99-96

INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

SECOND WORKSHOP NICONET

SECOND NICONET WORKSHOP

LOGICIELS NUMERIQUES D'AUTOMATIQUE :

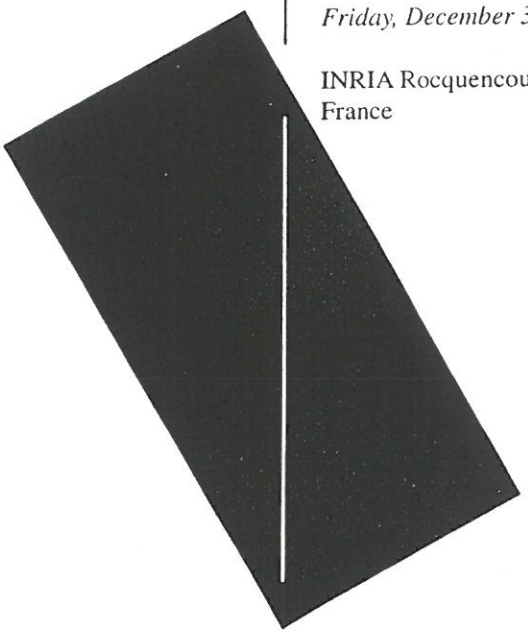
SLICOT, un outil pour l'industrie

NUMERICAL CONTROL SOFTWARE:

SLICOT, a useful tool in industry

Vendredi 3 décembre 1999
Friday, December 3, 1999

INRIA Rocquencourt
France



ACTES /PROCEEDINGS

SECOND NICONET WORKSHOP

NUMERICAL CONTROL SOFTWARE :

SLICOT, a useful tool in industry

Friday, December 3, 1999

INRIA Rocquencourt, France.

Sommaire / Contents

Nonsingular Jacobian Free Piecewise Linearization of Ordinary Differential Equations (ODE). 1
E. Arias Antunez, I. Blanquer Espert, V. Hernandez Garcia and J. Javier Ibanez Gonzalez.

Efficient Balanced Truncation Model Reduction Methods Using the Matrix Sign Function. 7
P. Benner, E.S. Quintana-Orti and G. Quintana-Orti.

Determination of Markov Parameters Directly from Noise-Free Input-Output Samples of MIMO Systems. 11
S. Bingulac and N. F. Al-Muthairi.

Evaluation of Frequency Response Function of Hamonically Excited Dynamic System. 17
R. Chumúrny and J. Stein.

Blocked Algorithms for Reduction of a Regular Matrix Pair to Generalized Schur Form. 21
K. Dackland and B. Kågström.

The Scilab-Slicot Environment. 23
F. Delebecque and S. Steer.

StratiGraph – Computation and Presentation of Graphs Displaying Hierarchies of Jordan and Kronecker Structures. 33
E. Elmroth, P. Johansson and B. Kagström.

Software for Solving Robust Performance Problems Based on Berstein Expansion. 35
J. Garloff.

Convex Optimization Over Positive Polynomials. 41
Y. Genin, Y. Hacher, Y. Nesterov, P. Van Dooren.

Real Stability Radii of Polynomial Matrices. 47
Y. Genin, R. Stefan, P. Van Dooren.

H_∞ Loop Shaping Desing Procedure Routines in SILICOT 53
D.W. Gu, P.Hr. Petkov and M.M. Konstantinov.

Fast Implementation and Stability Property of the QR Factorization in Subspace Identification. 55
N. Mastronardi, S. Van Huffel and P. Van Dooren.

Model Reduction of Large Systems with Slicot.. 59
P.M. Ndiaye and S. Steer.

Comparison of Riccati Equation Solvers in MATLAB and SILICOT. 61
P.Hr. Petkov, D.W. Gu, M.M. Konstantinov and V. Mehrmann.

SILICOT Drives Tractors ! 63
H. Faβbender and P. Benner.

Numerical Software for Redundancy Determination and Presolving Analysis of Large Scale Linear Programming Problems, Using MATLAB 5.2. 67
I. Ioslovich.

Recursive Blocked Algorithms for Solving Triangular Sylvester-Type Matrix Equations. 73
I. Jonsson and B. Kagström.

