

Reliability of Numerical Algorithms

Aurél Galántai

Óbuda University
John von Neumann Faculty of Informatics
Institute of Intelligent Engineering Systems

July 1, 2010

Reliability is important in many applications. After more than 60 year development of computer oriented algorithms, mathematical software and floating point arithmetic standards one would assume that at least the key algorithms and their software implementations are reliable on average. But it is not necessarily so.

We assume

- theoretically correct and sound algorithms,
- correct, efficient and standardized software implementations,
- the use of floating point arithmetic standard.

Today's basic/major numerical linear algebra algorithms and their efficient computer implementations have been developed by early 1970s (see, e.g. Wilkinson [38], [39]).

Major steps in the development of numerical linear algebra software "standards":

- LINPACK (J. Dongarra, J. Bunch, C. Moler, and G. Stewart, 1970s and early 1980s)
- BLAS 1-2-3 (since 1979)
- EISPACK (Argonne National Laboratory, Brian Smith, Jack Dongarra, 1972–1973)
- LAPACK (J. Dongarra, 1992, successor to LINPACK and EISPACK)
- ScaLAPACK (J. Dongarra, LAPACK for distributed memory MIMD parallel computers)
- PLAPACK (van de Geijn, Parallel Linear Algebra Package).

Development of error analysis and the floating point arithmetic standard:

- Basic theory of floating point arithmetic and error analysis (Wilkinson [36], 1960s)
- The first standard: IEEE 754-1985
- The new standard: IEEE 754-2008 (see, e.g. Muller, etal. [26])
- Tools for automatic error analysis (see, e.g. Higham [15]).

This lecture reports on two case studies in Matlab, which indicate the nature of problems that may occur [9], [10]. The MATLAB software applies a LAPACK implementation and complies with the floating point standard IEEE 754-1985.

The case studies are related to

- 1 Matrix eigenvalue computation by the QR method at the presence of multiple eigenvalues.
- 2 Acceleration of the Newton method and multiplicity estimate for multiple zeros of polynomials.

The reliability of (implemented) numerical algorithms may depend on various factors such as the nature of the mathematical problem (ill-posedness), the numerical stability, the used arithmetic, etc. This will be shown by the two case studies as well.

The QR method is considered as the best general algorithm for the eigenvalue problem both theoretical and software point of view (see, e.g. Golub, Uhlig [11]). It is built in all major software packages. The method was discovered in 1961 independently by Francis (UK) and Kublanovskaya (USSR) and it is considered as number 6 among the top 10 algorithm of the 20th century (see [6]).

For a matrix $A \in \mathbb{R}^{n \times n}$, the basic QR algorithm has the schematic form:

$$H = U_0^T A U_0 \quad (\text{Hessenberg reduction})$$

for $k = 1, 2, \dots$

 determine a scalar μ

$$H - \mu I = QR \quad (\text{QR-factorization})$$

$$H = RQ + \mu I$$

end

Parameter μ is called shifting, and there are various techniques for choosing it. Apart from peculiar cases, the "QR method" is theoretically convergent for all matrices (see, e.g. [5], [34], [2], [35]).

For simplicity we assume that A is of unreduced upper Hessenberg form. Note that there is one Jordan block to every distinct eigenvalue of an unreduced Hessenberg matrix (see. e.g. [17]).

Example

Matrix

$$T_1 = \begin{bmatrix} 4 & -8 & 0 & 0 \\ 2 & -2 & -4 & 0 \\ 0 & 1 & -6 & 1 \\ 0 & 0 & -16 & 4 \end{bmatrix}$$

has the quadruple eigenvalue 0. Matlab's built in solver (Version 6.1) returns the approximate eigenvalues

$$\begin{aligned} & -4.2928e - 004 \pm 4.2932e - 004i, \\ & 4.2928e - 004 \pm 4.2924e - 004i \end{aligned}$$

which is hardly acceptable. Matlab 5.3 yields similar numbers:

$$\pm 3.827e - 004i, \quad \pm 3.827e - 004.$$

T_1 has a 4×4 Jordan block and one pair of left and right eigenvectors.

If one asks for the right eigenvectors, Matlab returns four vectors. There is no hint that something is going wrong, the user may think, all is in order.

The question arises if the QR method is capable of indicating the presence of a multiple eigenvalue or equivalently: the presence of a not simple Jordan block.

Another question is, why the precision is so bad.

There are two possible explanations.

No. 1

If the perturbation of A is $O(\epsilon_M)$, then (see, e.g. [8]) the eigenvalue perturbations are of the size at most $O(\epsilon_M^{1/m})$, where m is the maximum size of the Jordan blocks in the Jordan normal form of A .

