Available at
http://pvamu.edu/aam
Appl. Appl. Math.
ISSN: 1932-9466
Vol. 12, Issue 2 (December 2017), pp. 988 – 1001

Applications and Applied Mathematics:
An International Journal
(AAM)

Numerical Experiments for Finding Roots of the Polynomials in Chebyshev Basis

M. Shams Solary

Department of Mathematics
Payame Noor University
PO Box 19395-3697
Tehran, Iran
shamssolary@pnu.ac.ir, shamssolary@gmail.com

Received: December 17, 2016; Accepted: June 19, 2017

Abstract

Root finding for a function or a polynomial that is smooth on the interval $[a, b]$, but otherwise arbitrary, is done by the following procedure. First, approximate it by a Chebyshev polynomial series. Second, find the zeros of the truncated Chebyshev series. Finding roots of the Chebyshev polynomial is done by eigenvalues of a $n \times n$ matrix such as companion or comrade matrices. There are some methods for finding eigenvalues of these matrices such as companion matrix and chasing procedures. We derive another algorithm by second kind of Chebyshev polynomials. We computed the numerical results of these methods for some special and ill-conditioned polynomials.

Keywords: Chebyshev polynomials; QR algorithm; Colleague matrix

MSC 2010 No.: 15B05, 15A18, 65F15

1. Introduction

Finding roots of a single transcendental equation is still an important problem for numerical analysis courses. There are different numerical methods that try to find and to improve all roots on the special interval. Let $f(x)$ be a smooth function on the interval $x \in [a, b]$, but otherwise arbitrary. The real-valued roots on the interval can always be found by the following procedure.

(i) Expand $f(x)$ as a Chebyshev polynomial series on the interval and truncate for sufficiently large n, more details about large n is described in Boyd (2002).
(ii) Find the roots of the truncated Chebyshev series.

Although some of the polynomials are treated as functions, the roots of an arbitrary polynomial of degree n, when written in the form of a truncated Chebyshev series, coincide with the eigenvalues of a $n \times n$ matrix. The matrix for the monomial basis is called the companion matrix. According to Good (1961) the matrix for the Chebyshev basis is called the colleague matrix. Root finding that leads to a tridiagonal-plus-rank-1 matrix is called comrade matrices. If the linearization is upper Hessenberg plus a rank one matrix, then sometimes confederate matrix is used. The following Hessenberg matrix can be written as the sum of a Hermitian plus low rank matrix. For more details see Section 3 of this paper and Vandebril and Del Corso (2010).

During the past few years many numerical processes for computing eigenvalues of rank structured matrices have been proposed (e.g. see Bini et al. (2005), Delvaux et al. (2006), and Vandebril et al. (2008)). In particular, the QR algorithm received a great deal of this attention. Vandebril and Del Corso (2010) developed a new implicit multishift QR algorithm for Hermitian plus low rank matrices matrices.

We try to use the second kind of Chebyshev polynomials for this work. We compared the results with each other for some special examples such as finding roots of some random polynomials, Wilkinson's polynomial and polynomials with high-order roots.

The sections of the article are as follows: Section 2, Companion matrix methods; Section 3, QR Algorithm for Hermitian plus low rank matrices; Section 4, The colleague matrices for finding roots of some Chebyshev series; Section 5, Numerical experiments; and Section 6, Conclusion.

2. Companion matrix methods

Chebfun, introduced in 2004 by Battles and Trefethen (2014), is a collection of Matlab codes to manipulate functions by resembling symbolic computing. The operations are performed numerically by polynomial representations. The functions are represented by Chebyshev expansions with an accuracy close to machine precision. In Chebfun, each smooth piece is mapped to the interval $[-1, 1]$ and created by a Chebyshev polynomials of the form

$$f_n(x) = \sum_{i=0}^{n} a_i T_i(x), \quad x \in [-1, 1], \tag{1}$$

where $T_i(x) = \cos(i \arccos(x))$. Chebfun computes the coefficients a_i by interpolating the target function f at $n + 1$ Chebyshev points,

$$x_i = \cos \frac{\pi i}{n}, \quad i = 0, 1, \ldots, n.$$

This expansion is well-conditioned and stable. Rootfinding of this function has a basic role in the Chebfun system. Boyd, in Boyd (2002) and Boyd (2014), showed two strategies for transferring a Chebyshev polynomial to a monomial polynomial with series of powers form.

One strategy is to convert a polynomial in Chebyshev form into a monomial polynomial. Then apply a standard polynomial-in-powers-of-x rootfinder. Another strategy is to directly form the colleague matrix. Namely we use the Chebyshev coefficients and compute the eigenvalues of the colleague matrix. As discussed in Boyd and Gally (2007) and Day and Romero (2005), the companion matrix with QR algorithm is well-conditioned and stable. The number of floating point operations (flops) is about $O(10n^3)$ in the Chebyshev case. Boyd and Gally (2007) developed a parity-exploiting reduction for Chebyshev polynomials. Then, in Boyd (2007), Boyd gave a similar treatment for general orthogonal polynomials and did the same for trigonometric polynomials.

