Elżbieta Tyrkiel

PROCEDURY NUMERYCZNE DLA ROZWIĄZANIA UOGÓLNIONEGO ZAGADNIENIA WŁASNEGO METODĄ QZ

33/1993

WARSZAWA 1993

ISSN 0208-5658

Praca wpłynęła do Redakcji dnia 29 sierpnia 1993 r.

Instytut Podstawowych Problemów Techniki PAN
Nakład 100 egz. Ark.wyd. 5,5 Ark.druk. 7,00
Oddano do drukarni w październiku 1993

Wydawnictwo Spółdzielcze sp. z o.o.
Warszawa, ul.Jasna 1

Elżbieta Tyrkiel
Samodzielna Pracownia
Aeroelastyczności

PROCEDURY NUMERYCZNE DLA ROZWIĄZANIA UOGÓLNIONEGO ZAGADNIENIA WŁASNEGO METODĄ QZ

Streszczenie

Opisano szczegółowo opracowane pakiety procedur numerycznych umożliwiających rozwiązanie regularnego uogólnionego zagadnienia własnego $Ax = \lambda Bx$ z dowolnymi macierzami kwadratowymi, rzeczywistymi lub zespolonymi. Podstawą algorytmów jest wprowadzona przez Molera i Stewarta metoda QZ, będąca uogólnieniem standardowej metody QR na zagadnienie własne z dwiema macierzami. Metoda QZ charakteryzuje się dużą stabilnością numeryczną dzięki zastosowaniu przekształceń unitarnych. Umożliwia ona wyznaczenie wszystkich wartości własnych (skończonych i nieskończonych) również w przypadku, gdy B jest macierzą osobliwą, gdyż nie wymaga obliczania macierzy B^{-1}.

Programy numeryczne napisane są w języku Turbo Pascal 4.0. Stanowią one uzupełnienie oprogramowania podstawowego mikrokomputerów IBM PC w zakresie algebry liniowej, utworzonego w Samodzielnej Pracowni Aeroelastyczności IPPT PAN. Procedury umożliwiają wyznaczenie pełnego zestawu wartości, jak również prawych i lewych wektorów własnych (znajomość lewych wektorów jest potrzebna w analizie zaburzeń). Działanie algorytmów zilustrowano na przykładach wyników wybranych testów numerycznych; wykazują one stabilność numeryczną skończonych wartości własnych również w przypadku, gdy B jest macierzą osobliwą.

I. WSTĘP

Problem uogólnionego zagadnienia własnego sprowadza się do znalezienia nietrywialnych rozwiązań równania:

$$A x = \lambda B x , \tag{1}$$

gdzie A i B są macierzami kwadratowymi stopnia n. Rozwiązanie to określone jest przez pełny zestaw wartości własnych λ oraz odpowiadających im niezerowych wektorów własnych x.

Jeśli macierz B jest nieosobliwa, wówczas zagadnienie to jest równoważne standardowemu zagadnieniu własnemu

$$B^{-1}A\,x = \lambda\,x\,, \tag{2}$$

które może być rozwiązane za pomocą dobrze znanych, efektywnych procedur i algorytmów [1,2].

Komputerowe rozwiązanie uogólnionego zagadnienia własnego ma szczególnie istotne znaczenie, gdy macierz B jest osobliwa lub bliska osobliwej (t.j. gdy zagadnienie odwrócenia macierzy B jest źle uwarunkowane). Wówczas liczba pierwiastków równania charakterystycznego $\det(A - \lambda B) = 0$ jest mniejsza niż n (współczynnik wielomianu charakterystycznego przy λ^n wynosi $\det(B)$). Jeśli zagadnienie jest *regularne*, co oznacza, że równanie charakterystyczne $\det(A - \lambda B) = 0$ nie jest spełnione tożsamościowo, wówczas "brakujące" wartości własne traktowane są jako istniejące i równe "nieskończoności". Może to być uzasadnione faktem, że przy niewielkim zaburzeniu elementów osobliwej macierzy B pojawia się pewna liczba dużych wartości własnych, rosnących nieskończenie podczas redukowania zaburzenia do zera.