The QR method is backward stable, which means that the QR method computes the exact eigenvalues of a perturbed matrix $A + E$ with $\|E\|_2 \approx \epsilon_M \|A\|_2$ (see Golub, Van Loan [12] or [13]).

Hence the output error is of the size $O(\epsilon_M^{1/m})$.

For double precision $\epsilon_M \approx 10^{-16}$ and for T_1 , $m = 4$ and the output error is $O(10^{-4})$.

No. 2

Consider the problem in the context of polynomial (characteristic) equations and assume that

$$f(x) = (x - x^*)^m g(x), \quad m > 1,$$

where $g(x^*) \neq 0$ ([28], [32], [41]).

There is a neighborhood of x^* - called the *error ball*, where the accurate computation of $f(x)$ is not possible because of cancellation errors. The *error ball* can be approximated by

$$\mathcal{S}_e : |x - x^*| \lesssim \epsilon_M^{1/m} \left| \frac{2^s m! c}{f^{(m)}(x^*)} \right|^{1/m} \leq \epsilon_M^{1/m} C,$$

where C is an appropriate constant.

Hence the attainable precision for zeros with multiplicity m is proportional to $10^{-t/m}$ in t decimal digit precision arithmetic.

For double precision $\epsilon_M \approx 10^{-16}$ and C may be considered close to 1 in the above example, hence the radius of the error ball now is roughly $(10^{-16})^{1/4} = 10^{-4}$ and that is shown by the computed numbers.

Rall [28], Stewart [32], Ypma [41] concluded that

"to get the same accuracy when $m > 1$ as when $m = 1 \dots$ that the precision of arithmetic ... must be increased m -fold".

The example clearly shows the need

- to detect and estimate the multiplicity
- to improve the output of QR method in the presence of multiplicity.

Paper [9] considers the Hyman method and the improvement of QR's output by using Hyman's method in context with the Newton method.

Hyman's method computes the characteristic polynomial and its derivatives in a numerically stable way (Wilkinson [38], [37], Higham [15]). It requires no changes to be introduced in the matrix. Hence input data are not subject to rounding errors.

The Newton method is numerically stable (Lancaster [19], Wozniakowski [40], Spellucci [31]).

The numerical testing was organized along the following guidelines:

- The use of the standard Newton method to improve the eigenvalue approximation given by the QR method;
- The use of Hyman's method to evaluate the characteristic polynomial $f(x)$ and its derivative $f'(x)$;
- Check if the attainable precision argument of Rall, Stewart and Ypma is valid.

Fact

The Newton method converges slowly at the presence of multiple zero. If the multiplicity is known in advance then Newton method

$$x_{k+1} = x_k - m \frac{f(x_k)}{f'(x_k)}$$

has quadratic convergence. The literature has plenty of methods that estimate m (see, e.g. [23], [24]). None of the tested method worked in this context (see Case study 2 later).

The test problems were

$$T_1 = \begin{bmatrix} 4 & -8 & 0 & 0 \\ 2 & -2 & -4 & 0 \\ 0 & 1 & -6 & 1 \\ 0 & 0 & -16 & 4 \end{bmatrix},$$
$$T_2 = \begin{bmatrix} 15 & -92 & 294 & -513 & 459 & -162 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix},$$

$$\lambda_1(T_2) = 1, \lambda_2(T_2) = 2, \lambda_{3,4,5,6}(T_2) = 3,$$

$$T_3 = \begin{bmatrix} -2 & -3 & -4 & \dots & -n+1 & -n & -n-1 \\ 1 & 0 & 0 & \dots & 0 & 0 & 0 \\ 0 & 1 & 0 & \dots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & 0 & 0 & 0 \\ 0 & 0 & 0 & \dots & 1 & 0 & -1 \\ 0 & 0 & 0 & \dots & 0 & 1 & 2 \end{bmatrix} \quad (\lambda = 0),$$

$$T_4 = \begin{bmatrix} -1 & -1 & -1 & \dots & -1 & -1 & -1 \\ 1 & 0 & 0 & \dots & 0 & 0 & 0 \\ 0 & 1 & 0 & \dots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & 0 & 0 & 0 \\ 0 & 0 & 0 & \dots & 1 & 0 & 0 \\ 0 & 0 & 0 & \dots & 0 & 1 & 1 \end{bmatrix} \in \mathbb{R}^{n \times n} \quad (\lambda = 0),$$

$$T_5 = \begin{bmatrix} -1 & -1 & -1 & -1 & -1 + 4a^4 \\ 1 & 0 & 0 & 0 & -4a^4 \\ 0 & 1 & 0 & 0 & -5a^2 \\ 0 & 0 & 1 & 0 & 5a^2 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix}$$

with $\sigma(T_5) = \{0, a, -a, 2a, -2a\}$.

