A} = \begin{pmatrix}
    a_{1,1} & w_{1,2} & \cdots & w_{1,j} & \cdots & w_{1,n-1} & w_{1,n} \\
    k_2 & a_{2,2} & \cdots & w_{2,j} & \cdots & w_{2,n-1} & w_{2,n} \\
    \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \vdots \\
    k_i & \cdots & w_{i,j} & \cdots & w_{i,n-1} & w_{i,n} \\
    \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \vdots \\
    a_{n-1,n-1} & w_{n-1,n} & \cdots & w_{n-1,2} & \cdots & \cdots & a_{n,n}
\end{pmatrix} \quad (3)$$

where $k_{i+1} = 2^{\beta_{i+1}}$, $w_{i,j} = 2^{-\beta_{i+1} - \cdots - \beta_j} h_{i+1} \cdots h_j a_{i,j}$, $(i = 1, \ldots, n - 1; \ j = i + 1, \ldots, n)$, $\beta_2, \ldots, \beta_n$ are integers.

**Proof** Let

$$D = \begin{pmatrix}
    1 & & & & & \\
    & d_2 & & & & \\
    & & \ddots & & & \\
    & & & \ddots & & \\
    & & & & \ddots & \\
    & & & & & d_n
\end{pmatrix},$$

where $d_i = 2^{\beta_1 + \cdots + \beta_i} h_{i-1}^{-1} \cdots h_2^{-1}, (i = 2, \ldots, n)$. Then $D^{-1}$ exists because $h_i \neq 0 (i = 2, \ldots, n)$, and a similar matrix of $A$ is $\tilde{A} = DAD^{-1}$. \hfill \blacksquare
Upper triangular elements of the matrix (3) are given by
\[ 2^{-\beta_{i+1}} \cdots 2^{-\beta_i} h_{i+1} \cdots h_j a_{ij}, \]  
(4)
which have rounding errors of continued product. In floating-point computation we have, see [10],
\[ f \ell(h_{i+1} \cdots \cdots h_j a_{ij}) = (1 + \varepsilon_{ij}) h_{i+1} \cdots h_j a_{ij}, \]
where error bounds satisfy strictly in mantissa digits \( t \) of computer
\[ |\varepsilon_{ij}| < (j-i)2^{-t}, \quad t_1 = t - 0.08406, \]  
(5)
\( (i = 1, \ldots, n-1; \quad j = i + 1, \ldots, n) \); or statistically for larger \( (j - i) \)
\[ |\varepsilon_{ij}| < (j-i)^{1/2}2^{-t}, \]  
(6)
\( (i = 1, \ldots, n-1; \quad j = i + 1, \ldots, n) \). We may regard \( 1 + \varepsilon_{ij} \) as initial relative perturbations of upper triangular elements (4) of the similar matrix (3). Evidently, the bounds of initial relative perturbations are increased with \( (j - i) \) because of estimate (5) or (6). The maximal bound of relative perturbations is given by
\[ 1 - (n-1)2^{-t} < 1 + \varepsilon_{1n} < 1 + (n-1)2^{-t} \]
or statistically
\[ 1 - (n-1)^{1/2}2^{-t} < 1 + \varepsilon_{1n} < 1 + (n-1)^{1/2}2^{-t}. \]
If we permit 7-digit binary precision loss for floating-point computation, then computed elements of the similar matrix (3) are acceptable for \( n \leq 121 \) at least.

Suppose in (3) of Theorem 1
\[ a_{i,j} = 2^n \xi_i \quad (i = 1, \ldots, n), \quad h_{i+1} \cdots h_j a_{i,j} = 2^n \psi_{ij} \eta_{ij} \quad (i = 1, \ldots, n-1; \quad j = i + 1, \ldots, n), \]
where \( \theta_i \) and \( \psi_{ij} \) are integers, and except number zero, \( \xi_i \) and \( \eta_{ij} \) satisfy
\[ \frac{1}{2} \leq |\xi_i| \leq 1, \quad \frac{1}{2} \leq |\eta_{ij}| \leq 1. \]
Then \( w_{i,j} = 2^{\psi_{i,-\beta_{i+1}} \cdots -\beta_i} \eta_{ij} \). We select the integers \( \beta_2, \beta_3, \ldots, \beta_n \) such that
\[ \beta_2 \geq \theta_2, \quad \beta_3 \geq \max\{\beta_2, \theta_3, \psi_{23} - \beta_2\}, \ldots, \]
\[ \beta_i \geq \max\{\beta_{i-1}, \theta_i, \phi_2, \ldots, \phi_{ji}, \ldots, \phi_{i-1,i}\}, \]
\[ \ldots, \beta_n \geq \max\{\beta_{n-1}, \theta_n, \phi_{2n}, \ldots, \phi_{n-1,n}\}, \]  
(7)
where \( \phi_{ji} = \psi_{ji} - \sum_{k=j}^{i-1} \beta_k \quad (i = 3, \ldots, n; \quad j = 2, \ldots, i - 1) \). Then the following integer inequalities hold in (3)
\[ \beta_2 \leq \beta_3 \leq \cdots \leq \beta_n; \]
\[ \beta_2 \geq \theta_2, \beta_2 \geq \psi_{23} - \beta_3, \ldots, \beta_2 \geq \psi_{2n} - \sum_{k=3}^{n} \beta_k; \]
\[ \ldots, \beta_i \geq \theta_i, \beta_i \geq \psi_{i,i+1} - \beta_{i+1}, \ldots, \beta_i \geq \psi_{in} - \sum_{k=i+1}^{n} \beta_k; \]
\[ \ldots, \beta_n \geq \theta_n. \]  
(8)
After the integers \( \beta_2, \ldots, \beta_n \) are defined by the scaling strategy (7), all the elements of upper Hessenberg matrix (3) are given. Equation (3) is called scaling-matrix of (1).
Using elementary similarity transformations, we can reduce the scaling-matrix (3) to the Frobenius-like form. The reduction consists of \((n - 1)\) major steps; at the beginning of the \(r\)th-step the current matrix for case \(n = 6, r = 3\), is given by

\[
\begin{pmatrix}
0 & 0 & h_{13} & h_{14} & h_{15} & h_{16} \\
k_2 & 0 & h_{23} & h_{24} & h_{25} & h_{26} \\
k_3 & h_{33} & h_{34} & h_{35} & h_{36} \\
k_4 & a_{44} & w_{45} & w_{46} \\
k_5 & a_{55} & w_{56} \\
k_6 & a_{66}
\end{pmatrix}
\]

The \(r\)th-step is as follows.