Cholesky of QR Factorization for Data Compression in Subspace-based Identification ? 75
V. Sima.

Accurate Computation of Eigenvalues and Real Schur Form of 2x2 Real Matrices. 81
V. Sima.

A SILICOT Based Descriptor Systems Toolbox for MATLAB. 87
A. Varga.

99-96

Accurate computation of eigenvalues and real Schur form of 2×2 real matrices*

Vasile Sima†

Abstract

This paper summarizes numerical approaches proposed and/or incorporated in the previous and latest LAPACK releases for the solution of the 2×2 real Schur factorization problem.

1 Introduction

The computation of eigenvalues and real Schur form of a 2×2 matrix is a fundamental issue in numerical linear algebra, because it is required by any practical algorithm for solving the algebraic eigenvalue problem for real nonsymmetric matrices. It is also very important in control systems analysis and design, since most related algorithms are based on a real Schur form reduction. Therefore, an accurate implementation of this basic calculation is very important.

Commonly used codes for this simple mathematical problem are included, e.g., in the well-known packages EISPACK [4] and LAPACK [1]. EISPACK incorporates the corresponding in-line code in its general eigensolver `HQR2`, while LAPACK Library offers the specialized routines `*LANV2` (where `*` is `C`, `D`, `S`, or `Z`). Previous numerical experiments have pointed out a possible lack of relative accuracy of the smallest real eigenvalue computed by the double precision routine `DLANV2`, compared to the EISPACK or Matlab calculations. Although the absolute accuracy (compared with the matrix norm) is very good, certain applications, e.g., pole assignment in control theory, could benefit when high relative accuracy is also obtained.

Example 1.1 [1] This is an example with two real eigenvalues of quite different magnitudes. Matrix a is $a = [160000, -14000; 1, 0]$ (using Matlab notation). The "exact" eigenvalues, computed with Maple with 20 digits floating point representation result from the following Maple statement:

```
> eigenvals(a);
          159999.91249995214839, .087500047851614837715
```

EISPACK computes both eigenvalues to full accuracy, as can be seen from the MATLAB statement:

```
>> eig(a)
ans =
    1.599999124999521e+005
    8.750004785161485e-002
```

The results obtained with `DLANV2` indicate a loss of about 7 decimal digits accuracy in computing the smallest eigenvalue, but practically full accuracy for the large eigenvalue, as shown below:

*This work was supported by the European Community BRITE-EURAM III *Thematic Networks Programme NICONET* (project BRRT-CT97-5040).

†National Institute for Research & Development in Informatics, Bd. Mareşal Averescu, Nr. 8–10, 71316 Bucharest 1 (Romania), Fax: (+40)-1-224.05.39, Phone: (+40)-1-224.07.36, ext. 156, E-mail: `vsima@u3.ici.ro` and Department of Electrical Engineering (ESAT), Katholieke Universiteit Leuven, Kardinaal Mercierlaan 94, B–3001 Leuven–Heverlee (Belgium) Fax: (+32)-(0)-16-32 19 70, Phone: (+32)-(0)-16-32 18 42, E-mail: `Vasile.Sima@esat.kuleuven.ac.be`

[1] Provided by Dr. A. Varga, private communication.

```
RT1R, RT1I =   8.7500047855428420D-02     .0000000000000000D+00
RT2R, RT2I =   1.5999991249995213D+05     .0000000000000000D+00
```

where RT1R, RT1I, RT2R, and RT2I denote the computed real and imaginary parts of the first and second eigenvalues. □

This paper comments on the EISPACK and LAPACK implementations and describes some changes made in the LAPACK library code DLANV2, which improved the accuracy of the smallest computed real eigenvalue.