Matrix T_5 is the model of a zero cluster.

Test problems T_1 - T_4 are exactly representable in computer arithmetic. All test problems have no small subdiagonal elements, which otherwise should and can be avoided (see Wilkinson [38]).

Used programs:

- Matlab 7.5 (2007b) in Windows XP environment
- multiple precision package `mptoolbox_1.1` of Ben Barrowes

Source: Mathworks' File Exchange:

`http://www.mathworks.com/matlabcentral/`

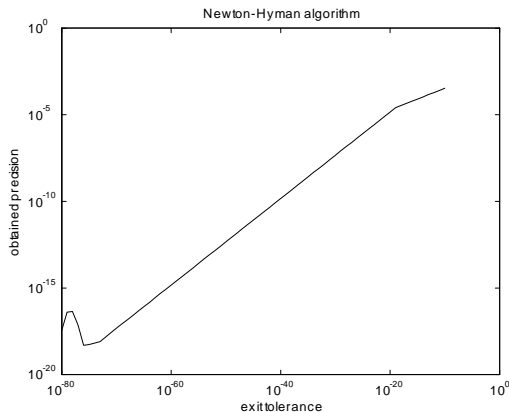
The corresponding parameter in testing:

`precision=number of significant binary digits`

Exit condition of the Newton-Hyman iteration:

$$|x_{n+1} - x_n| < tol \vee |f(x_n)| < tol \vee n > itmax.$$

For T_1 , the maximum error of Matlab's eig routine is $4.5658e - 4$. In higher precision the obtained precision of the Newton-Hyman method versus exit tolerance looks like:



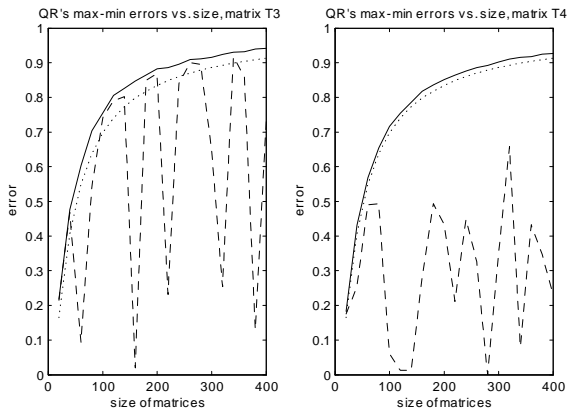
matrix T_1 , precision = 150 \approx 45 decimals

There is a roughly loglinear relationship between the exit tolerance and the obtained precision.

It also indicates, although implicitly, that the attainable precision argument of Rall and Stewart is approximately valid at least in the case of T_1 .

For the single eigenvalues of T_2 Matlab's precision is extremely high and there is no need to improve them.

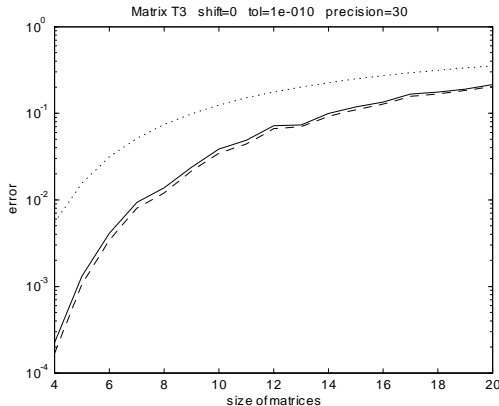
The error distribution of the double precision Matlab eig routine on matrices T_3 and T_4 from $n = 20$ to $n = 400$ is shown on the next figure.



Error distribution of Matlab's eig routine

solid line=maximum error, dashed line=minimum error, dotted line=graph of $\epsilon_M^{1/n}$, where ϵ_M is the machine epsilon and n is the order (size) of the matrix. Indicates that the *attainable precision arguments may be valid in double precision*.