We know that the roots of a polynomial in monomial form,

$$f_n(x) = \sum_{i=0}^{n} c_i x^i, \tag{2}$$

are the eigenvalues of the matrix called the "companion matrix". For $n = 6$, the companion matrix is

$$C = \begin{bmatrix} 1\,0\,0\,0\,0 & -\frac{c_1}{c_6} \\ 0\,1\,0\,0\,0 & -\frac{c_2}{c_6} \\ 0\,0\,1\,0\,0 & -\frac{c_3}{c_6} \\ 0\,0\,0\,1\,0 & -\frac{c_4}{c_6} \\ 0\,0\,0\,0\,1 & -\frac{c_5}{c_6} \end{bmatrix}. \tag{3}$$

Boyd, in Boyd (2002), shows that the Chebyshev coefficients c_i can be converted to the power coefficients a_i by a vector-matrix multiplication where the elements of the conversion matrix can be computed by a simple recurrence.

Unfortunately, the condition number for this grows as $(1+\sqrt{2})^n$, which means that this strategy is successful only for rather small n matrix (Gautschi (1979)). The colleague matrix is therefore a valuable alternative.

For the case $n = 6$, the colleague matrix is

$$A = \begin{bmatrix} 0\,\frac{1}{2}\,0\,0\,0 & -\frac{a_0}{2a_6} \\ 1\,0\,\frac{1}{2}\,0\,0 & -\frac{a_1}{2a_6} \\ 0\,\frac{1}{2}\,0\,\frac{1}{2}\,0 & -\frac{a_2}{2a_6} \\ 0\,0\,\frac{1}{2}\,0\,\frac{1}{2} & -\frac{a_3}{2a_6} \\ 0\,0\,0\,\frac{1}{2}\,0 & -\frac{a_4}{2a_6}+\frac{1}{2} \\ 0\,0\,0\,0\,\frac{1}{2} & -\frac{a_5}{2a_6} \end{bmatrix}. \tag{4}$$

The error $|f(x) - f_n(x)|$ grows exponentially as x moves away from this interval for either real or complex x (see Boyd and Gally (2007)). Consequently, the only interesting roots are those that either lie on the canonical interval or are extremely close. However, this matrix can be simply transformed into a Hermitian-plus-rank-1 matrix by applying a diagonal scaling matrix.

3. QR Algorithm for Hermitian plus low rank matrices

In Vandebril and Del Corso (2010), the authors develop a new implicit multishift QR algorithm for Hessenberg matrices that are the sum of a Hermitian plus a low rank correction. Its authors claim the proposed algorithm exploits both the symmetry and the low rank structure to obtain a QR-step involving only $O(n)$ flops instead of the standard $O(n^2)$ flops needed for performing a QR-step on a dense Hessenberg matrix.

The authors of Vandebril and Del Corso (2010) are applying a QR-method directly to a Hermitian plus rank m matrix. They suppose the initial matrix A is written as the sum of a Hermitian plus a rank m matrix, namely $A = S + uv^H$, where u and $v \in C^{n\times m}$. A standard reduction of the matrix A to Hessenberg form costs $O(n^3)$ flops, when one does not exploit the available low rank structure. In particular, if S is a band matrix with bandwidth b, the cost reduces to $O((b+m)n^2)$ for transforming the involved matrix A to Hessenberg form. The reduction to Hessenberg form the eigenvalues since it is a unitary similarity transformation. This gives $H = Q^H AQ = Q^H SQ + Q^H uv^H Q = \hat{S} + \hat{u}\hat{v}^H$. The Hessenberg matrix H can be written as the sum of a Hermitian matrix $\hat{S}$ plus a rank m correction matrix $\hat{u}\hat{v}^H$.

The implementation is based on the Givens-weight representation and unitary factorization to represent the matrix results in the incapability of using the standard deflation approach. This process was developed by an alternative deflation technique (see Vandebril and Del Corso (2010)). In the simple form of this algorithm $m = 1$ and $A = S + uv^H$, a summary of their algorithm is here.

Step 1. Determine the orthogonal transformation $\hat{Q}_1$, such that

$$\hat{Q}_1{}^H (A - \mu I)e_1 = \beta e_1, \ \ \beta = \| (A - \mu I)e_1 \|_2$$

and μ is a suitable shift.

Let $H_1 = \hat{Q}_1{}^H H \hat{Q}_1$, and note that H_1 is not upper Hessenberg form since a bulge is created with the tip in position (3, 1).

Step 2. (Chasing steps) Perform a unitary similarity reduction on H_1 to bring it back to Hessenberg form.

Let $\hat{Q}_c$ (the subscript c refers to the chasing procedure) be such that $\hat{H} = \hat{Q}_c{}^H H \hat{Q}_c = \hat{Q}_c{}^H \hat{Q}_1{}^H H \hat{Q}_1 \hat{Q}_c$ is Hessenberg form. Set $\hat{Q} = \hat{Q}_1 \hat{Q}_c$.

In Vandebril and Del Corso (2010), it is shown that the orthogonal matrix Q_c in the chasing step is the cumulative product of $(n - 2)$ Givens factors. We called this process Givens chasing method.

4. Colleague matrices for finding roots of Chebyshev series

In this section we present another algorithm with two theorems. For our algorithm we try to change some steps of the chasing algorithm in Section 3. Numerical results in Section 5 show this work decreases the errors and saves time. By the results of the last section, we need to write Chebyshev companion matrix by $A = S + uv^H$. For this work we use some definitions and formulae for Chebyshev polynomials,

$$T_0(x) = 1, \quad T_1(x) = x, \quad T_n(\cos\theta) = \cos n\theta, \tag{5}$$

$$U_0(x) = 1, \quad U_1(x) = 2x, \quad U_n(\cos\theta) = \frac{\sin(n+1)\theta}{\sin\theta},$$

Theorem 4.1.