**Algorithm 1**  For each value of \(i\) from 1 to \(r\) perform steps (i) and (ii):

(i) Let \(\alpha_{i,r+1} = h_{ir}/k_{r+1}\), but not compute \(\alpha_{i,r+1}\);

(ii) Subtract \(\alpha_{i,r+1} \times \text{row}(r+1)\) from \text{row}(i), namely

\[
h_{i,r+1} := h_{i,r+1} - \left(\frac{a_{r+1,r+1}}{k_{r+1}}\right) h_{ir},
\]

\[
h_{ij} := h_{ij} - \left(\frac{w_{r+1,i}}{k_{r+1}}\right) h_{ir} \quad (j = r + 2, \ldots, n),
\]

where \(k_{r+1} = 2^{\beta_{r+1}}\);

(iii) For each value of \(i\) from 1 to \(r\) add \(\alpha_{i,r+1} \times 2^{\beta_{r+1}}\) to \(h_{i+1,r+1}\), namely

\[
h_{2,r+1} := h_{2,r+1} + 2^{\beta_2 - \beta_{r+1}} h_{1r}, \ldots,
\]

\[
h_{i,r+1} := h_{i,r+1} + 2^{\beta_i - \beta_{r+1}} h_{i-1,r}, \ldots,
\]

\[
h_{r,r+1} := h_{r,r+1} + 2^{\beta_r - \beta_{r+1}} h_{r-1,r},
\]

\[
h_{r+1,r+1} := h_{r+1,r+1} + h_{rr},
\]

In the computations of steps (ii) and (iii), the multipliers of \(h_{ir}\) are always located in the interval \([-1, 1]\) because of (8), so that the reduction is numerically stable.

**Theorem 2**  The reduction from the Hessenberg matrix (3) to the Frobenius-like form (2) is numerically stable.

The final Frobenius-like matrix is given by (2), here \(O(n^3)\) flops are required.

### 3. Quasi-Routh array of Frobenius-like matrix

To determine polynomial stability, that is to determine whether the real-part of all the roots of polynomial are less than zero, the Routh array has been already used over 100 years. From the viewpoint of numerical computation, the Routh array does subtractions by directly using products of two coefficients. And only the Frobenius form of a matrix can produce coefficients of the eigenpolynomial. This makes no scaling strategy use for the Routh array. No attention has been paid to the Frobenius form for matrix stability for many years.

Fortunately, the quasi-Routh array for determining polynomial stability can change this situation. We first notice
\textbf{Theorem 3} The stability of eigenpolynomial \((-1)^{n-1}\text{det}(A - \lambda I)\) may be determined by \((n - 2)\) ratios (determining coefficients)

\[
\begin{align*}
\alpha_1 &= \frac{q_0 q_3}{q_1 q_2} = \frac{2^{\beta_3} p_0 p_3}{2^{\beta_3} p_1 p_2}, \quad \alpha_2 = \frac{q_1 q_4}{q_2 q_3} = \frac{2^{\beta_4} p_1 p_4}{2^{\beta_3} p_2 p_3}, \ldots, \\
\alpha_i &= \frac{q_{i-1} q_{i+2}}{q_i q_{i+1}} = \frac{2^{\beta_{i+1}} p_{i-1} p_{i+2}}{2^{\beta_{i+1}} p_i p_{i+1}}, \ldots, \quad \alpha_{n-2} = \frac{q_{n-3} q_n}{q_{n-2} q_{n-1}} = \frac{2^{\beta_{n-2}} p_{n-3} p_n}{p_{n-2} p_{n-1}} = \frac{-2^{\beta_{n-1}} p_{n-3}}{p_{n-2} p_{n-1}}.
\end{align*}
\]

where \(A\) is the matrix (1), \(I\) denotes an identity matrix, \(q_i\) and \(p_i\) are the elements of the Frobenius form and Frobenius-like form, respectively. If matrix \(A\) is stable, then \(p_i < 0\) \((i = 0, \ldots, n - 1)\) and \(0 < \alpha_i < 1\) \((i = 1, \ldots, n - 2)\). And, if \(p_i < 0\) \((i = 0, \ldots, n - 1)\) and \(0 < \alpha_i < 0.46557\) \((i = 1, \ldots, n - 2)\), then \(A\) is stable. For \(n = 3\), matrix \(A\) is stable if and only if \(p_i < 0\) \((i = 0, 1, 2)\) and \(0 < \alpha_1 < 1\).

\textbf{Proof} See [6].

Theorems 2 and 3 show that the numerically stable Algorithm 1 may be used for determining the stability of the Hessenberg matrix.

The quasi-Routh array of (2) comes essentially from the \((n - 2)\) ratios (9), see [7]. For odd-number \(n = 2m + 1\) case \((m \geq 1)\), the quasi-Routh array \(\{r_{i,j}\}_{i=0, j=0}^{m-1, m-i}\) is given by

\[
r_{0,j} = \frac{2^{\beta_3} p_0 p_{2j+1}}{2^{\beta_{2j+1}} p_1 p_{2j+1}} (j = 0, \ldots, m, p_{2m+1} = p_n := -1, \beta_{2m+2} = \beta_{n+1} := 0),
\]

\[
r_{1,j} = \frac{2^{\beta_3 + \beta_4} p_0 p_{2j+2}(r_{0,0} - r_{0,j+1})}{2^{\beta_{2j+2}} p_2 p_{2j+2} p_2 p_{2j+2} (r_{0,j} - r_{0,m})} (j = 0, \ldots, m - 1),
\]

\[
r_{i,j} = \frac{r_{i-1,j}(r_{i-1,0} - r_{i-1,j+1})(r_{i-2,0} - r_{i-2,m-i+2})}{(r_{i-2,0} - r_{i-2,j+1})(r_{i-1,j} - r_{i-1,m-i+1})} (i = 2, \ldots, m - 1; j = 0, \ldots, m - i).
\]