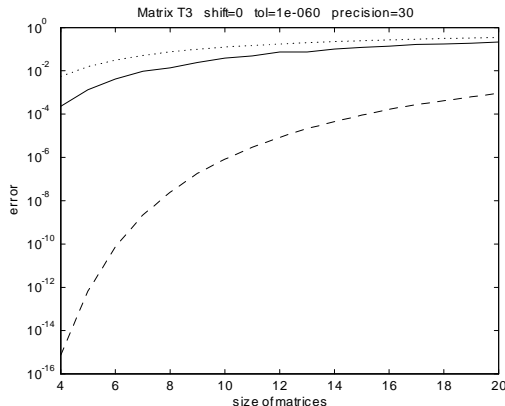
The next figures show the obtained precision for matrix T_3 for various tolerance and precision parameters.



output error, $\text{precision} \approx \text{single precision}$

Dotted line = $2^{-\text{precision}/n}$, solid line = minimum error of Matlab's eig

routine, dashed line=output error of the Newton-Hyman algorithm in less (roughly single) precision.



output error, *precision* \approx single precision

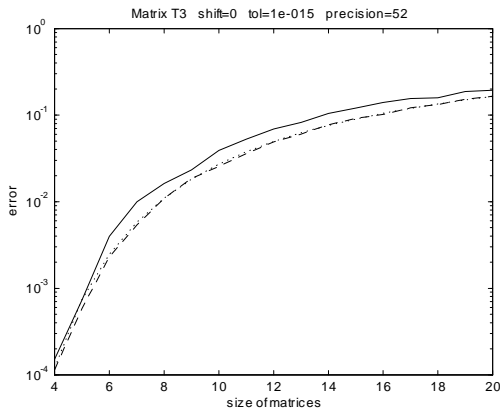
Surprising situation: the single precision Newton-Hyman approach

can significantly improve the double precision output of Matlab's QR algorithm.

The error of the Newton-Hyman method follows a precision rule similar to those of Rall-Stewart-Ypma but it is related to the exit tolerance instead of the arithmetic precision.

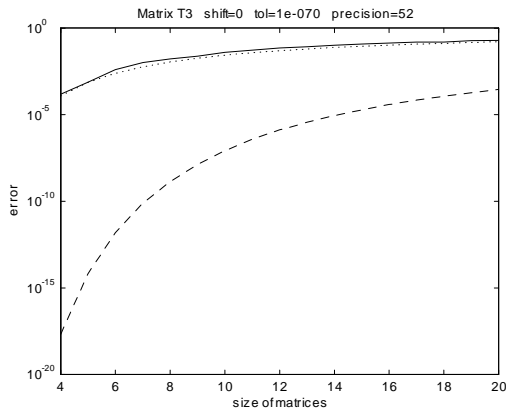
Further increase of the exit tolerance or the maximum iteration number will not give significant improvement.

We can observe the above situation in double precision as well.



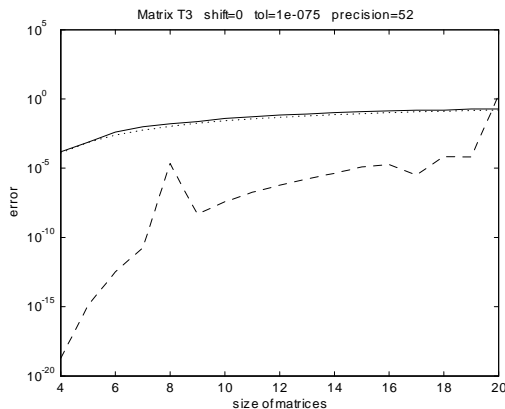
output error, *precision* \approx double precision

Case study 1: Eigenvalue computation



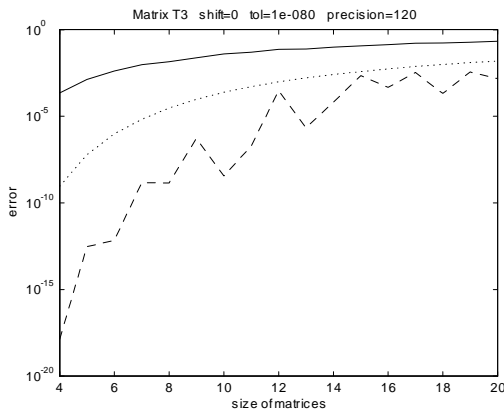
output error, *precision* \approx double precision

Further increase of the exit precision to 10^{-75} results in loss of output precision:



output error, *precision* \approx double precision

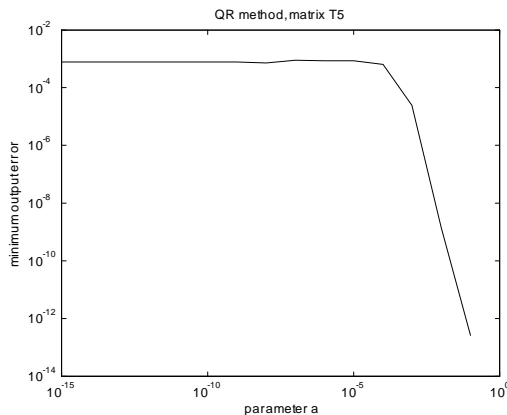
One can think that a simple increase of arithmetic precision solves the problem. The following example shows that it is not the case:



output error, precision ≈ 35 decimal digits

It can be seen that no significant improvement was achieved.