Any Chebyshev polynomial can alternatively be written in the monomial polynomial and vice versa.

Proof:

Let $f_n(x) = a_0T_0(x) + a_1T_1(x) + a_2T_2(x) + \ldots + a_nT_n(x)$. By relations in (5) we have:

$$\begin{aligned} f_n(x) &= (a_0 - a_2 + a_4 - a_6 + \ldots) + (a_1 - 3a_3 + 5a_5 - 7a_7 + \ldots)x \\ &+ 2(a_2 - 4a_4 + 9a_6 - \ldots)x^2 + 4(a_3 - 5a_5 + 14a_7 - \ldots)x^3 \\ &+ 8(a_4 - 6a_6 + 20a_8 - \ldots)x^4 + \ldots + 2^{n-3}(a_{n-2} - na_n)x^{n-2} \\ &+ 2^{n-2}a_{n-1}x^{n-1} + 2^{n-1}a_nx^n. \end{aligned}$$

Now, by transferring the sequence above to matrix form and using induction, we obtain:

$$\underbrace{\begin{bmatrix} 1 & 0 & -1 & 0 & 1 & 0 & -1 & \cdots \\ 0 & 1 & 0 & -3 & 0 & 5 & 0 & \cdots \\ 0 & 0 & 1 & 0 & -4 & 0 & 9 & \cdots \\ 0 & 0 & 0 & 1 & 0 & -5 & 0 & \cdots \\ 0 & 0 & 0 & 0 & 1 & 0 & -6 & \cdots \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & \cdots \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{bmatrix}}_{M} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \\ a_7 \\ \vdots \end{bmatrix} = \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ \frac{c_3}{2} \\ \frac{c_4}{4} \\ \frac{c_5}{8} \\ \frac{c_6}{16} \\ \frac{c_7}{32} \\ \vdots \end{bmatrix}.$$

Generally, $M_{n\times n} = (m_{ij})$ is an upper triangular matrix with the following elements:

$$m_{jj} = 1, \quad j = 1, 2, \ldots, n,$$

$$m_{1j} = 0, \quad j = 2, 4, 6, \ldots,$$

$$m_{1j} = -m_{1,j-2}, \quad j = 3, 5, 7, \ldots,$$

$$m_{2j} = -\text{sign}(m_{2,j-2})(|m_{2,j-2}| + 2), \quad j = 4, 6, 8, \ldots,$$

$$m_{ij} = -\text{sign}(m_{i,j-2})(|m_{i,j-2}| + |m_{i-1,j-1}|), \quad i \geq 3, \quad j = 5, 7, 9, \ldots,$$

and other elements are zero. These relations give us the coefficients in (2). We can use the following matrix and its inverse for transferring Chebyshev polynomial to monomial polynomials and vice versa. ■

By the theorem above we can transfer a monomial polynomial to Chebyshev polynomial. Finding roots of a Chebyshev polynomial is stable and well-conditioned with about $O(n^2)$ flops (see Gautschi (1979) and Shams Solary (2016)).

For finding eigenvalues of matrix A in (4), by the following algorithm in Section 3, we cannot use $A = S + uv^H$, that is, the sum of a Hermitian plus a non-Hermitian low rank correction, because a non-zero element in position $(2, 1)$ makes a non symmetric submatrix in matrix S. So we must choose another way for the bulge chasing.

In Vandebril and Del Corso (2010) it is suggested to use

$$A = S + uv^H + xy^H, \tag{6}$$

since that can be used to remove position $(2, 1)$ of matrix A in (4).

This strategy has good theory but it needs about two times flops rather than $A = S + uv^H$ for chasing.

We try to complete the following algorithm by (6) or the way described in the next theorem. This theorem tries to adopt the Chebyshev series such that the matrix S in $A = S + uv^H$ is symmetric. So we do not need (6) for chasing and we can save time and memory.

Theorem 4.2.

Let $f_n(x) = a_0T_0(x) + a_1T_1(x) + a_2T_2(x) + \ldots + a_nT_n(x)$. Write $f_n(x)$ by a truncated series of Chebyshev polynomials that has a symmetric submatrix in the companion matrix form.

Proof:

By using some properties of Chebyshev polynomials, we have,

$$T_i(x) = \frac{1}{2}U_i(x) - \frac{1}{2}U_{i-2}(x),$$

and

$$T_{-i}(x) = T_i(x), \;\; U_{-i}(x) = -U_{i-2}(x), \;\; U_{-1}(x) = 0,$$

$$S_i(x) = U_i\left(\frac{x}{2}\right).$$

So we have,

$$f_n(x) = b_0S_0(x) + b_1S_1(x) + b_2S_2(x) + \ldots + b_nS_n(x),$$

that,

$$b_j = \frac{1}{2}(a_j - a_{j+2}), \;\; j = 2, \ldots, n-2$$

and,

$$b_{n-1} = \frac{a_{n-1}}{2}, \;\; b_n = \frac{a_n}{2}, \;\; b_1 = a_1 - \frac{1}{2}a_3.$$