Uogólnione zagadnienie własne zapisać można w dogodniejszej dla interpretacji postaci:

$$\beta\,A\,x = \alpha\,B\,x \tag{3}$$

gdzie wartość własna $\lambda = \alpha/\beta$. Zapis ten ilustruje symetrię ról macierzy A i B oraz fakt, że skończone i nieskończone ($\beta = 0$) wartości własne traktowane są identycznie. W ten sposób każde regularne uogólnione zagadnienie własne posiada n wartości własnych, reprezentowanych przez odpowiadające sobie pary wartości (α,β).

W przypadku, gdy równanie charakterystyczne $\det(A - \lambda B) = 0$ jest spełnione tożsamościowo (t.j. gdy macierze A i B mają wspólną podprzestrzeń zerową), zagadnienie określa się jako *singularne*, a jego rozwiązanie jest zwykle niejednoznaczne. Ten

przypadek nie jest brany w pracy pod uwagę; omawiana metoda obliczeń dotyczy wyłącznie zagadnień regularnych.

W praktyce numerycznej nie ma ścisłego rozgraniczenia między "osobliwością" i "nieosobliwością" macierzy, ze względu na skończoną dokładność arytmetyki zmiennoprzecinkowej i błędy zaokrągleń. Jeśli uwarunkowanie macierzy B jest nieznane przed przystąpieniem do problemu, powinno być ono zbadane (poprzez wyznaczenie wskaźnika uwarunkowania [3]), ponieważ w przypadku złego uwarunkowania rozwiązanie równania (2) obarczone będzie zawsze dużymi błędami, nawet jeśli obliczenia przebiegną bez zakłóceń.

Metody rozwiązania zagadnienia (1) w przypadku, gdy B jest macierzą osobliwą lub bliską osobliwej, były przedmiotem prac szeregu autorów. Peters i Wilkinson [4] zaproponowali podejście polegające na przybliżonym wyznaczeniu podprzestrzeni zerowej macierzy B i eliminacji jej z zagadnienia, a więc deflację problemu. Fix i Heiberger [5] zastosowali wariant tej metody dla przypadku, gdy macierze A i B są symetryczne, a macierz B jest bliska półokreślonej (jeśli macierz B jest dodatnio określona, rozwiązanie oparte jest na znanej metodzie Choleskiego - Wilkinsona [6]).

Metoda QZ, wprowadzona przez Molera i Stewarta [7], jest rozszerzeniem i uogólnieniem algorytmu iteracyjnego QR (stosowanego dla rozwiązania standardowego zagadnienia własnego [1], [2]) na przypadek zagadnienia z dwiema macierzami. Dzięki zastosowaniu elementarnych przekształceń unitarnych metoda ta przewyższa poprzednio wspomniane metody pod względem stabilności numerycznej, choć nie zachowuje ona symetrii macierzy, a także jest bardziej czasochłonna. Metoda QZ pozwala wyznaczyć (przy użyciu jednolitej techniki obliczeniowej) pełny zestaw (skończonych i nieskończonych) wartości własnych każdego regularnego zagadnienia własnego, przy czym uzyskiwana dokładność obliczeń wartości skończonych jest niezależna od tego, czy B jest macierzą osobliwą.

Idea metody QZ oparta jest na następującym twierdzeniu ([8],[9]), będącym uogólnieniem twierdzenia Schura na przypadek regularnej pary macierzy:

Twierdzenie. Jeśli A i B są macierzami opisującymi regularne uogólnione zagadnienie własne, to istnieją takie macierze unitarne Q i Z, że macierze QAZ oraz QBZ są trójkątne górne.

Jeśli macierz B jest nieosobliwa, to, wprowadzając oznaczenia: $\hat{A} = QAZ$ oraz $\hat{B} = QBZ$, otrzymamy:

$$\hat{A}\hat{B}^{-1} = QAZZ^{H}B^{-1}Q^{H} = QAB^{-1}Q^{H} \quad ,$$

czyli macierz $\hat{A}\hat{B}^{-1}$ jest unitarnie podobna macierzy AB^{-1}.