For eigenvalue clusters (matrix T_5) Matlab's QR method gives the following results for $a = 10^{-1}, \dots, 10^{-15}$:



Matlab's output error

Example

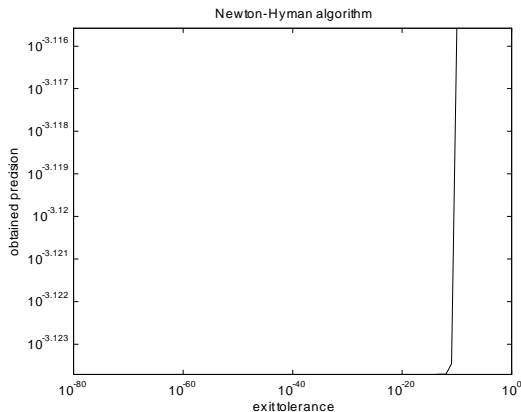
Particularly, for $a = 0.0001$, the Matlab result is

$$10^{-3} [-0.6804 \pm 0.4801i, 0.2599 \pm 0.7770i, 0.8411],$$

which is hardly acceptable when compared to the true values $10^{-3} [\pm 0.1, \pm 0.2, 0]$.

The next figure shows the behavior of the Newton-Hyman method on this problem with parameters $a = 0.0001$, $10^{-80} \leq tol \leq 10^{-10}$ and *precision* = 500.

Case study 1: Eigenvalue computation



Newton-Hyman output for $a = 0.0001$

It is seen that we can not improve the precision of the output of QR method using the investigated Newton-Hyman approach.

Solution

A simple analysis reveals that the denominator of the Newton iteration function is $5\lambda^4 - 15a^2\lambda^2 + 4a^4$ with the four roots

$$\pm a\sqrt{\frac{3}{2} - \frac{\sqrt{145}}{10}}, \quad \pm a\sqrt{\frac{3}{2} + \frac{\sqrt{145}}{10}}.$$

If $a \rightarrow 0$, then these roots also tend to 0, which explains the failure of the Newton method.

Summary of Case study 1

The QR method is not working properly on multiple eigenvalues. The multiple eigenvalues obtained by the QR method can be improved by using the Newton method with Hyman's evaluation process using multiple precision.

The eigenvalue clusters can not be improved in this way.

This investigation is an aftermath of case study 1 and it is about the estimate of multiplicity and acceleration of convergence to a polynomial zero/matrix eigenvalue. For details, see [10].

Assume that

$$f(z) = a_0 z^d + a_1 z^{d-1} + \cdots + a_{d-1} z + a_d \quad (1)$$

is a complex polynomial of degree d ($d \geq 2$).

The Newton method

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} \quad (2)$$

has local quadratic convergence for simple zeros and linear convergence for multiple zeros

Schröder [30]: the modified Newton method

$$x_{n+1} = x_n - m \frac{f(x_n)}{f'(x_n)} \quad (3)$$

has local quadratic convergence if and only if m is the multiplicity of the zero x^* to which $\{x_i\}$ converges (see also [4], [27]).

For known m , formula (3) saves the day. If not, three types of approach are advised.

1st approach:

$$x_{n+1} = x_n - m_n \frac{f(x_n)}{f'(x_n)}, \quad (4)$$

where m_n is the n th estimate of multiplicity based on the computed sequence $\{x_k, f(x_k), f'(x_k), f''(x_k)\}$.

2nd approach: apply the Newton method directly to the $\hat{m} - 1$ -st derivative

$$f^{(\hat{m}-1)}(z) = 0, \quad (5)$$

where \hat{m} is an estimate of the multiplicity.

3rd approach (Schröder, 1870): replaces f by a function g that is constructed from f that x^* is a simple zero of g (for an analysis, see [32], [41]).

McNamee [23], [24] compared several modified Newton methods (computational time and number of function evaluations).

We are interested in *the available precision and reliability*.