2 EISPACK Approach

The eigenvalues of 2×2 matrices can be found using simple relations. Let

$$A = \begin{bmatrix} \alpha & \beta \\ \gamma & \delta \end{bmatrix}. \tag{1}$$

Denote $w := \beta\gamma$, $p := (\alpha - \delta)/2$, and let λ_1 and λ_2 be the eigenvalues of the matrix A. As $\lambda_1 + \lambda_2 = \alpha + \delta$ and $\lambda_1\lambda_2 = \alpha\delta - w$, it follows that

$$\{ \lambda_1, \lambda_2 \} = \{ \delta + p \pm (p^2 + w)^{1/2} \}. \tag{2}$$

Complex conjugate roots are obtained if $p^2 + w < 0$. Otherwise, denoting $z := p + \mathrm{sign}(p)(p^2 + w)^{1/2}$, then $\lambda_1 = \delta + z$, and $\lambda_2 = \lambda_1$, if $z = 0$, or $\lambda_2 = \delta - w/z$, if $z \neq 0$.

If the eigenvalues λ_1 and λ_2 are real, then the matrix should be reduced to the real Schur form (RSF), representing the maximum reduction achievable by real arithmetic, exclusively using orthogonal transformations. This is performed by a special triangularization, using a suitably chosen plane rotation. Note that the following transformation should be accomplished:

$$\begin{bmatrix} c & s \\ -s & c \end{bmatrix} \begin{bmatrix} \alpha & \beta \\ \gamma & \delta \end{bmatrix} \begin{bmatrix} c & -s \\ s & c \end{bmatrix} = \begin{bmatrix} \lambda_1 & * \\ 0 & \lambda_2 \end{bmatrix}, \tag{3}$$

where $*$ denotes an element whose value is meaningless in determining the rotation. By zeroing the (2,1) element, we obtain the equation $c^2\gamma - cs(\alpha - \delta) - s^2\beta = 0$, or, equivalently, $c^2\gamma - 2pcs - s^2\beta = 0$. Hence, the ratio c/s must have the value $c/s = (p \pm (p^2 + w)^{1/2})/\gamma$. So, we may choose c and s so that $c/s = z/\gamma$. Taking $c = z/r$, $s = \gamma/r$, where $r = \gamma^2 + z^2$, the condition $c^2 + s^2 = 1$ holds. Equivalently, c and s define the rotation which annihilates the second element of the vector $(z, \gamma)^T$.

3 LAPACK 2.0 Approach

The LAPACK 2.0 routines use a different algorithm, that also brings the 2×2 matrix to a canonical real Schur form. Specifically, this algorithm computes the Schur factorization of a real 2×2 general matrix in a standard form:

$$\begin{bmatrix} \alpha & \beta \\ \gamma & \delta \end{bmatrix} = \begin{bmatrix} c & -s \\ s & c \end{bmatrix} \begin{bmatrix} \widetilde{\alpha} & \widetilde{\beta} \\ \widetilde{\gamma} & \widetilde{\delta} \end{bmatrix} \begin{bmatrix} c & s \\ -s & c \end{bmatrix}, \tag{4}$$

where either $\widetilde{\gamma} = 0$, so that $\widetilde{\alpha}$ and $\widetilde{\delta}$ are real eigenvalues of the matrix (and $|\widetilde{\alpha}| \geq |\widetilde{\delta}|$), or $\widetilde{\alpha} = \widetilde{\delta}$ and $\widetilde{\beta}\widetilde{\gamma} < 0$, so that $\widetilde{\alpha} \pm (\widetilde{\beta}\widetilde{\gamma})^{1/2}$ are complex conjugate eigenvalues.

If $\gamma = 0$, then $c = 1$, and $s = 0$, and the given values are not modified. If $\beta = 0$, then $c = 0$, and $s = 1$; so, $\widetilde{\alpha} = \delta$, $\widetilde{\delta} = \alpha$, $\widetilde{\beta} = -\gamma$, $\widetilde{\gamma} = 0$. Otherwise, if the given matrix is not in standardized form, a plane rotation is first chosen which makes the diagonal elements equal. Denoting $t := \alpha - \delta$, $\sigma := \beta + \gamma$, from (4) with $\widetilde{\delta} = \widetilde{\alpha}$, it follows that c and s should satisfy

$$tc + \sigma s + (\widetilde{\beta} + \widetilde{\gamma})s = 0, \qquad \sigma c - ts - (\widetilde{\beta} + \widetilde{\gamma})c = 0.$$

These equations give the "stable" formula $c/s = -(\sigma + \text{sign}(\sigma)\tau)/t$, where $\tau := (\sigma^2 + t^2)^{1/2}$. Using $c^2 + s^2 = 1$, the definitions for c and s in Algorithm `Schur2` below, essentially the same as in [3], can be obtained. Finally, if $\widetilde{\beta}\widetilde{\gamma} > 0$ in the transformed matrix, then another plane rotation, with parameters $\widetilde{c}$ and $\widetilde{s}$, is chosen to annihilate the $(2,1)$ element. The relations for $\widetilde{c}$ and $\widetilde{s}$ follow by specializing those described previously (see (3)), for $\alpha = \delta$.