Also by induction, we can show

$$U_n(x) + b_{n-1}U_{n-1}(x) + \ldots + b_0 = \begin{vmatrix} 2x & -1 & 0 & \ldots & 0 & 0 \\ -1 & 2x & -1 & \ldots & 0 & 0 \\ & & \ddots & \ddots & \ddots & \\ 0 & 0 & 0 & \ldots & 2x & -1 \\ b_0 & b_1 & b_2 & \ldots & -1 + b_{n-2} & 2x + b_{n-1} \end{vmatrix}. \tag{7}$$

We can see that the characteristic polynomial of the matrix

$$B = \begin{bmatrix} 0 & 1 & 0 & \dots & 0 \\ 1 & 0 & 1 & \dots & 0 \\ & \ddots & \ddots & \ddots & \vdots \\ 0 & \dots & 1 & 0 & 1 \\ -b_0 & -b_1 & \dots & 1 - b_{n-2} & -b_{n-1} \end{bmatrix}, \tag{8}$$

is

$$a(\lambda) = S_n(\lambda) + b_{n-1}S_{n-1}(\lambda) + \dots + b_0. \tag{9}$$

$B^T = \text{transpose}(B)$ is the sum of a Hermitian plus a non-Hermitian low rank correction. ■

Obviously running the Givens chasing algorithm by $B^T = S + uv^H$ saves time and memory in compared with $A = S + uv^H + xy^H$ in (6). S is a tridiagonal matrix with entries $(1, 0, 1)$ on the diagonal band. We called this strategy S-Matrix method (the script S refers to the $S_i(x)$ sequence in Theorem 4.2).

With regards, the complexity of the Givens chasing algorithm is more than finding roots of colleague matrix (companion matrix algorithm) and S-Matrix methods. Numerical examples in the last section show these results. Then we prefer to use of (8) for finding roots of function $f_n(x)$.

In the next section we compare the results of these processes with each other for some special examples, such as random polynomials, Wilkinson's notoriously ill-conditioned polynomial, and polynomials with high-order roots.

5. Numerical experiments

In this section, we try to applied different processes for finding roots of Chebyshev polynomials for some examples in Boyd and Gally (2007). We compared the results of these algorithms with each other in Matlab software.

Example 5.1.

Polynomials with random coefficients. We created an ensemble of polynomials with random coefficients that these coefficients chosen from the uniform distribution on $[-1, 1]$. Then the error is averaged within each ensemble between three different processes. The rootfinding is by the colleague matrix in Section 2, Givens chasing in Section 3, and S-Matrix methods in Section 4.

Chebyshev series converge like a geometric series, so it is experimented with multiplying the random coefficients by a geometrically decreasing factor, $\exp(-qj)$ where $q \geq 0$ is a constant and j is the degree of the Chebyshev polynomial multiplying this factor.

Figure 1 shows that the companion matrix algorithm and S-Matrix method are remarkably accurate. Independent of the decay rate and also of the degree of the polynomial, the maximum error

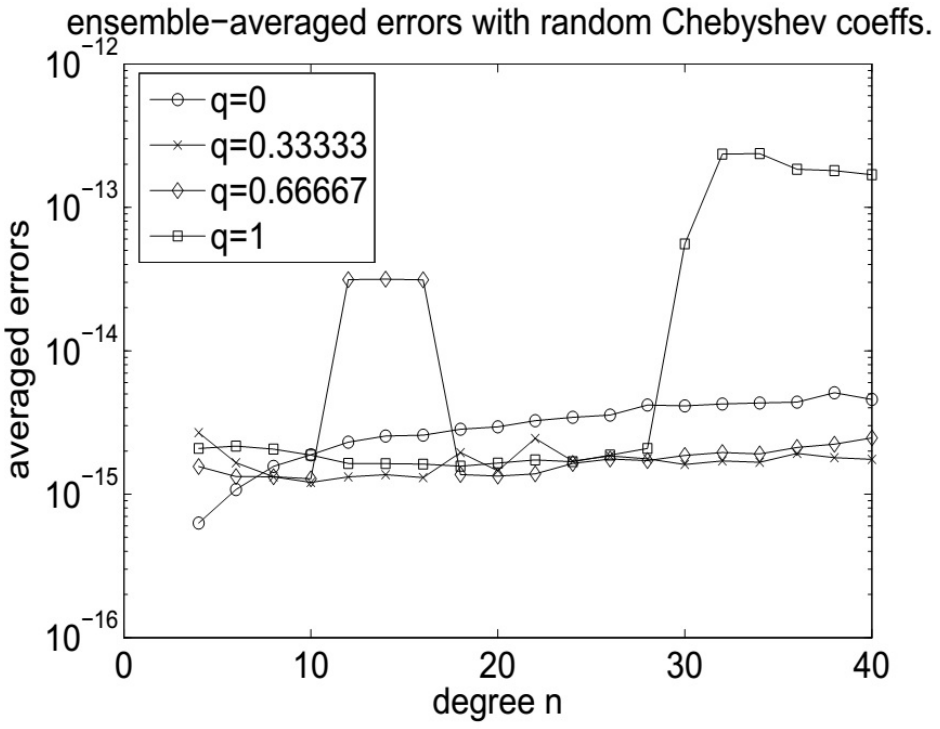

Figure 1. Ensemble-averaged maximum error in the roots for four different decay rates: $q = 0$ (no decay) (circles), $q = \frac{1}{3}$ (x's), $q = \frac{2}{3}$ (diamonds) and $q = 1$ (squares) by companion matrix and S-Matrix methods. Each ensemble included 150 polynomials.

in any of the roots is on average only one or two order of magnitude greater than machine epsilon, 2.2×10^{-16}.