For even-number \(n = 2m + 2\) case \((m \geq 1)\), the quasi-Routh array \(\{r_{i,j}\}_{i=0, j=0}^{m, m-i+1}\) is given by

\[
r_{0,j} = \frac{2^{\beta_3} p_0 p_{2j+1}}{2^{\beta_{2j+1}} p_1 p_{2j}} (j = 0, \ldots, m, r_{0,m+1} := 0),
\]

\[
r_{1,j} = \frac{2^{\beta_3 + \beta_4} p_1 p_{2j+2}(r_{0,0} - r_{0,j+1})}{2^{\beta_{2j+1}} p_2 p_{2j+1} (r_{0,0} - r_{0,1})} (j = 0, \ldots, m, p_{2m+1} = p_n := -1, \beta_{2m+3} = \beta_{n+1} := 0),
\]

\[
r_{2,j} = \frac{2^{\beta_3 + \beta_4} p_1 p_{2j+3}(r_{1,0} - r_{1,j+1})}{2^{\beta_{2j+3}} p_3 p_{2j+1} (r_{1,1} - r_{1,m})} (j = 0, \ldots, m - 1),
\]

\[
r_{i,j} = \frac{(r_{i-1,0} - r_{i-1,j+1})(r_{i-2,0} - r_{i-2,m-i+3})}{(r_{i-2,0} - r_{i-2,j+1})(r_{i-1,j} - r_{i-1,m-i+2})} (i = 3, \ldots, m; j = 0, \ldots, m - i + 1).
\]

\textbf{Theorem 4} Suppose (2) is the Frobenius-like form of the Hessenberg matrix (1) and \(p_i < 0\) \((i = 0, \ldots, n - 1)\). Then (1) is stable if and only if

\[
1 \geq r_{i,0} > \cdots > r_{i,m-i} > 0 \quad (i = 0, \ldots, m - 1)
\]

for \(n = 2m + 1\), or

\[
1 \geq r_{i,0} > \cdots > r_{i,m-i+1} > 0 \quad (i = 0, \ldots, m)
\]

for \(n = 2m + 2\), where \(r_{i,j}\) are computed by equation (10) or (11). If \(r_{i,j+1}/r_{i,j} \geq 1\) or \(r_{i,j+1}/r_{i,j} \leq 0\) for a couple of \((i, j)\), then (1) is unstable. And if \(0 < r_{i,j+1}/r_{i,j} \leq 0.46557\) for a couple of \((i, i + 1)\) and all \(j\), namely for all the elements of two successive rows, then (1) is stable.
Proof. See [6] and [7].

Since \( r_{0,0} = 1 \), Theorems 3 and 4 will ensure that almost all the arithmetic operations of the quasi-Routh array may take place in the interval \((0, 1)\). The computations of (10) or (11) have adopted pivoting strategy, so that the process completing the quasi-Routh array is numerical stable.

Each \( r_{i,j} \) in (10) or (11) requires \( 8 \) flops at most, so the quasi-Routh array requires \((n - 1)(n + 3)\) or \((n - 2)(n + 6)\) flops at most. In fact, Theorem 4 shows that, in order to determine stability of the matrix (1), it is not necessary to complete the whole quasi-Routh array for most cases. Even without the quasi-Routh array, the ratios (9) can be used for determining many unstable or 'strongly' stable matrices.

Using the error analysis, we may come to the conclusion that the Frobenius-like form and quasi-Routh array can determine the stability of the Hessenberg matrix with high accuracy.

4. Origin shifts of the Frobenius-like matrix

In order to find the eigenvalues with the largest real-part, we will perform serial origin-shifted transformations with the form \( F_i - s I \) and their stability determinations on the Frobenius-like form (2) of the matrix (1), where \( s \) is a real-number and \( I \) is an identity matrix. If \( F_i - (s_r + \epsilon)I \) is stable while \( F_i - (s_r - \epsilon)I \) is unstable, where \( \epsilon > 0 \), then the largest real-part of eigenvalues of \( F_i \) is \( s_r \) and the accuracy of \( s_r \), referred to as shifted-accuracy, is \( \epsilon \). This accuracy is also considered the shifted-accuracy of eigenvalues with the largest real-part because finding imaginary-parts of the eigenvalue are sufficiently accurate in our algorithm, see [8].

Using Theorem 1, the matrix \( F_i - s I \) is similar to its scaling-matrix given by

\[
\tilde{F}_i = \begin{pmatrix}
-s & 0 & \cdots & 0 & 0 & \cdots & 0 & w_{1,n} \\
2^\beta_1 s & -s & \cdots & 0 & 0 & \cdots & 0 & w_{2,n} \\
\vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\
2^\beta_i s & \cdots & 0 & \cdots & 0 & \cdots & 0 & w_{i,\alpha} \\
\vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\
-s & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & w_{n-1,n} \\
2^\beta_n s & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & p_{n-1} - s
\end{pmatrix}
\]

where \( w_{i,n} = 2^\beta_{i+1} - \beta_i \cdots + \beta_0 P_{i-1}, i = 1, \ldots, n - 1; \beta_2, \ldots, \beta_n \) are integers which may be defined by the scaling strategy. Using Algorithm 1, \( \tilde{F}_i \) is reduced to the Frobenius-like form, then, the stability of \( \tilde{F}_i \) may be determined with Theorems 2 and 3.

Let \( L \) be the bounds of eigenvalues of (2), for example, see [8],

\[
L = 1 + \max\{2^\beta_2 + \cdots + \beta_n |p_0|, 2^\beta_3 + \cdots + \beta_n |p_1|, \ldots, 2^\beta_n |p_{n-2}|, |p_{n-1}| \}.
\]

And let \( s_0 = 0 \). If \( F_i - s_0 I \) is unstable, then \( s_1 = s_0 + L/2 \), otherwise \( s_1 = s_0 - L/2 \); if \( F_i - s_1 I \) is unstable, then \( s_2 = s_1 + L/2^2 \), otherwise \( s_2 = s_1 - L/2^2 \); \ldots; if \( F_i - s_k I \) is unstable, then \( s_{k+1} = s_k + L/2^{k+1} \), otherwise \( s_{k+1} = s_k - L/2^{k+1} \), \( k \leq r - 1 \). When \( r \) is big enough, \( s_r \) is the largest real-part of eigenvalues exactly or approximately for \( L/2^r \leq \epsilon \). Therefore, we can get the largest real-part of eigenvalues after \( r = \lfloor (\log L - \log \epsilon) / \log 2 \rfloor \) origin-shifted transformations at most, where \( \lfloor y \rfloor \) denotes the least integer larger than or equal to \( y \). \( r \) is only dependent on \( L \) and \( \epsilon \) if the stability is exactly concluded in the shifted-accuracy \( \epsilon \).
Let

\[ F_r = \begin{pmatrix}
0 & 0 & 0 & \cdots & 0 & p_0^r \\
2^{\beta_2^r} & 0 & 0 & \cdots & 0 & p_1^r \\
2^{\beta_3^r} & 0 & \cdots & 0 & p_2^r \\
\vdots & & & & & \vdots \\
2^{\beta_n^r} & & & & & p_{n-1}^r
\end{pmatrix} \]  \quad (13)

be the Frobenius-like form generated from the matrix \( F_l - s_r I \).