Zagadnienia $Ax = \lambda Bx$ oraz $\hat{A}y = \lambda\hat{B}y$ określa się jako **unitarnie równoważne**. Wartości własne tych zagadnień są takie same, a wektory własne związane są zależnością:

$$x = Zy \ .$$

Jeśli rozpatrujemy zagadnienie z lewymi wektorami własnymi, to, analogicznie, unitarnie równoważne są zagadnienia $u^{T}A = \lambda u^{T}B$ oraz $v^{T}\hat{A} = \lambda v^{T}\hat{B}$, a lewe wektory własne wiąże wzór:

$$u^{T} = v^{T}Q.$$

Wartości własne zagadnienia $\hat{A}y = \lambda\hat{B}y$ z macierzami trójkątnymi $\hat{A}$ i $\hat{B}$ są równe ilorazom odpowiadających sobie elementów diagonalnych, reprezentujących, zgodnie z (3), pary wartości (α_i, β_i):

$$\lambda_i = \frac{\hat{a}_{ii}}{\hat{b}_{ii}} = \frac{\alpha_i}{\beta_i} \ , \qquad i = 1,..,n$$

i mogą wydzielać się w trakcie obliczeń w dowolnej kolejności. Jeśli $\beta_i = 0$, wówczas i-tą wartość własną traktuje się jako "nieskończoną", a odpowiadający jej wektor własny jest niezerowym wektorem z podprzestrzeni zerowej macierzy B, t.j. spełniającym równanie $Bx = 0$. Podobnie gdy $\alpha_i = 0$, wektor własny odpowiadający wartości $\lambda_i = 0$ jest niezerowym wektorem z pod-

przestrzeni zerowej macierzy A.

Numeryczna realizacja metody QZ polega na wykonaniu jednoczesnego przekształcenia obu macierzy A i B, opisanego przytoczonym uogólnionym twierdzeniem Schura. Pełny algorytm rozwiązania uogólnionego zagadnienia własnego składa się z czterech etapów:

1. redukcja pary macierzy (A,B) do uogólnionej postaci Hessenberga, t.j. takiej, w której macierz A jest macierzą górną Hessenberga, macierz B - macierzą trójkątną górną;

2. sprowadzenie macierzy A metodą iteracyjną do postaci quasi-trójkątnej (t.j. zawierającej bloki diagonalne 1x1 oraz 2x2), z jednoczesnym zachowaniem trójkątnej postaci macierzy B;

3. dalsza redukcja macierzy A do postaci trójkątnej - wyodrębnienie elementów diagonalnych;

4. obliczenie wektorów własnych otrzymanego systemu z macierzami trójkątnymi oraz ich transformacja do wyjściowego układu współrzędnych.

Zasadniczym i najbardziej czasochłonnym etapem algorytmu jest etap 2 - iteracje QZ. Realizowany jest on przy użyciu analogicznych strategii iteracyjnych, jak w metodzie QR, określonych w szczególności sposobem wyboru przesunięcia w celu przyspieszenia zbieżności iteracji. Również większość własności algorytmu QR, takich jak: wydzielanie się wartości własnych wzdłuż diagonali od dołu w górę, możliwość deflacji macierzy w przypadku "małości" dwóch sąsiadujących ze sobą elementów poddiagonalnych, sposób testowania końca iteracji, przenosi się, w formie uogólnionej, na algorytm QZ.

Iteracje QZ inicjowane są standardowym krokiem QR odniesionym do macierzy AB^{-1}. Jednak zastosowana technika obliczeniowa zapewnia pomyślny przebieg obliczeń również w przypadku, gdy B jest macierzą osobliwą, gdyż umożliwia wyznaczenie macierzy transformacji Q i Z bez konieczności obliczania pełnej macierzy

B^{-1}. Dla prawidłowego funkcjonowania algorytmu wymagane jest jedynie spełnienie znacznie słabszych warunków "lokalnej nieosobliwości", które będą omówione w dalszej części pracy.