Notations:

$$f_n = f(x_n), \quad f'_n = f'(x_n), \quad u = f/f', \quad u_n = u(x_n).$$

Schröder's method:

$$x_{n+1} = x_n - \frac{f_n f'_n}{(f'_n)^2 - f_n f''_n} \left(m_n = \frac{(f'_n)^2}{(f'_n)^2 - f_n f''_n} \right).$$

Traub's method: $m_{est} = \ln |f| / \ln |u|$ and

$$x_{n+1} = x_n - \left(\frac{\ln |f_n|}{\ln |u_n|} \right) u_n.$$

The Aitken-Steffensen acceleration:

$$x_{n+1} = x_n - u_n,$$

$$x_{n+2} = x_{n+1} - u_{n+1},$$

$$x_{n+3} = x_{n+2} - \frac{u_{n+1}^2}{u_n - u_{n+1}}.$$

Ostrowski's algorithm:

$$x_{n+1} = x_n - u_n,$$

$$x_{n+2} = x_{n+1} - \frac{u_n}{u_n - u_{n+1}} u_{n+1}.$$

Rall's method:

$$x_{n+1} = x_n - \frac{u_{n-1}}{u_{n-1} - u_n} u_n \quad \left(m_n = \frac{1}{1 - u_n / u_{n-1}} \right).$$

Madsen's multiplicity estimate [20], [21] relates to line search minimization (see also [22]).

The multiplicity is estimated by the smallest positive integer m_{est} , that satisfies both inequalities

$$|f(x - m_{est} u(x))| \leq |f(x - (m_{est} + 1) u(x))|$$

and

$$|f(x - ju(x))| > |f(x - (j+1) u(x))| \quad (j = 1, \dots, m_{est} - 1).$$

McNamee [23] found that Madsen's method was the fastest among those he tested for speed.

Details of numerical testing:

- 29 test polynomials
- pairwise comparisons between the plain Newton method and the modified Newton methods with a built-in multiplicity estimate.
- testing the multiplicity estimate on the output of the plain Newton method whenever applicable.
- exit condition:

$$|x_{i+1} - x_i| < tol \vee |f(x_i)| < tol \vee i > itmax$$

with parameters: $itmax = 120$,

$tol = 1e - 5, 1e - 10, 1e - 15$.

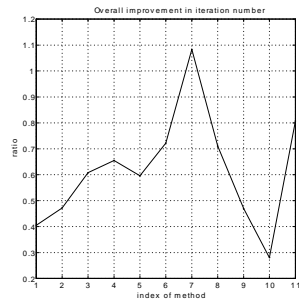
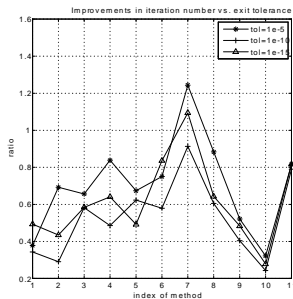
- comparisons only on "solved" test problems. A test problem was "solved", if both the Newton and the modified Newton method terminated in less than or equal to $itmax$ iterations, and the output approximations and multiplicity estimates were finite numbers.

Observed quantities:

- the average iteration numbers (on the "*solved*" problems),
- the number of "*solved*" problems,
- the average error of the multiplicity estimate on the output,
- the number of cases, when the multiplicity estimate gave exact result or the absolute value of its error was less than 0.5.

Tested estimates and methods:

Index	Method
1	Newton-Schröder (in fact Schröder)
2	Newton-Schröder rounded (to the nearest integer)
3	Newton-Rall
4	Newton-Rall rounded (to the nearest integer)
5	Newton-Traub
6	Newton-Traub rounded (to the nearest integer)
7	Newton-Aitken (-Steffensen)
8	Newton-Ostrowski
9	Newton-Hansen-Patrick
10	Newton-Madsen
11	Crouse-Putt-Newton (CPNewt)

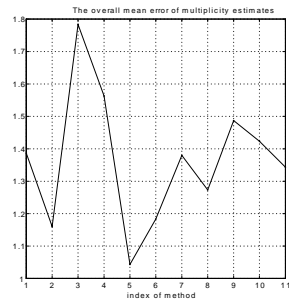
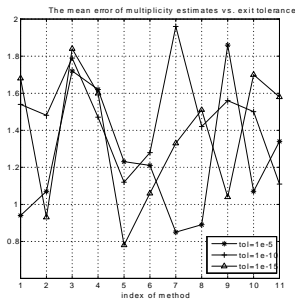


improvement in iterations

Significant improvements in the average iteration number, except for the Newton-Aitken method. The winners are

1. Newton-Madsen (28%)
2. Newton-Schröder (41%)
3. Newton-Hansen-Patrick (47%).

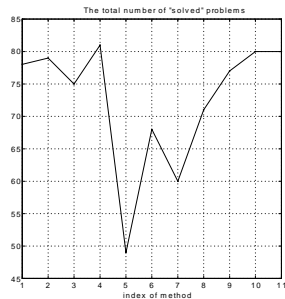
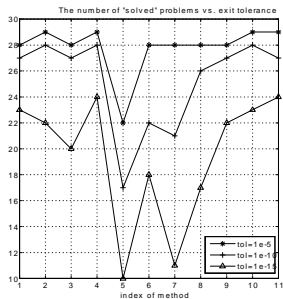
The higher exit tolerance gives smaller iteration numbers although this effect is not unlimited.