If

\[ 2^{\beta_2^r + \cdots + \beta_n^r} | p_0^r | \leq \varepsilon, \quad 2^{\beta_3^r + \cdots + \beta_n^r} | p_1^r | \leq \varepsilon, \quad \ldots, \quad 2^{\beta_{n+1}^r + \cdots + \beta_n^r} | p_u^r | \leq \varepsilon, \]  \quad (14)

then \( s_r \) is \((u + 1)\) multiple real-eigenvalues of the Frobenius-like matrix \((13)\) with the satisfactory-threshold \( \varepsilon \), here \( 0 \leq u \leq n - 1 \), \( \beta_n^r = 0 \), \( \varepsilon \) is a very small positive number; and \((13)\) may be deflated by \((u + 1)\) orders. For real-eigenvalue \( s_r \), the satisfactory-threshold is an expression of \( \det(F_l - s_r I) \approx 0 \).

Assume now that the shifted-accuracy \( \varepsilon \) is small enough and \( 2^{\beta_2^r + \cdots + \beta_n^r} | p_0^r | > \varepsilon \). If \( s_r \) is exactly the largest real-part of eigenvalues, then \( F_l - s_r I \) is called critical-matrix. Usually, of course, \( F_l - s_r I \) is only approximately critical. Suppose \( n = 2m + 1 \) \((m \geq 1)\). According to refs. [7] and [8], there is an integer \( l \geq 0 \) such that

\[ |r_{l,j-1} - r_{l,j}| \leq \varepsilon, \quad (j = 1, \ldots, m - l, \quad l \geq 1), \]  \quad (15)

where \( r_{l,j} \) comes from the quasi-Routh array of the Frobenius-like matrix \((13)\). So, for the largest real-part \( s_r \) of complex-eigenvalue, the shifted-accuracy \( \varepsilon \) is an expression of approximately critical-matrix. The imaginary-part \( \eta \) of the complex-eigenvalue is included in all the negative real-roots of the following polynomial \( b(y) \),

\[ b(y) = 1 + y + \pi_2 y^2 + \pi_2 \pi_4 y^3 + \cdots + \pi_2 \pi_4 \cdots \pi_{2(m-l-1)} y^{m-l}, \]

\[ -y = \left( \tau_0 \tau_1 \cdots \tau_{l-1} \frac{2^{-\beta_2^r - \beta_n^r} p_2^r}{p_0^r} \right) \eta^2, \]  \quad (16)

where,

\[ \pi_2 = \frac{(r_{l-1,0} - r_{l-1,1})(r_{l-1,2} - r_{l-1,m-l+1})}{(r_{l-1,0} - r_{l-1,2})(r_{l-1,1} - r_{l-1,m-l+1})}, \]

\[ \pi_{2k} = \frac{(r_{l-1,0} - r_{l-1,1})(r_{l-1,k+1} - r_{l-1,m-l+1})}{(r_{l-1,0} - r_{l-1,k+1})(r_{l-1,1} - r_{l-1,m-l+1})}, \]

\[ \pi_{2(m-l-1)} = \frac{(r_{l-1,0} - r_{l-1,1})(r_{l-1,m-l} - r_{l-1,m-l+1})}{(r_{l-1,0} - r_{l-1,m-l})(r_{l-1,1} - r_{l-1,m-l+1})}, \]

\[ \tau_0 = \frac{r_{0,1} - r_{0,m}}{r_{0,0} - r_{0,m}}, \]

\[ \tau_1 = \frac{r_{1,1} - r_{1,m-l}}{r_{1,0} - r_{1,m-l}}, \]

\[ \tau_{l-1} = \frac{r_{l-1,1} - r_{l-1,m-l+1}}{r_{l-1,0} - r_{l-1,m-l+1}}. \]

Since \( m - l = 1 \) generally in \((16)\), the imaginary-part \( \eta \) of the complex-eigenvalue with the largest real-part \( s_r \) is as follows,

\[ \eta = \pm \left( \tau_0 \tau_1 \cdots \tau_{l-1} \frac{2^{-\beta_2^r - \beta_n^r} p_2^r}{p_0^r} \right)^{-1/2}, \]  \quad (17)
$r_{i,j}, 2^{-\beta_i - \beta_j} p_2^r$ and $p_0^r$ come from the Frobenius-like matrix (13) and its quasi-Routh array. If $m - l \geq 3$, then we may find all the negative real-roots of the polynomial $b(y)$ in (16) using the following Hessenberg matrix,

$$F_y = \begin{pmatrix}
0 & 0 & 0 & \cdots & 0 & -1 \\
\pi_2^{-1} & 0 & 0 & \cdots & 0 & -\pi_2^{-1} \\
\pi_4^{-1} & 0 & 0 & \cdots & 0 & -\pi_4^{-1} \\
\cdots \\
\pi_{2(m-l-1)}^{-1} & -\pi_{2(m-l-1)}^{-1}
\end{pmatrix}$$

(18)

The eigenvalues of (18) are the roots of $b(y)$. $F - s_r I$ is critical if and only if (18) has only negative real-eigenvalues, see [8]. In order to find all the eigenvalues of (18), we may use the origin-shifted algorithm for (18) naturally.