Unfortunately, while rootfinding is done by Givens chasing method in Matlab for low degree polynomials, we need so much time with great errors or we get a loop that regularly iterate.

Example 5.2.

Wilkinson polynomial. About 50 years ago, Wilkinson showed that a polynomial with a large number of evenly spaced real roots was spectacularly ill-conditioned. This polynomial was discussed by others, (e.g. Bender and Orszag (1978)).

We shift and rescale Wilkinson's example so that the Wilkinson polynomial of degree n has its roots evenly spaced on the canonical interval $x \in [-1, 1]$. This work helps us for our purposes:

$$W(x; n) = \prod_{i=1}^{n} \left(x - \frac{2i - n - 1}{n - 1} \right).$$

One difficulty is that the power coefficients c_j vary by nearly a factor of a billion,

$$W(x; 20) = 0.11 \times 10^{-7} - 0.50 \times 10^{-7} x^2 + 0.33 \times 10^{-3} x^4 - 0.80 \times 10^{-2} x^6 + 0.092 x^8 \qquad (10)$$
$$-0.58x^{10} + 2.09x^{12} - 4.48x^{14} + 5.57x^{16} - 3.68x^{18} + x^{20}.$$

The Chebyshev coefficients exhibit a much smaller range,

$$W(x; 20) = \frac{1}{0.000049} \{-1 - 0.18T_2(x) - 0.12T_4(x) - 0.036T_6(x) + 0.045T_8(x) \qquad (11)$$

$$+0.10T_{10}(x) + 0.12T_{12}(x) + 0.093T_{14}(x) + 0.054T_{16}(x) + 0.021T_{18}(x) + 0.0039T_{20}(x)\}.$$

When an asymptotic form is known, accuracy can be improved by multiplying $W(x)$ by a scaling function, namely,

$$f_n(x) \approx \exp(-n_{\text{Wilkinson}} x^2/2) W(x; n_{\text{Wilkinson}}), \tag{12}$$

so Wilkinson's polynomial becomes essentially a sine function with a very small dynamic range. For more details see Boyd and Gally (2007).

Figure 2 for the Wilkinson polynomial without such scaling shows the errors grow with the degree of the Wilkinson polynomial so that it is not possible to obtain any accuracy at all for $W(x; 60)$. Rootfinding by Givens chasing method on the Wilkinson polynomial without such a scaling gives us a divergent loop.

As just pointed out above and described in Boyd and Gally (2007), accuracy in the Wilkinson polynomial can be improved by an exponential scaling function that multiplies in the Wilkinson polynomial. The scaling works as advertised in Figure 3.

With scaling, the degree n of the Chebyshev interplant, which is the matrix size, must be chosen larger than the degree $n_{\text{Wilkinson}}$ of the Wilkinson polynomial. The method is more costly with scaling than without in different algorithms, at least for the special case that $f(x)$ is a polynomial.

The errors have been plotted against the roots themselves in the colleague matrix and S-Matrix methods. They are shown that without scaling, the errors near $x = \pm 1$ where the Wilkinson polynomial has its largest amplitude, are just as tiny as with the scaling-by-Gaussian-function. However, without the scaling factor in the colleague matrix and S-Matrix methods, the errors of the roots near the origin, where the unscaled polynomial is oscillating between very tiny maxima and minima, are relatively huge. The weak results of this process for Givens chasing method are coincided with scaling or without scaling.

Example 5.3.

Multiple roots. The "power function" $f_{pow}(x; k, x_0) = (x - x_0)^k$ allows us to examine the effects of multiple roots on the accuracy of the companion matrix algorithm. We expanded f_{pow} as a Chebyshev series of degree n and then computed the eigenvalues for various n on the interval $[-1, 1]$.

For the extreme values of $x_0 = 0$ (center of the interval) and $x_0 = 1$ (endpoint), multiple roots are very sensitive because small perturbations will split a k-fold root into a cluster of k simple roots x_k. It is called "multiple-root starburst" (see Boyd and Gally (2007)),

$$f_{pow}(x; k, x_0) - \varepsilon = 0 \rightarrow x_k = x_0 + \varepsilon^{1/k} exp(2i\pi j/k), \quad j = 1, \quad 1, 2, \ldots k. \tag{13}$$

As shown in Figure 4 there are errors for the power function with roots at the end of the canonical interval as functions of the order k of the zeros and of the interval tolerance τ, $|\Re(x)| \leq 1 + \tau, \quad |\Im(x)| \leq \tau$.