Triangularyzacja macierzy A i B realizowana jest przy użyciu elementarnych przekształceń unitarnych (w przypadku rzeczywistym - ortogonalnych) Q i Z, t.j. odbić (macierze Householdera) oraz obrotów (macierze Givensa). Zapewnia to numeryczną stabilność metody według Wilkinsona [1], t.j. uzyskane macierze QAZ i QBZ są unitarnie równoważne jedynie lekko zaburzonym macierzom wyjściowym (A + δA) oraz (B + δB). Jeżeli pewna skończona wartość własna problemu $Ax = \lambda Bx$ jest dobrze uwarunkowana w tym sensie, że jest nieczuła na małe zaburzenia δA i δB macierzy, to można uznać, że jest ona policzona dokładnie jako wartość własna problemu $(A + \delta A)x = \lambda(B + \delta B)x$; dokładność ta jest niezależna od tego, czy pozostałe wartości własne zagadnienia mają wartość skończoną, czy nieskończoną (t.j. czy B jest macierzą osobliwą).

Moler i Stewart sformułowali metodę QZ dla zagadnienia własnego z macierzami rzeczywistymi; w opracowanym przez nich algorytmie iteracje QZ wykonywane są za pomocą uogólnionej strategii podwójnego kroku z niejawnymi przesunięciami, stosowanej w iteracjach QR macierzy rzeczywistej [1,2]. Umożliwia ona prowadzenie procesu iteracyjnego wyłącznie w arytmetyce rzeczywistej nawet wówczas, gdy wartości własne zagadnienia są liczbami zespolonymi. Jednak w iteracjach QZ metoda ta może powodować pewne problemy numeryczne w przypadku, gdy B jest macierzą osobliwą, wynikające z możliwości wystąpienia nieskończonych lub też zaniedbywalnie małych wartości przesunięć. Nieco później Ward [10] zaproponował inną, efektywniejszą wersję iteracji QZ dla zagadnienia rzeczywistego, umożliwiającą wybór strategii iteracyjnej (pojedynczego lub podwójnego kroku) oraz odpowiedni dobór przesunięcia na początku każdej kolejnej iteracji, w zależności od typu wyznaczanych wartości własnych (rzeczywiste czy zespolone) oraz aktualnych wartości elementów diagonalnych macierzy B. Strategia ta (tzw. algorytm QZ z kombinowanym przesunięciem)

pozwoliła wyeliminować problemy związane z metodą podwójnego kroku, a także okazała się mniej czasochłonna. W pracy [10] dokonano również uogólnienia metody QR dla wykrywania dwóch sąsiadujących ze sobą "małych" elementów poddiagonalnych na przypadek iteracji QZ z pojedynczym przesunięciem.

Celem niniejszej pracy było opracowanie procedur numerycznych umożliwiających rozwiązanie regularnego uogólnionego zagadnienia własnego $Ax = \lambda Bx$ z dowolnymi macierzami rzeczywistymi lub zespolonymi metodą QZ. Opracowane algorytmy realizują czteroetapowy schemat obliczeń Molera - Stewarta, przedstawiony na str. 7. Zastosowane techniki iteracji QZ oparto na metodach opisanych przez Warda [10]. W przypadku zagadnienia z macierzami zespolonymi iteracje QZ realizowane są uogólnioną metodą pojedynczego kroku z niejawnym przesunięciem; dla zagadnienia z macierzami rzeczywistymi zastosowano algorytm iteracyjny z kombinowanym przesunięciem. Wektory własne zredukowanego zagadnienia z macierzami trójkątnymi QAZ i QBZ wyznaczane są uogólnioną metodą "podstawienia wstecznego" [2,11], a następnie sprowadzane do wektorów zagadnienia wyjściowego z użyciem zapamiętanych w trakcie wykonywania przekształceń macierzy transformacji Z (wektory prawe) oraz Q (wektory lewe).