If $n = 2m + 1$ and $l = 0$ in (16), namely the approximately critical matrix $F_l - s_r I$ satisfies $|1 - r_{0,j}/r_{0,j-1}| \leq \varepsilon$, $(j = 1, \ldots, m)$, then $b(y)$ is given by

$$b(y) = 1 + y + \alpha_2^r y^2 + (\alpha_4^r)! y^3 + \cdots + (\alpha_{2m}^r)! y^m,$$

$$-y = \left(\frac{2^{-\beta_i - \beta_j} p_2^r}{p_0^r}\right) \eta^2,$$

(19)

where $\alpha_{2m-2}^r = \alpha_2^r \alpha_4^r \cdots \alpha_{2m-2}^r$, and $\alpha_i^r$ is generated by (9) and (13).

If $n = 2m + 2$ ($m \geq 1$) and integer $l \geq 2$, then (16) is all right after $r_{i,j}$ ($i \geq 1$) in (11) is regarded as $r_{i-1,j}$ of (10), and (17) is replaced by $\eta = \pm (\tau_0 \tau_1 \cdots \tau_{l-2} 2^{-\beta_i - \beta_j} p_2^r/p_1^r)^{-1/2}$.

If $n = 2m + 2$ and $l = 1$, then $|1 - r_{1,j}/r_{1,j-1}| \leq \varepsilon$, $(j = 1, \ldots, m)$, and (19) is replaced by the following polynomial

$$b(y) = 1 + y + \frac{(r_{0,2}/r_{0,1})(1 - r_{0,1})}{1 - r_{0,2}} y^2 + \frac{(r_{0,2}/r_{0,1})(1 - r_{0,1})}{1 - r_{0,2}} \frac{(r_{0,3}/r_{0,1})(1 - r_{0,1})}{1 - r_{0,3}} y^3$$

$$+ \cdots + \frac{(r_{0,2}/r_{0,1})(1 - r_{0,1})}{1 - r_{0,2}} \cdots \frac{(r_{0,m}/r_{0,1})(1 - r_{0,1})}{1 - r_{0,m}} y^m,$$

$$-y = \frac{2^{-\beta_i - \beta_j} p_1^r}{p_1^r} \eta^2,$$

(20)

where $r_{0,j}$, $p_1^r$, etc. are defined by (11) and (13).

It is possible that $n = 2m + 2$ and $m + 1$ pair of complex-eigenvalues have the same real-part. In this special case, the following inequalities

$$2^{\beta_i + \cdots + \beta_j} |p_1^r| \leq \varepsilon, 2^\beta \cdots 2^\beta_k |p_2^r| \leq \varepsilon, \ldots, 2^{\beta_{n-2} + \beta_k} |p_{n-3}^r| \leq \varepsilon, |p_{n-1}^r| \leq \varepsilon$$

hold for the Frobenius-like form (13), and $F_r$ is changed into

$$F_r^* = \begin{pmatrix}
0 & 0 & 0 & \cdots & 0 & 2^{\beta_k} p_0^r \\
2^{\beta_i + \beta_j} & 0 & 0 & \cdots & 0 & 2^{\beta_k} p_2^r \\
2^{\beta_i + \beta_j} & 0 & 0 & \cdots & 0 & 2^{\beta_k} p_4^r \\
\cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\
2^{\beta_{n-2} + \beta_k} & 2^{\beta_k} p_{n-2}^r
\end{pmatrix}$$

The $m + 1$ negative real-eigenvalues of $F_r^*$ are the square of imaginary-parts.
There is another way to avoid the previous special case. Let, in the Gerschgorin’s disk theorem,
\[
L_m = \min\{|a_{i,i} - |h_i| - \sum_{j=i+1}^{n} |a_{i,j}| : 1 \leq i \leq n\} - 2^k, \quad (k = 0 \text{ or } -1). \tag{21}
\]
The real-part of eigenvalues of (1) is larger than \(L_m \neq 0\). The \(n + 1\) order Hessenberg matrix
\[
A' = \begin{pmatrix}
a_{1,1} & a_{1,2} & a_{1,3} & \cdots & a_{1,n-1} & a_{1,n} & 0 \\
a_{2,1} & a_{2,2} & a_{2,3} & \cdots & a_{2,n-1} & a_{2,n} & 0 \\
a_{3,1} & a_{3,2} & a_{3,3} & \cdots & a_{3,n-1} & a_{3,n} & 0 \\
\vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\
a_{n-1,1} & a_{n-1,2} & a_{n-1,3} & \cdots & a_{n-1,n} & 0 & 0 \\
a_{n,1} & a_{n,2} & a_{n,3} & \cdots & a_{n,n} & 0 & 0 \\
1 & L_m
\end{pmatrix}
\tag{22}
\]
has the all eigenvalues of (1) and a real-eigenvalue \(\lambda = L_m\). And the real-eigenvalue \(\lambda = L_m\),
which may be utilized for checkup of the eigenvalues of (1), is found in last.

Now the above discussions can be summarized as a theorem:

**Theorem 5**  The largest real-part of eigenvalues of the Frobenius-like matrix (2) may be found
by serial origin-shifted transformations and the numerically stable Algorithm 1; the imaginary-parts
of the complex-eigenvalue with the largest real-part are found by (16–20) and usually by (17).

Suppose (13) is a critical-matrix with a pair of purely imaginary eigenvalues (17). To deflate (13)
by order 2, we consider the following similar matrix of (13),
\[
F'_r = \begin{pmatrix}
0 & -2^{-\beta_0} \eta^2 & 0 & \cdots & 0 & p'_0 \\
2^{-\beta_1} & 0 & 0 & \cdots & 0 & p'_1 \\
2^{-\beta_2} & 0 & 0 & \cdots & 0 & p'_2 \\
\vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
2^{-\beta_n} & 0 & 0 & \cdots & 0 & p'_{n-1}
\end{pmatrix}
\tag{23}
\]
In order to get (23), firstly (13) is transformed into its scaling-matrix, then Algorithm 1 is used.
If \(p'_0 = p'_1 = 0\), then (23) may be clearly deflated by order 2. This is true. In fact, we have

**Theorem 6**  If (13) is a critical-matrix with a pair of purely imaginary eigenvalues (17), and
\(p'_0 \neq 0\), then \(p'_0 = p'_1 = 0\) hold in (23).