Algorithm Schur2 : $\boxed{(\alpha, \beta, \gamma, \delta, c, s, \lambda_1, \lambda_2) = \texttt{Schur2}(\ \alpha, \beta, \gamma, \delta\ )}$

/* Compute the canonical real Schur form and the eigenvalues */
/* of a real 2×2 general matrix. */

IF $\gamma = 0$, THEN $c = 1$; $s = 0$; GO TO 3;
IF $\beta = 0$, THEN
  $c = 0$; $s = 1$; $\alpha \leftrightarrow \delta$; $\beta = -\gamma$; $\gamma = 0$; GO TO 3;
IF $\alpha = \delta$ AND $\beta\gamma < 0$, THEN GO TO 3;

/* 1. Equate the diagonal elements. */

$t = \alpha - \delta$; $p = t/2$; $\sigma = \beta + \gamma$; $\tau = (\sigma^2 + t^2)^{1/2}$;
$c = ((1 + |\sigma|/\tau)/2)^{1/2}$; $s = -\text{sign}(\sigma)\, p/(\tau c)$;

$$\begin{bmatrix} \alpha & \beta \\ \gamma & \delta \end{bmatrix} \leftarrow \begin{bmatrix} c & s \\ -s & c \end{bmatrix} \begin{bmatrix} \alpha & \beta \\ \gamma & \delta \end{bmatrix} \begin{bmatrix} c & -s \\ s & c \end{bmatrix};$$

$m = (\alpha + \delta)/2$; $\alpha = m$; $\delta = m$;
IF $\gamma \neq 0$, THEN
  IF $\beta \neq 0$, THEN
    IF $\beta\gamma > 0$, THEN

/* 2. Triangularize a matrix with real eigenvalues. */

      $p = \text{sign}(\gamma)\,(\beta\gamma)^{1/2}$; $\tau = 1/|\beta + \gamma|^{1/2}$;
      $\widetilde{c} = |\beta|^{1/2}\tau$; $\widetilde{s} = |\gamma|^{1/2}\tau$;
      $\alpha = m + p$; $\delta = m - p$; $\beta \leftarrow \beta - \gamma$; $\gamma = 0$;
      $t = c\widetilde{c} - s\widetilde{s}$; $s \leftarrow c\widetilde{s} + s\widetilde{c}$; $c = t$;
    END IF
  ELSE
    $\beta = -\gamma$; $\gamma = 0$; $t = c$; $c = -s$; $s = t$;
  END IF
END IF

/* 3. Set eigenvalues (real, if $\gamma = 0$). */

$\tau = |\beta\gamma|^{1/2}$; $\lambda_1 = \alpha + j\tau$; $\lambda_2 = \delta - j\tau$;

No doubt, the transformations just described are useless if only eigenvalues are to be found, but they are essential in the computation of Schur vectors, and eigenvectors. For real eigenvalues of highly different magnitudes, the LAPACK 2.0 approach could lead to catastrophic cancellation, because the need to restore a small eigenvalue from two comparatively large, equal diagonal elements. Depending on the sign of m and p, defined in Step 1 and Step 2, respectively, of Algorithm `Schur2`, either $\alpha = m + p$; or $\delta = m - p$; can produce a bad result (due to cancellation errors).

4 Modified LAPACK Approaches

This section describes some changes made by the author in the `DLANV2` code, Release 2.0, which improved the accuracy of the smallest eigenvalue for matrices with real eigenvalues. Two different approaches have been tried.