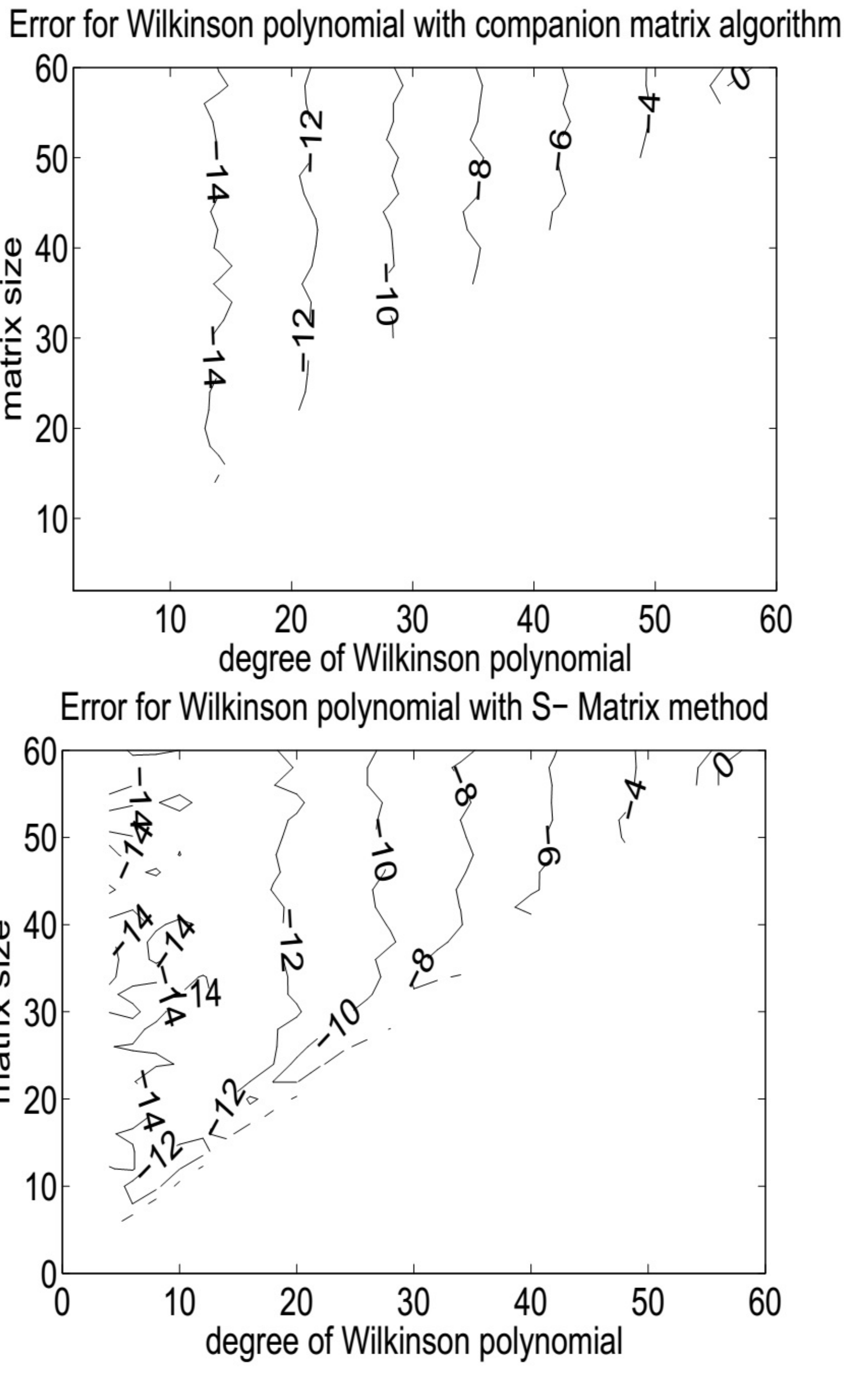

Figure 2. A contour has plotted of the base-10 logarithm of the errors in computing the roots of the Wilkinson polynomial. (The contour labeled -8 thus denotes an absolute error, everywhere along the isolate, of 10^{-8}.) The horizontal axis is the degree of the Wilkinson polynomial; the vertical axis is the degree of the interpolating polynomial and the size of the colleague matrix or S-Matrix, which may be greater than the degree of the Wilkinson polynomial.

The numerical results of different algorithms are similar, with difference in consumed time. The Givens chasing method takes about 10 times more time than other the introduced methods in