**Proof**  Consider \(\det(F'_r - i \eta I)\), where \(i \eta\) is a purely imaginary eigenvalue of (13). Clearly,
using (17) and \(p'_0 \neq 0\),
\[
|\det(F'_r - i \eta I)| = 2^{\beta_1 + \cdots + \beta_n} |p'_0 + i \eta 2^{-\beta_1} p'_1|.
\]
Therefore \(p'_0 = p'_1 = 0\) hold since \(\det(F'_r - i \eta I) = 0\). \(\blacksquare\)

The result of Theorem 6 is in theory. However, for practical computation, we can only expect that
\[
\max \left\{ 2^{\beta_1 + \cdots + \beta_n} |p'_0|, 2^{\beta_1 + \cdots + \beta_n} |\eta p'_1| \right\} = \delta, \tag{24}
\]
where \(\delta\) should be a very small positive number which may check up the accuracy of \(s_r\) and \(\eta\). If
\(\delta > \varepsilon\), then let \(\varepsilon := \delta\) and determine a real-eigenvalue by (14). Only when (14) does not hold for
\(u = 0\), namely \(|p'_0| < |p'_0|, s_r \pm i \eta\) is a pair of complex-eigenvalues.

The deflated matrix of (13) is still the Frobenius-like form, but the origin of new coordinate
plane is shifted to \(s_r\). Theorems 5 and 6 may be continually applied to the deflated matrix until
the order of the deflated matrix is less than or equal to 2.
5. Origin-shifted algorithm for matrix eigenvalues

There are two controlling-parameters, shifted-accuracy \( \varepsilon \) and satisfactory-threshold \( \varepsilon \), in our algorithm. These two parameters are related to each other for different real-matrix \( A \). Generally speaking, the shifted-accuracy \( \varepsilon \) is a parameter for stopping the origin shift, and the satisfactory-threshold \( \varepsilon \) is a parameter for allowing the Frobenius-like matrix deflation. If \( \varepsilon \) does not meet \( \varepsilon \), then it is possible that the origin shift has stopped with \( L/2^t \leq \varepsilon \), but the Frobenius-like matrix cannot deflate because neither the condition (14) or (15) holds. Therefore we must regulate \( \varepsilon \) such that it can match \( \varepsilon \), namely, \( \varepsilon \) and \( \varepsilon \) should be automatically selected in the computational process.

At the start, we may select \( \varepsilon \) and \( \varepsilon \) to close in the error bounds \( 2^{-t} \), where \( t \) is mantissa digit of computer, for example, \( \varepsilon = \varepsilon / \min(256, n^2) = c_0 2^{-t} = c_0 2^{-52} \approx c_0 10^{-15.6} \) for \( t = 53 \), \( c_0 = 1 \). When the origin-shifted accuracy reaches \( \varepsilon \), the algorithm automatically magnifies \( \varepsilon \) and \( \varepsilon \) such that the Frobenius-like matrix may be just deflated. The magnified \( \varepsilon \) and \( \varepsilon \) are the optimal parameter matching original \( \varepsilon \).

The bounds \( \pm c_0 2^{-t} \) may be considered zero in Theorems 3 and 4.

It is possible that occasionally the condition (14) or (15) holds but \( L/2^t \gg 2^{-t} \). In this situation, the Frobenius-like matrix may be deflated immediately.

The parameter \( \delta \) in the inequalities (24) may be used for checking up the accuracy of eigenvalues in the determinant. Clearly \( \delta = \varepsilon \) when the eigenvalue is a real number. However, \( \delta \) is not \( \varepsilon \) for the complex-eigenvalue. If \( \delta > \varepsilon \), then we should give a warning message.

The bounds of eigenvalues \( L \) defined by (12) are usually too large for the origin-shifted iterations. Let \( \gamma = 1 \) or other integer in (2). We may roughly determine the axis of opposite stability to \( F_i \) by the origin shifts \( j 2^r \), \( (i = 1, 2, \ldots) \). Suppose \( F_i \) and \( F_i - j 2^r I \) \( (j > 0 \) or \( j < 0 \) have, first, the opposite stability, then let

\[
 s_0' = j 2^r, \quad L = 2^r, \tag{25}
\]

and start the iterations for the largest real-part of eigenvalues of \( F_i - s_0' I \). If \( s_0' > 0 \), then \( F_i - s_0' I \) is stable but \( F_i - (s_0' - L) I \) unstable, otherwise \( F_i - s_0' I \) is unstable but \( F_i - (s_0' + L) I \) stable. In order to find the largest real-part of eigenvalues of \( F_i - s_0' I \), using (25), \( r = \gamma + i_1 \) origin-shifted transformations are required at most.

Notice that the reduction from the Hessenberg matrix \( F_i - s_0' I \) to the Frobenius-like form, using the scaling-matrix \( F_i^{s_0'} \), needs only \( 3.5n(n - 1) \) flops which is much less than the flops of an iterative step in the QR algorithm.

We now describe the origin-shifted algorithm for matrix eigenvalues, Algorithm 2, as follows.

**Step 1** Set up \( n \) \((n \geq 3)\) and a general real-matrix \( A \). Using Householder's transformations, reduce \( A \) to upper Hessenberg form (1). Let \( \varepsilon := \min(256, n^2) c_0 2^{-t} \) and verify \( h_i > \varepsilon \) \((i = 2, \ldots, n)\). Get \( L_m \) in (21), let \( n := n + 1 \), replace (1) by (22). Let \( s_r := 0, n' := n, \ s_k := 2^i \).

**Step 2** Let \( \varepsilon := c_0 2^{-t}, \ \delta := \varepsilon \). \( \varepsilon = \min(256, n^2) \varepsilon \).

**Step 3** If \( n' < n \), then go to Step 5.

**Step 4** Using scaling strategy and Algorithm 1, reduce the upper Hessenberg matrix to the Frobenius-like form (2). If (14) holds, then go to Step 13. If (15) holds, then go to Step 11.

**Step 5** Determine the stability of the Frobenius-like matrix \( F_i \) using Theorems 3 and 4. If (14) holds for \( F_i \), then go to Step 13. If (15) holds for \( F_i \), then go to Step 11.

**Step 6** Roughly determine the axis of opposite stability to \( F_i \), get \( s_0' \) and \( L \) according to (25). Let \( s_r := s_r + s_0', s_k := L \).
Step 7 Regard the Frobenius-like matrix origin-shifted as $F_r$ of (13). If $F_r$ satisfies (14), then go to Step 13. If $F_r$ satisfies (15), then go to Step 11.

Step 8 Let $s_k := s_k/2$. If $s_k \leq \varepsilon$, then go to Step 10.