Opracowane programy obliczeniowe napisane są w języku Turbo Pascal 4.0. Stanowią one uzupełnienie oprogramowania podstawowego mikrokomputerów IBM PC w zakresie algebry liniowej, utworzonego uprzednio i opisanego w pracach Nowaka [11] i [12]. Oprogramowanie to tworzone było z uwzględnieniem potrzeby rozwiązywania dużych (rzędu 100) układów równań z pełnymi macierzami (rzeczywistymi i zespolonymi) oraz różnych zagadnień własnych macierzy, co spowodowało konieczność opracowania szeregu metod usprawniania programów; polegają one na zastosowaniu odpowiedniej techniki korzystania z pełnej pamięci mikrokomputera (bez ograniczeń na długość segmentu danych), połączonej z dynamiczną rezerwacją miejsca na tablice, oraz przyspieszaniu działania programów poprzez wykorzystanie wstawek (procedur) w języku assemblera.

Podczas tworzenia algorytmu QZ starano się zachować jednolitość stylu z istniejącym już oprogramowaniem, korzystając z zawartych w nim usprawnień. Wykorzystane zostały (w postaci wstawek kodowych) procedury assemblerowe z pakietów VECREAL i VECCOMP, przyspieszające obliczenia związane z działaniami na macierzach (VecScal, VecProd, NormE, VecConjug, Househ, Rotate, Move), jak również procedury z pakietu działań na liczbach zespolonych COMPUNIT (Add, Sub, CMult, RMult, DivC, DivR, CSqr, CSqrt, CAbs, Neg). Wykorzystano także plik NEWTYPE definiujący nowe typy zmiennych (IntPtr1, DoublePtr1, DoublePtr2, CompPtr1, CompPtr2). Zgodnie ze sposobem segmentacji programów przewidzianym przez Turbo Pascal, procedury realizujące algorytm QZ zgromadzone zostały w postaci pakietów (units), zawierających części INTERFACE i IMPLEMENTATION. Pakiety te mogą być kompilowane niezależnie od programu głównego. Odwołanie się do procedur określonego pakietu musi być poprzedzone w programie głównym dyrektywą uses z nazwą tego pakietu; powoduje to automatycznie, że część INTERFACE pakietu staje się "widoczna" z programu głównego.

Sposób budowania procedur realizujących metodę QZ oraz zabiegi techniczne usprawniające przebieg obliczeń wzorowane były, o ile to było możliwe, na schematach zastosowanych w pakietach REALEIGEN i COMPEIGEN, opisanych w [11] i [12] i przeznaczonych dla rozwiązania zwykłego zagadnienia własnego metodą QR. Zachowano również częściowo zastosowane w powyższych programach nazewnictwo procedur i funkcji (iter, step, nextw, fin). Ze względu na nieco odmienną gospodarkę pamięcią, zwłaszcza przy zapamiętywaniu wyników obliczeń, zbudowano po trzy odrębne pakiety (dla przypadku rzeczywistego i zespolonego); odpowiadają one obliczaniu tylko wartości własnych (QZVALR, QZVALC), wartości i prawych wektorów własnych (QZVECR, QZVECC) oraz pełnego uogólnionego zagadnienia własnego, t.j. wartości oraz prawych i lewych wektorów własnych (QZEIGR, QZEIGC). Analogiczne pakiety opracowane dla zwykłego zagadnienia własnego opisane są w [11].

Pakiety QZEIGR i QZEIGC utworzone zostały przede wszystkim

w celu ich ewentualnego wykorzystania w analizie zaburzeń, wymagającej znajomości lewych wektorów własnych [1,11,14].

W opisach algorytmów stosuje się następujące oznaczenia:

x,y,... /małe litery/ - wektory,

A,B,... /duże litery/ - macierze,

a_{ij} - element (i,j) macierzy A,

a_{ij}^* - wartość sprzężona (w sensie zespolonym) elementu a_{ij},

A^T - macierz transponowana,

$A^H = (A^T)^*$ - macierz hermitowsko sprzężona,

A^{-1} - macierz odwrotna,

$\|x\| = \|x\|_E$ - norma euklidesowa wektora x.