mean error of multiplicity estimates

The average error of the multiplicity estimates are generally increasing when applied in the modified Newton framework. The ratio of the mean errors of the modified Newton and the plain Newton methods varies in the interval $(1.04, 1.78)$. The worst case is the Newton-Rall (1.78) , while the best three methods are

1. Newton-Traub (1.04)
2. Newton-Schröder rounded (1.14)
3. Newton-Traub rounded (1.18) .

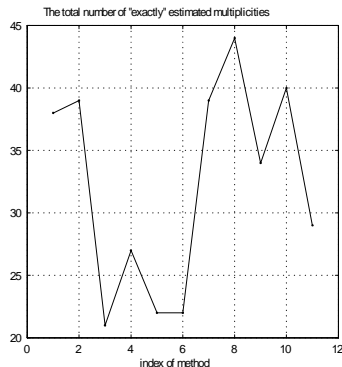


number of "solved" problems

For increasing exit tolerance, the number of solved problems decreases.

The winner are the rounded Newton-Rall method closely followed by the Newton-Madsen and CPNewt methods.

The number of "exactly" estimated multiplicities, although it is related to the previous measure, gives another order:



number of "exactly" estimated multiplicities

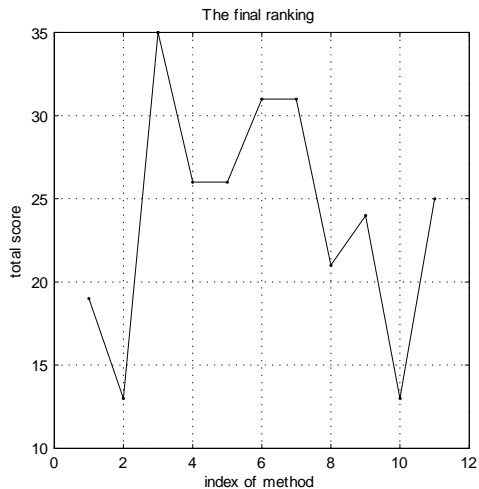
For increasing exit tolerance, the number of "exactly" estimated multiplicities is also decreasing.

The first three methods are Newton-Ostrowski (44),
Newton-Madsen (40) and Newton-Schröder rounded (39).

The final ranking of the tested methods was made by a standard scoring technique.

For each comparison we assigned unique scores from 1 to 11, where the best is 1 and the worst is 11.

The final ranking is given by the sum of individual scores and the best method will be the one with the smallest total score.



final ranking

The overall winners:

1. Newton-Schröder rounded (scores: 13)
2. Newton-Madsen (scores: 13)
3. Newton-Schröder (scores: 19).

Conclusions






The built in multiplicity estimates can significantly decrease the number of Newton iterations.







These multiplicity estimates combined with the Newton method usually give worse multiplicity estimate opposite to their intended purpose.

They cannot be really considered as reliable multiplicity estimators but can be used as useful convergence accelerators.







Problem






If not the multiplicity estimate improves the convergence speed of the plain Newton method, then what else.







-  [1] Anderson, E., et al.: LAPACK Users' Guide, Release 3.0, SIAM, 1999
-  [2] Batterson, S.: Convergence of the shifted QR algorithm on 3×3 normal matrices, Numerische Mathematik, 58, 1990, 341–352.
-  [3] Bauer, F.L: Origins and Foundations of Computing, Springer, 2010
-  [4] Bodewig, E.: On Types of Convergence and Behavior of Approximations in the Neighborhood of a Multiple Root of an Equation. Quart. Appl. Math. 7, 1949, 325–333
-  [5] Buurema, H.J.: A geometric proof of convergence for the QR method. Ph.D. Thesis, Rijksuniversiteit te Groningen, The Netherlands, 1970