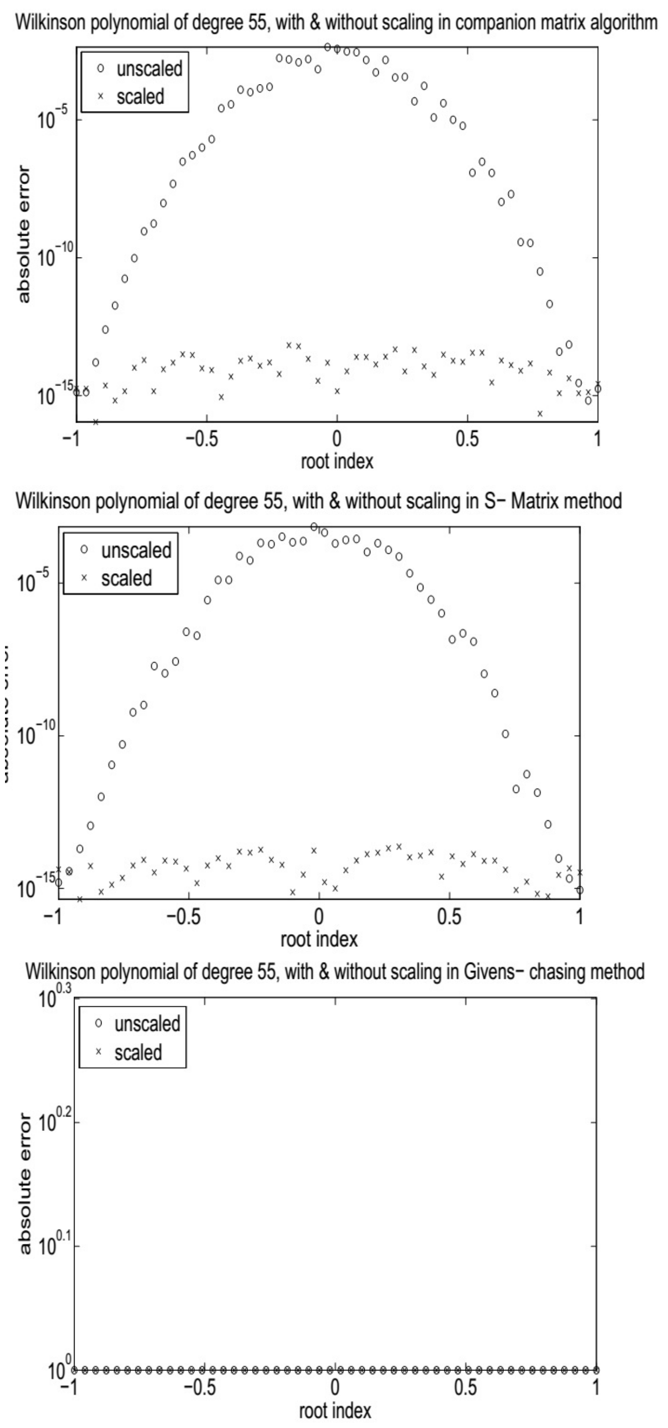

Figure 3. Errors in computing the roots of the Wilkinson polynomial of degree $n_{\text{Wilkinson}} = 55$, with scaling (crosses) and without scaling (circles) in different algorithms.

Sections 2 and 4 .

Figure 1 shows the L_∞ matrix norms in the colleague matrix and S-Matrix in multiple roots. It is important that the size of the matrix elements increases rapidly for high degree Chebyshev coefficients that is created as these matrices are very ill-conditioned. For more examples matrix

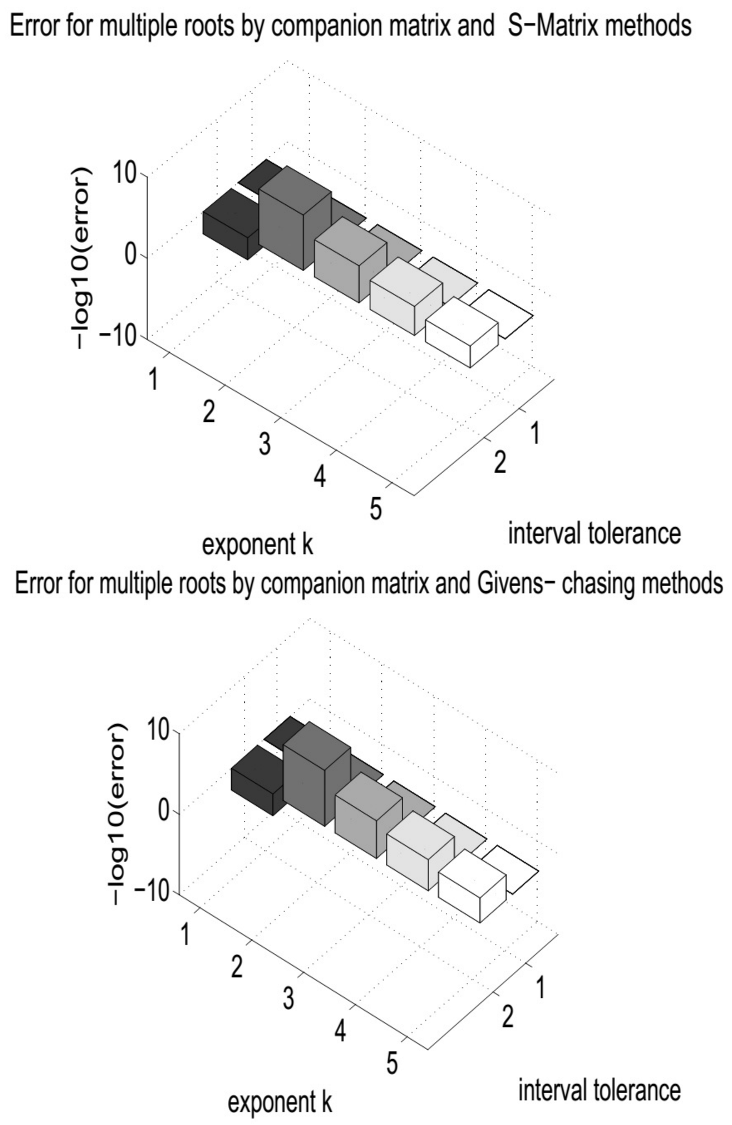

Figure 4. The negative of the base-10 logarithm of the error in computing roots of $(x-1)^k$ for $k = 1$ to 5 for various choices of the interval tolerance τ. The flat squares denote that algorithms failed for those values of k and τ.

norms created by S-Matrix are smaller than the colleague matrix. Finally we can say that the results for the colleague matrix and S-Matrix methods are similar. They designed with a different glance one of them with the first kind Chebyshev polynomials and other with the second kind Chebyshev polynomials.

6. Conclusion

In this article three processes were investigated for finding roots of a polynomial. These processes are the colleague matrix, Givens chasing and S-Matrix methods. We compared the results of these processes with each other for some special examples. Consumed time for the colleague matrix and S-matrix methods are approximately similar, but Givens chasing method needs so much more time for running similar samples with weak results.