Approach A. The first approach tried has lead to a new implementation, called here `DLANV2A`, with the following main features:

1. It usually behaves like `DLANV2` 2.0 code, but, for matrices with two real eigenvalues, the possible lack of accuracy by cancellation errors is detected, and one of three alternate formulas is used for computing the smallest eigenvalue. The computations essentially follow the previously

implemented algorithm, but some new tests are added, and some calculations are simplified. The stage of equating the diagonal elements is preserved for real eigenvalues too. Cancellation errors are still possible when forming t (if α is very close to δ), and σ (if β is very close to $-\gamma$), at the beginning of Step 1 of Algorithm `Schur2`. Note that cancellation errors could also appear in the EISPACK approach when computing p (if α is very close to δ), and $p^2 + w$.

2. For matrices with two real eigenvalues (which must be reduced to a Schur form), the eigenvalue returned in `RT1R`, `RT1I` satisfies indeed the inequality `ABS( RT1R ) >= ABS( RT2R )`, as stated in the documenting comments for `DLANV2` version, but which was not ensured by `DLANV2` 2.0. Indeed, 5,528 test matrices built by the LAPACK `DGET33` test routine for `DLANV2` (out of 20,736 matrices), did not satisfy that inequality, but all of them did for the `DLANV2A` code.

The strategy used in `DLANV2A` routine may be summarized as follows: • Use the LAPACK scheme whenever possible. • Use the simplest formula avoiding cancellation errors as much as possible. • Try an alternate formula, when a simpler formula does not work safely. Details are given below. After Step 1 of Algorithm `Schur2`, the elements of the transformed matrix, $\widehat{A} = \begin{bmatrix} \widehat{\alpha} & \widehat{\beta}; & \widehat{\gamma} & \widehat{\alpha} \end{bmatrix}$, satisfy $(\lambda_1 + \lambda_2)/2 = \widehat{\alpha} = (\alpha + \delta)/2 =: m$, the mean of the diagonal elements of A, and $\lambda_1\lambda_2 = \widehat{\alpha}^2 - \widehat{\beta}\,\widehat{\gamma} = \det(A)$, which give the formulas used in `DLANV2`, $\lambda_1, \lambda_2 = m \pm (\widehat{\beta}\widehat{\gamma})^{1/2}$. When $\widehat{\beta}\widehat{\gamma} \geq 0$, real eigenvalues are obtained, which can be computed using the formulas in Algorithm `Schur2`. However, these formulas do not ensure the condition $|\lambda_1| \geq |\lambda_2|$, since the signs of m and p could be different. In order to enforce this condition, the modified code uses slightly different formulas, namely

$$p = \mathrm{sign}(m)\,(\widehat{\beta}\widehat{\gamma})^{1/2}, \quad \widetilde{\alpha} = m + p; \quad \widetilde{\delta} = m - p. \tag{5}$$

The last formula is actually used only when a cancellation error cannot appear (see below). In addition, the formulas for obtaining the plane rotation annihilating $\widehat{\gamma}$ must be changed, giving

$$\widetilde{c} = \mathrm{sign}(m)\,\mathrm{sign}(\widehat{\gamma})\,|\widehat{\beta}|^{1/2}\tau; \quad \widetilde{s} = |\widehat{\gamma}|^{1/2}\tau;$$

to guarantee that this plane rotation actually performs the required transformation. Indeed, $\widetilde{c}^2 + \widetilde{s}^2 = 1$, and, for $\widehat{\beta}\,\widehat{\gamma} > 0$, one obtains

$$\begin{bmatrix} \widetilde{c} & \widetilde{s} \\ -\widetilde{s} & \widetilde{c} \end{bmatrix} \begin{bmatrix} \widehat{\alpha} & \widehat{\beta} \\ \widehat{\gamma} & \widehat{\alpha} \end{bmatrix} \begin{bmatrix} \widetilde{c} & -\widetilde{s} \\ \widetilde{s} & \widetilde{c} \end{bmatrix} = \begin{bmatrix} \widehat{\alpha} + \widetilde{c}\widetilde{s}(\widehat{\beta} + \widehat{\gamma}) & \widetilde{c}^2\widehat{\beta} - \widetilde{s}^2\widehat{\gamma} \\ \widetilde{c}^2\widehat{\gamma} - \widetilde{s}^2\widehat{\beta} & \widehat{\alpha} - \widetilde{c}\widetilde{s}(\widehat{\beta} + \widehat{\gamma}) \end{bmatrix} = \begin{bmatrix} \widetilde{\alpha} & \widetilde{\beta} \\ 0 & \widetilde{\delta} \end{bmatrix}, \tag{6}$$