-  [6] Cipra, B.A.: The best of the 20th century: editors name top 10 algorithms. SIAM News, 33, 2000, Available at <http://www.siam.org/news/news.php?id=637>
-  [7] Crouse, J.E., Putt, C.W.: Roots of polynomials by ratio of successive derivatives. technical report, NASA TN D-6793, NASA, Washington. (1972)
-  [8] Galántai, A., Hegedűs, C.J., Perturbation bounds for polynomials, Numerische Mathematik, 109, 2008, 77–100.
-  [9] Galántai, A., Hegedűs, C.J.: Hyman's method revisited, JCAM, 226, 2009, 246–258
-  [10] Galántai, A., Hegedűs, C.J.: A study of accelerated Newton methods for multiple polynomial roots, Numerical Algorithms, 54, 2010, 219–243
-  [11] Golub, G., Uhlig, F.: The QR algorithm: 50 years later its genesis by John Francis and Vera Kublanovskaya and







subsequent developments, IMA Journal of Numerical Analysis, 29, 2009, 467–485

-  [12] Golub, G., Van Loan, C.F.: Matrix Computations, The Johns Hopkins University Press, 1983.
-  [13] Golub, G., Van Loan, C.F.: Matrix Computations, 3rd edition, The Johns Hopkins University Press, 1996
-  [14] Hansen, E., Patrick, M.: Estimating the Multiplicity of a Root. Numerische Mathematik 27, 1976, 121–131
-  [15] Higham, N.J.: Accuracy and Stability of Numerical Algorithms, SIAM, 1996.
-  [16] Householder, A.J.: The Theory of Matrices in Numerical Analysis, Blaisdell, New York, 1964.
-  [17] Horn, R., Johnson, C.: Matrix Analysis, Cambridge University Press, 1985.

-  [18] Jenkins, M., Traub, J.: Principles for Testing Polynomial Zerofinding Programs. ACM Transaction on Mathematical Software 1, 1975, 26–34
-  [19] Lancaster, P.: Error analysis for the Newton-Raphson method, Numerische Mathematik, 9, 1966, 55–68.
-  [20] Madsen, K.: A Root-Finding Algorithm Based on Newton's Method. BIT 13, 1973, 71–75
-  [21] Madsen, K., Reid, J.K.: FORTRAN Subroutines for Finding Polynomial Zeros. technical report A.E.R.E. R.7986, Computer Science and System Division, A.E.R.E. Harwell, Oxfordshire (1975)
-  [22] Mathews, J.H.: An improved Newton's method. The AMATYC Review, 10, 1989, 9–14

-  [23] McNamee, J.: A Comparison of Methods for Accelerating Convergence of Newton's Method for Multiple Polynomial Roots, ACM SIGNUM Newsletter, 33(2), 1998, 17–22.
-  [24] McNamee, J.M.: Numerical Methods for Roots of Polynomials, Part I, Elsevier, 2007.
-  [25] Moré J.J., Sorensen, D.C.: Newton's method, technical report, ANL-82-8, Argonne National Laboratory, Argonne, 1982.
-  [26] Muller, J.-M., et al: Handbook of Floating-Point Arithmetic, Birkhäuser, 2010
-  [27] Ostrowski, A.: Solution of Equations and Systems of Equations. Academic Press, 1960
-  [28] Rall, L.: Convergence of the Newton Process to Multiple Solutions, Numerische Mathematik, 9, 1966, 23–37.

-  [29] Rice, J.R.: Matrix Computations and Mathematical Software, McGraw-Hill, 1983
-  [30] Schröder, E.: Über unendlich viele Algorithmen zur Auflösung der Gleichungen. Mathematische Annalen **2**. 317–365 (1870). English translation by G.W. Stewart: On Infinitely Many Algorithms for Solving Equations, technical report TR-92-121, University of Maryland, Department of Computer Science, 1992.
-  [31] Spellucci, P.: An approach to backward analysis for linear and nonlinear iterative methods, Computing, 25, 1980, 269–282.
-  [32] Stewart, G.W.: The behavior of a multiplicity independent root-finding scheme in the presence of error, BIT, 20, 1980, 526–528.
-  [33] Traub, J.: Iterative Methods for the Solution of Equations. Englewood Cliffs, N.J.. Prentice-Hall, Inc., 1964.

-  [34] Watkins, D.S.: Understanding the QR algorithm. SIAM Rev., 24, 1982 427–440
-  [35] Watkins, D.S.: The QR algorithm revisited. SIAM Rev., 50, 2008, 133–145
-  [36] Wilkinson, J.H.: Rounding Errors in Algebraic Processes, Prentice-Hall, Inc., 1963
-  [37] Wilkinson, J.H.: Rounding Errors in Algebraic Processes, Dover, 1994.
-  [38] Wilkinson, J.H.: The Algebraic Eigenvalue Problem, Oxford University Press, 1965
-  [39] Wilkinson, J.H., Reinsch, C. (eds.): Handbook of Automatic Computationm vol. 2: Linear Algebra, Springer, 1971



[40] Wozniakowski, H.: Numerical stability for solving nonlinear equations, *Numerische Mathematik*, 27, 1977, 373–390.



[41] Ypma, T.J: Finding a multiple zero by transformations and Newton-like methods, *SIAM Review*, 25, 1983, 365-378.