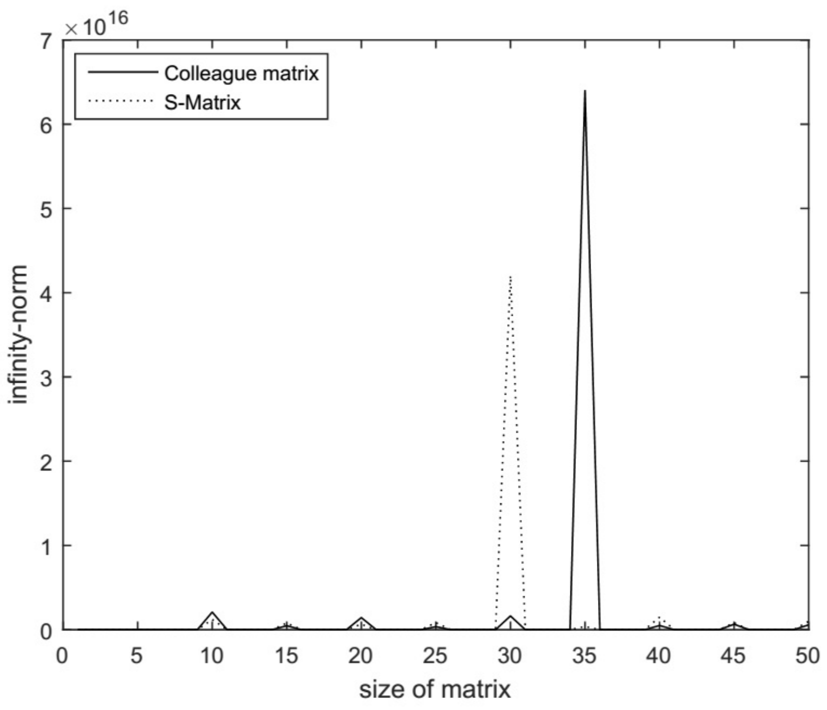

Figure 5. L_∞ matrix norms of created by colleague matrix and S-Matrix with different dimension.

The Givens chasing method shows its efficiency for large n. The numerical experiments presented here show that the colleague matrix and S-Matrix methods are reliable ways for them to find the roots of the first and second Chebyshev polynomials.

Acknowledgement:

The author wishes to thank Professor John P. Boyd in University of Michigan for advise. The author gratefully thanks the authors whose work largely constitutes this sample work such as Professors R. Vandebril and G. M. Del Corso. Also the author is grateful to the referees for constructive comments and suggestions that helped to improve the presentation.

REFERENCES

Bender, C.M., Orszag, S.A. (1978). *Advanced Mathematical Methods for Scientists and Engineers*, McGraw-Hill, New York.

Bini, D. A., Gemignani, L. and Pan, V. Y. (2005). Fast and stable QR eiegenvalue algorithms for generalized companion matrices and secular equations, Numer. Math., Vol. 100, pp. 373–408.

Boyd, J. P. and Gally, D.H. (2007). Numerical experiments on the accuracy of the Chebyshev Frobenius companion matrix method for finding the zeros of a truncated series of Chebyshev polynomials, J. Comput. Appl. Math., Vol. 205, pp. 281–295.

Boyd, J. P. (2007). Computing the zeros of a Fourier series or a Chebyshev series or general orthogonal polynomial series with parity symmetries, Comp. Math. Appl., Vol. 54, pp. 336–349.

Boyd, J. P. (2002). Computing zeros on a real interval through Chebyshev expansion and polynomial rootfinding, SIAM J. Numer. Anal., Vol 40, No. 5, pp. 1666–1682.

Boyd, J. P. (2014). *Solving Transcendental Equations: The Chebyshev Polynomial Proxy and Other Numerical Rootfinders*, Perturbation Series and Oracles, SIAM.

Day, D. and Romero, L. (2005). Roots of polynomials expressed in terms of orthogonal polynomials, SIAM J. Numer. Anal., Vol. 43, No. 5, pp. 1969–1987.

Delvaux, S. and Van Barel, M. (2006). Structures preserved by the QR-algorithm, J. Comput. Appl. Math., Vol. 187, pp. 29–40.

Driscoll, T. A., Hale, N. and Trefethen, L. N. (2014). *Chebfun Guide*, Pafnuty Publications, Oxford.

Gautschi, W. (1979). The condition of polynomials in power form, Mathematics of Computation, Vol. 33, No. 145, pp. 343–352.

Good, I.J. (1961). The colleague matrix, a Chebyshev analogue of the companion matrix, Quart. J. Math., Vol. 12, pp. 61–68.

Shams Solary, M. (2016). Sparse sums with bases of Chebyshev polynomials of the third and fourth kind, Turkish Journal of Mathematics, Vol. 40, pp. 250–271.

Trench, W. F. (1964). An algorithm for the inversion of finite Toeplitz matrices, J. SIAM, Vol. 12, pp. 515–522.

Vandebril, R. and Del Corso, G. M. (2010). An implicit multishift QR-algorithm for Hermitian plus low rank matrices, SIAM J. SCI. COMPUT., Vol. 32, No. 4, pp. 2190–2212.

Vandebril, R., Van Barel, M. and Mastronardi, N. (2008). *Matrix computations and semiseparable matrices*, Volume II: Eigenvalue and singular value methods, The Johns Hopkins University Press, Baltimore, MD.