because

$$\widetilde{c}/\widetilde{s} = \mathrm{sign}(m)\,\mathrm{sign}(\widehat{\gamma})\,(\widehat{\beta}/\widehat{\gamma})^{1/2}; \qquad \widetilde{c}^2\widehat{\gamma} - \widetilde{s}^2\widehat{\beta} = \widetilde{s}^2((\widetilde{c}/\widetilde{s})^2\widehat{\gamma} - \widehat{\beta}) = 0;$$

$$\widehat{p} := \widetilde{c}\widetilde{s}\,(\widehat{\beta} + \widehat{\gamma}) = \mathrm{sign}(m)\,\mathrm{sign}(\widehat{\gamma})\,(\widehat{\beta}\widehat{\gamma})^{1/2}(\widehat{\beta} + \widehat{\gamma})/|\,\widehat{\beta} + \widehat{\gamma}\,| = p.$$

Clearly, $\widetilde{\alpha}$ and $\widetilde{\delta}$ satisfy (5), and $\widetilde{\beta} = \widetilde{c}^2\,\widehat{\beta} - \widetilde{s}^2\,\widehat{\gamma} = (|\,\widehat{\beta}\,|\,\widehat{\beta} - |\,\widehat{\gamma}\,|\,\widehat{\gamma})\tau^2 = \widehat{\beta} - \widehat{\gamma}$. More details are given now concerning the calculation of eigenvalues. If $M := \max(|\alpha|, |\beta|, |\gamma|, |\delta|) > 0$, which holds for $\beta \neq 0$, or $\gamma \neq 0$, then the quantity $(p/M)^2 + (\beta/M)(\gamma/M)$ (with $p = (\alpha - \delta)/2$), can be computed without overflows. If this quantity is greater than or equal to zero, then the eigenvalues are real, and two logical variables are set in the code to define the specific path to be followed.

Some ratios, defined as $r(a, b) = \min(|a|, |b|)/\max(|a|, |b|) \leq 1$[2] are used to check if cancellation errors when forming $a + b$ or $a - b$ could appear. If $r \leq \theta$, with θ a given threshold, then a and b are considered as being far enough and the above mentioned sums can be computed safely. `DLANV2A` code uses $\theta = 0.1$, but it appears that it is rather insensitive to the value of θ in a wide range. Taking $\theta = 1$ will essentially provide the same results as the previous `DLANV2` implementation.

The largest real eigenvalue, λ_1, is stably computed using $\lambda_1 = m + p$, where $\mathrm{sign}(m) = \mathrm{sign}(p)$. Then, if $r_1 := r(m, p) \leq \theta$, λ_2 can be obtained safely as $\lambda_2 = m - p$. Otherwise, the scaled determinant of A is found using the original or updated data. Specifically, let $d := (\alpha/M)\,\delta - (\beta/M)\,\gamma$ be the scaled determinant computed using the original elements of A, and $\widehat{d} := (\widehat{\alpha}\,/\,\widehat{M})\,\widehat{\alpha} - (\widehat{\beta}\,/\,\widehat{M})\,\widehat{\gamma}$