Step 9 If $F_r$ is stable, then let $s_r := s_r - s_k$, reduce the upper Hessenberg matrix $F_r + s_k I$ to the Frobenius-like form (13), go to Step 7; else let $s_r := s_r + s_k$, reduce the upper Hessenberg matrix $F_r - s_k I$ to the Frobenius-like form (13), go to Step 7.

Step 10 If $F_r$ is stable, then let $s_r := s_r - s_k$ and $\tilde{F}_r = F_r + s_k I$; else let $s_r := s_r + s_k$ and $\tilde{F}_r = F_r - s_k I$. Reduce $\tilde{F}_r$ to the Frobenius-like form (13) and record it. If (15) is not satisfied for the Frobenius-like form, then magnify $\varepsilon$ and $\varepsilon$ such that the Frobenius-like form may be just deflated, and go to Step 11 or Step 13.

Step 11 According to (15), get $l$. (l is usually $m' - 1$ or $m'$ when $n' = 2m' + 1$ or $n' = 2m' + 2$.)

According to (16)-(20), find the imaginary-parts $\pm \eta$ of complex-eigenvalues. According to (24), take $\delta = 2^{p_1^* + \ldots + p_u^*} \max\{|p_0^*|, |2^{-\eta} np_1^*|\}$. Print $n', s_r \pm \eta, \varepsilon, \varepsilon, \delta$. If $\delta \leq \varepsilon$, then go to Step 12; else let $\varepsilon := \delta$ and go to Step 16.

Step 12 Let $n' := n' - 2$ and $\varepsilon := s_k$. Get the Frobenius-like matrix deflated by order 2 from (23). If $n' \leq 2$, then go to 14; else go to Step 2.

Step 13 Let $n' := n' - (u + 1)$ and $\varepsilon := s_k$. Get the Frobenius-like matrix deflated by order $(u+1)$ from (13). Print $n', s_r, u + 1, \varepsilon, \varepsilon, \delta$. If $n' > 2$, then go to Step 2.

Step 14 Get 2 or 1 eigenvalue from order 2 or 1 Frobenius-like matrix. Print $n', s_r$, the eigenvalues.

Step 15 Stop.

Step 16 Write ‘Warning: lower accuracy’. If the Frobenius-like form of $\tilde{F}_r$ satisfies (14) for $u = 0$, namely $|p_0^*| \leq |p_0^*|$, then go to Step 13; else go to Step 12.

Remark 1 In computation, the inequalities in Theorem 4 should be $r_{i,j} > r_{i,j+1} + 2^{-\eta}$, and $h_{i}$ ($2 \leq i \leq n$) of (1) has been supposed to satisfy $h_{i} > \varepsilon$ or $h_{i} < -\varepsilon$.

Remark 2 In Steps 11 and 12, the Frobenius-like matrix is deflated each time by order 2, even if $m - l \geq 3$ in (16).

Remark 3 Algorithm 2 can also find all the eigenvalues of (18).

Correct implementation of the algorithm is dependent upon the determination for stability of the Frobenius-like matrix. If the error Frobenius-like form of some matrix leads to mistaken determination of stability, then the algorithm gets mistaken solution or no solution and may print a lot of mistaken information. Therefore the solution without mistaken information given by Algorithm 2 is fiducial and with high accuracy.

6. Numerical tests

All the numerical tests are completed on a PC. The program is run using MATLAB7.0 under WindowsXP. The order of matrices is from 5 to 35. Our experimental results show that the origin-shifted algorithm, generally, overtops or is comparable to the QR algorithm with respect to accuracy of eigenvalues. It is able to say that Algorithm 2 is fiducial and efficient when the order of general matrix is not over 25.

We will investigate some examples of high-order Hessenberg matrices, where the eigenvalues of each matrix are known. For each example, we find all the eigenvalues using two methods, the origin-shifted algorithm and the MATLAB Function eig(A) (QR method). In result, we suppose that the MATLAB functions cos, sin, log2 and pow2 are exact.
Table 1. Comparative results of two methods for Example 1.

| $|z - z^*|$ | z       | Time | $|z - z^*|$ | z       | Time |
|-----------|---------|------|-----------|---------|------|
| $P_{9}^m$ | 2.687e-6 | -0.000003+1.000000i | 0.859 | $P_{9}^m$ | 7.167e-6 | -0.000007+1.000001i | 0.094 |
| $P_{10}^m$ | 2.176e-14 | -0.809017+1.175571i | 0.391 | $P_{10}^m$ | 8.882e-16 | -1.000000+0.000000i | 0.063 |
| $P_{15}^m$ | 0       | -2.000000+0.000000i | 0.109 | $P_{15}^m$ | 0.160 | -3.839980+0.000000i | 0.188 |

Since the order number of the matrix is too large in the examples, we can only give the maximal error module $\max[|\lambda - \lambda^*|]$ between the computational and exact eigenvalues. And $\max[|\lambda - \lambda^*|] = 0$ means $\max[|\lambda - \lambda^*|] \leq \text{eps} = 2^{-52} = 2.220446e-16$.

**Example 1** Let $A_1, A_2, A_3$ be, respectively, the Frobenius form of $P_9$, $P_{10}$ and $P_{15}$, where

\[
P_9 = -z^9 - z^8 - 4z^7 - 6z^5 + 6z^4 - 4z^3 + 8z^2 - z + 3,
\]
\[
P_{10} = -z^{10} + 1,
\]
\[
P_{15} = -z^{15} - 42z^{14} - 811z^{13} - 9540z^{12} - 76363z^{11} - 440010z^{10} - 1882821z^9
\]
\[
-6083544z^8 - 14941956z^7 - 27854704z^6 - 39032640z^5 - 40321792z^4 - 29690880z^3
\]
\[
-14700544z^2 - 4374528z - 589824,
\]

for which the roots are known, see [8], that is,

\[
z_9 = \pm i, \pm i, \pm i, -1 \pm i^{2^{1/2}}, 1;
\]
\[
z_{10} = \pm 1, \pm \cos \left(\frac{\pi}{5}\right) \pm i \sin \left(\frac{\pi}{5}\right), \pm \cos \left(\frac{2\pi}{5}\right) \pm i \sin \left(\frac{2\pi}{5}\right);
\]
\[
-z_{15} = 1, 1, 1, 1, 1, 2, 2, 3, 3, 4, 4, 4, 4, 4, 4, 4.
\]

Table 1 gives the comparative results on Algorithm 2 and the MATLAB Function eig(A), where three columns give $\max[|z - z^*|]$, the corresponding computational root $z$ and CPU time in second, respectively. $P^o$ and $P^m$ represent the origin-shifted algorithm and the MATLAB Function eig(A) respectively.

It is easy to see that the origin-shifted algorithm overtops or is comparable to the MATLAB Function eig(A) with respect to accuracy of eigenvalues. In particular, the MATLAB Function eig(A) gives a mistaken solution for $P_{15}$.

**Example 2** Let $H_{n}^{**}$ be the following Hessenberg matrix of order $n$,

\[
H_{n}^{**} = \begin{pmatrix}
-1 & 0 & 0 & \cdots & 2^{-27} & 0 \\
2 & -1 & 0 & \cdots & 0 & 2^{-27} \\
2 & -1 & \cdots & 0 & 0 \\
& & \cdots \\
& & & -1 & 0 \\
& & & 2 & -1 \\
\end{pmatrix}
\]

We have known that, see [5], the eigenvalues $\lambda$ satisfy

\[
((\lambda + 1)^{n-1} - 2^{n-28})(\lambda + 1) = 0.
\]

To investigate the cases $n = 19$, $n = 22$ and $n = 25$. 
Table 2. Comparative results of two methods for Example 2.

| $|\lambda - \lambda^*|$ | $\lambda$ | Time | $|\lambda - \lambda^*|$ | $\lambda$ | Time |
|------------------|---------|------|------------------|---------|------|
| $H_{19}^o$       | 1.059e−9 | −0.458325 + 0.454519i | 1.297   | $H_{19}^m$       | 3.494e−13 | −0.646447 + 0.612372i | 0.078   |
| $H_{22}^o$       | 1.096e−7 | −1.739097 + 0.355930i | 1.406   | $H_{22}^m$       | 3.764e−13 | −0.179665 + 0.000000i | 0.203   |
| $H_{25}^o$       | 4.665e−6 | −1.885763 + 0.237338i | 2.344   | $H_{25}^m$       | 2.786e−12 | −1.458502 + 0.794149i | 0.313   |

Table 2 gives the comparative results of the two algorithms, where three columns give $\max(|\lambda - \lambda^*|)$, the corresponding computational eigenvalue and CPU time in second, respectively, $H^o$ and $H^m$ represent the origin-shifted algorithm and the MATLAB Function eig(A), respectively.

In this example, the MATLAB Function eig(A) outperforms the origin-shifted algorithm with respect to accuracy of eigenvalues, although results of the origin-shifted algorithm may be applied. Moreover, the origin-shifted algorithm may print a lot of warning information, for instance, there are three warning messages about instability at medial point of critical interval during the computing process of $H_{25}^{**}$.

The CPU time finding eigenvalues used by the origin-shifted algorithm is a little more than by the MATLAB Function eig(A) because of a large collection and treatment of data for mistaken and warning information.

Example 3  Let $A_{n,k}$ ($k > 1$) be the following matrix,

$$A_{n,k} = \begin{pmatrix}
0 & 0 & 0 & \cdots & 0 & 0 & k \\
1 & 0 & 0 & \cdots & 0 & 0 & 0 \\
0 & 1 & 0 & \cdots & 0 & 0 & 0 \\
& \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\
0 & 0 & 0 & \cdots & 0 & 0 & 0 \\
0 & 0 & 0 & \cdots & 1 & 0 & 0 \\
k & 0 & 0 & \cdots & 0 & 1 & 0
\end{pmatrix}.$$  

Using stabilized elementary transformation, $A_n$ is reduced to Hessenberg matrix,

$$H_{n,k} = \begin{pmatrix}
0 & k & 0 & \cdots & 0 & 0 & 0 \\
k & 0 & 0 & \cdots & 0 & 1 & 0 \\
0 & 1/k & 0 & \cdots & 0 & 0 & 1 \\
& \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\
0 & 0 & 0 & \cdots & 0 & 0 & 0 \\
0 & 0 & 0 & \cdots & 1 & 0 & 0 \\
0 & 0 & 0 & \cdots & 0 & -1/k & 0
\end{pmatrix}.$$  

The eigenvalues $\lambda$ satisfy

$$\lambda^n - k^2\lambda^{n-2} - k = 0.$$  

Set $\lambda = k\zeta$, we have $\zeta^n - \zeta^{n-2} - k^{-1} = 0$. When $k = 32$ and $n > 12$, $\zeta^n - \zeta^{n-2} \approx 0$. Therefore $\lambda = 0, \pm 32$ should be the MATLAB eigenvalues of $A_{n,32}$ ($n > 12$). We investigate the cases $n = 15$, $n = 25$ and $n = 35$.

Table 3 gives the comparative results of the two algorithms, where three columns give $\max(|\lambda - \lambda^*|)$, the corresponding computational eigenvalue and CPU time in second, respectively, $A^o$ and $A^m$ represent the origin-shifted algorithm and the MATLAB Function eig(A), respectively.
Table 3. Comparative results of two methods for Example 3.

|       | $|\lambda - \lambda^*|$ | $\lambda$ | Time | $|\lambda - \lambda^*|$ | $\lambda$ | Time |
|-------|--------------------------|----------|------|--------------------------|----------|------|
| $A_{15}^{p}$ | 0                        | 32       | 0.078| $A_{15}^{m}$ | 0.766    | $-0.766017$| 0.109|
| $A_{25}^{p}$ | 0                        | 32       | 0.203| $A_{25}^{m}$ | 0.860    | $-0.860146$| 0.422|
| $A_{35}^{p}$ | 0                        | 32       | 0.500| $A_{35}^{m}$ | 0.900    | $-0.900326$| 0.531|

The MATLAB Function eig(A) gives mistaken solutions for examples of this kind except $\lambda = \pm 32$. Setting the parameter $\gamma = 4$ in (25), the origin-shifted algorithm gives exact solutions. If $\gamma \neq 4$, then the origin-shifted algorithm gives possibly mistaken solutions or no solution; however, it generally prints a lot of mistaken information which can imply how to modify the parameters $\gamma$, $c_0$, etc.

Acknowledgement

The authors are grateful to Professor C. Li for his actual suggestions.

References