[2] with $\max(|a|, |b|) > 0$, in our context

be the scaled determinant computed using the values obtained after Step 1 of the algorithm, where $\widehat{M} := \max(|\,\widehat{\alpha}\,|, |\,\widehat{\beta}\,|, |\,\widehat{\gamma}\,|) > 0$. If $\text{sign}(\alpha) \neq \text{sign}(\delta)$ and $\text{sign}(\beta) = \text{sign}(\gamma)$, then $\widehat{d}$ does not need to be calculated, and λ_2 is computed as $\lambda_2 = d\,M/\lambda_1$. (Actually, in order to avoid possible overflows in this calculation, the product dM is formed only if $M \leq 1$; otherwise, the code first compute d/λ_1, and then multiplies it by M.) Otherwise, one computes $\widehat{M}$, $\widehat{d}$, $r_2 := r((\alpha/M)\delta, (\beta/M)\gamma)$, and $r_3 := r((\widehat{\alpha}\,/\,\widehat{M})\,\widehat{\alpha}, (\widehat{\beta}\,/\,\widehat{M})\,\widehat{\gamma})$, and the formula corresponding to the minimum between r_1, r_2, and r_3, $r_{\min} = \min(r_1, r_2, r_3)$, is used for computing λ_2. Specifically, if $r_{\min} = r_3$, then $\lambda_2 = \widehat{d}\,\widehat{M}/\lambda_1$. (Again, care is taken to avoid overflows.)

Other differences between `DLANV2A` and `DLANV2` 2.0 are:

1. The calculations for equating α and δ are performed only when these quantities differ.
2. The calculations of the first plane rotation (for equating α and δ) are simplified.
3. Some special cases are dealt with more accurately or directly, e.g., when $|\widehat{\beta}| = |\widehat{\gamma}|$, then p is just set to $p = \text{sign}(m)\,|\widehat{\beta}|$, or when $m = p$, then λ_2 is set to zero.
4. Moreover, m and the scaled determinant are computed or not for the updated values obtained after Step 1 of Algorithm `Schur2`, depending on whether their values can or cannot be accurately determined using the original data. For instance, the original data are used for computing m if $\text{sign}(\alpha) = \text{sign}(\delta)$, or $r(\alpha, \delta) \leq \theta$.

`DLANV2A` code tries to reduce the overhead added to the previous implementation, but improves the accuracy of real λ_2, when possible. When the previous calculations can be performed safely, `DLANV2A` performs very few additional operations (e.g., it saves the matrix elements, and computes M).

Unfortunately, the preliminary code based on the approach A strangely reported "invalid operations" on Sun workstations, during the extensive LAPACK tests.

Approach B. The second approach consisted in a less significant change in the `DLANV2` 2.0 code. The new code, called here `DLANV2B`, essentially uses the EISPACK approach for matrices with two real eigenvalues. The calculations are, however, scaled in order to avoid overflows. (Note that the EISPACK code is not robust enough, and it does not work for matrices with entries close to the overflow limit, as those included in the LAPACK `DGET33` test, while the scaled version worked.) To give some details, z in Section 2 is computed and used as follows. First,

$$z = (p/\widehat{\sigma})p + (\max(|b|, |c|)/\widehat{\sigma})\,\min(|b|, |c|)\text{sign}(b)\text{sign}(c),$$

is evaluated, where $\widehat{\sigma} = \max(|p|, |b|, |c|)$. If $z \geq 4\varepsilon_M$, then z is updated, $z = p + \widehat{\sigma}^{1/2} z^{1/2}\text{sign}(p)$, and EISPACK formulas are used. Otherwise, hence if z is of the order of the machine accuracy, it is not used to decide on the nature of eigenvalues, and LAPACK scheme, based on equating the diagonal elements, is applied. This strategy prevents for taking wrong decisions. Hence, for pairs of real eigenvalues, `DLANV2B` uses EISPACK strategy for most of the time, but switches to LAPACK strategy whenever needed. Cancellation errors are still possible in `DLANV2B` when computing the scaled version of $p^2 + bc$, or the eigenvalues, but these errors are here not so dangerous as when computing, e.g., $p - (p^2 + bc)^{1/2}$. Moreover, it is no more possible to guarantee that $|\lambda_1| \geq |\lambda_2|$, without a sensible complication of the code, and a corresponding reduction in efficiency.

5 Numerical tests

The new implementations has been tested using the LAPACK test routine for `DLANV2`, called `DGET33`, as well as for the calculation of the "periodic" Schur form, which is equivalent to the Schur form of a product of q square matrices A_i, $i = 1{:}q$, but it is computed essentially without forming their product.

The accuracy of computed eigenvalues was checked by comparing their sum and scaled product with the sum of the diagonal elements, and scaled determinant, respectively. The following code was added in the inner loop of `DGET33`:

```
      EPRO = MAX( ABS( RT1R ), ABS( RT2R ) )
      ERR1 = ABS( RT1R + RT2R - ( T1( 1, 1 ) + T1( 2, 2 ) ) )
      IF( EPRO.NE.ZERO ) ERR1 = ERR1 / EPRO
      EPRO = MAX( ABS( RT1R ), ABS( RT1I ), ABS( RT2R ), TNRM )
      SMOD = ( RT1R / EPRO )*( RT2R / EPRO ) + ( RT1I / EPRO )**2
      ERR2 = ABS( SMOD - ( T1( 1, 1 ) / EPRO )*( T1( 2, 2 ) / EPRO ) +
     $                   ( T1( 1, 2 ) / EPRO )*( T1( 2, 1 ) / EPRO ) )
      ERR = ERR + MAX( ERR1, ERR2 )
```

where ERR was initialized to zero on entry in the DGET33 routine, and EPRO, ERR, ERR1, ERR2, and SMOD are added double precision local scalars. The resulted ERR (the cummulated sum of error values for all generated test matrices with real eigenvalues) was .4773959D-12 for the LAPACK 2.0 DLANV2 version, .287160D-13 for DLANVA, and only .988456D-288 for DLANV2B.

The output of DGET33 test routine (for all 20,736 test matrices), was RMAX = .10D+02, LMAX = 16394, for DLANV2 2.0, and RMAX = .10D+02, LMAX = 18344, for DLANV2A, so the value of the largest test ratio, RMAX, was the same, but the new version achieved it on a larger example number, LMAX, hence more problems were solved with a smaller test ratio by DLANV2A than by DLANV2. But DLANV2B gave RMAX = .61D+01, LMAX = 11453, hence RMAX has been almost halved. Therefore, DLANV2B has been retained in LAPACK Release 3.0. Even better results could be obtained by removing the standardization for a matrix with complex eigenvalues, but this could not be accepted in LAPACK.

In few tests related to the calculation of the periodic Schur form, both the old and new versions gave very similar results, but for one example, the error norm, $e := \sum_{i=1}^{q} \| Z_i^T A_i Z_{i+1} - \widetilde{A}_i \|_F$, where $\widetilde{A}_i$, $i = 1{:}q$, are the computed matrices, and Z_i, $i = 1{:}q-1$, and $Z_p = Z_1$ are orthogonal, had the value 4.48D-12, using LAPACK 2.0 routine DLANV2, and the value 7.20D-15, using DLANV2A implementation.

6 Conclusions

Algorithmic and implementation details for computing the eigenvalues and real Schur form of 2×2 real matrices have been described. The changes performed in the LAPACK routine DLANV2 enabled to improve the accuracy of the smallest eigenvalue for matrices with real eigenvalues, while delivering the same results for other classes of matrices. The best numerical strategy found has been incorporated in the DLANV2 routine of LAPACK Release 3.0.

References

[1] Anderson, E., Bai, Z., Bischof, C., Demmel, J., Dongarra, J., Du Croz, J., Greenbaum, A., Hammarling, S., McKenney, A., Ostrouchov, S., and Sorensen, D. (1995). *LAPACK Users' Guide: Second Edition*, SIAM, Philadelphia.

[2] Golub, G. H., and Van Loan, C. F. (1983). *Matrix Computations*, The Johns Hopkins University Press, Baltimore, Maryland.

[3] Sima, V. (1996). *Algorithms for Linear-Quadratic Optimization*, volume 200 of *Pure and Applied Mathematics: A Series of Monographs and Textbooks*, Marcel Dekker, Inc., New York.

[4] Smith, B. T., Boyle, J. M., Dongarra, J. J., Garbow, B. S., Ikebe, Y., Klema, V. C., and Moler, C. B. (1976). *Matrix Eigensystem Routines—EISPACK Guide*, volume 6 of *Lect. Notes in Computer Science*, Springer-Verlag, New York, second edition.

[5] Stewart, G. W. (1973). *Introduction to Matrix Computations*, Academic Press, New York.

[6] Wilkinson, J. H. (1965). *The Algebraic Eigenvalue Problem*, Oxford University Press (Clarendon), London.