$\|A\|_E$ - norma euklidesowa macierzy A,

$$\|A\|_1 = \max_{1 \le j \le n} \sum_{i=1}^{n} |a_{ij}| ,$$

eps - dokładność arytmetyki zmiennoprzecinkowej.

II. ELEMENTARNE PRZEKSZTAŁCENIA UNITARNE

W algorytmie QZ triangularyzacja macierzy A i B dokonywana jest za pomocą sekwencji lewostronnych (Q) oraz prawostronnych (Z) elementarnych przekształceń unitarnych, mających na celu wyzerowanie elementów poddiagonalnych tych macierzy. Zastosowano 2 rodzaje tych przekształceń: odbicia (przekształcenia Householdera) oraz obroty (przekształcenia Givensa).

Przekształcenie Householdera dokonywane jest za pomocą hermitowskiej, unitarnej macierzy odbicia H_r stopnia n o budowie:

$$H_r = I - (\alpha u_r) u_r^H$$

gdzie wektor u_r jest wektorem o n składowych, z czego pierwszych r-1 składowych jest równych zeru; ponadto wektor ten

jest normowany tak, by jego r-ta składowa była równa 1.

Dla określonego wektora x można tak dobrać elementy wektora u_r, aby przy lewostronnym mnożeniu przez H_r wektor

$$H_r x = x - (\alpha u_r)(u_r^H x)$$

miał składowe r+1,...,n równe zeru, składową r zmienioną, a składowe 1,...,r-1 niezmienione. W tym celu wektor u_r oraz skalar α wybiera się następująco [1]:

$$\begin{aligned} u_{ri} &= 0, && i = 1,\ldots,r-1, \\ u_{rr} &= 1, \\ u_{rj} &= \frac{x_j}{x_r \pm S}, && j = r+1,\ldots,n, \\ \alpha &= \frac{x_r \pm S}{\pm S}, && S = \left(\sum_{i=r}^{n} |x_i|^2 \right)^{1/2}. \end{aligned} \tag{4}$$

W powyższych równaniach znak ± dobiera się tak, aby był równy znakowi elementu x_r. Nowa wartość elementu x_r jest równa $\pm S_r$.

Podobnie przy prawostronnym mnożeniu wektora x^T przez macierz H_r można anulować elementy 1,...,r-1 tego wektora. Wówczas nowy wektor $x^T H_r$ ma postać:

$$x^T H_r = x^T - \alpha(x^T u_r) u_r^H$$

gdzie wektor u_r dobiera się następująco:

$$\begin{aligned} u_{rj} &= \frac{x_j}{x_r \pm S}, && j = 1,\ldots,r-1, \\ u_{rr} &= 1, \\ u_{ri} &= 0, && i = r+1,\ldots,n, \end{aligned}$$

$$\alpha = \frac{x_r \pm S}{\pm S}, \qquad S = \left(\sum_{i=1}^{r} |x_i|^2 \right)^{1/2}.$$

Lewostronne (prawostronne) przekształcenia Householdera stosuje się zwykle dla jednoczesnego wyzerowania więcej niż jednego elementu poddiagonalnego w określonej kolumnie (wierszu) macierzy A lub B.

Przekształcenie obrotu w płaszczyźnie (i,j) dokonywane jest za pomocą unitarnej macierzy Givensa T_{ij} (j > i), różniącej się od macierzy jednostkowej tylko czterema elementami:

$$\begin{array}{c} \begin{array}{ccccc} & & i & & j \end{array} \\ \begin{bmatrix} 1 & & & & \\ & \ddots & & & \\ & & c & -s^* & \\ & & s & c^* & \\ & & & & 1 \end{bmatrix} \begin{array}{c} \\ \\ i \\ j \\ \\ \end{array} \end{array}$$

spełniającymi równanie: $|c|^2 + |s|^2 = 1$.

Lewostronne mnożenie macierzy A przez macierz obrotu T_{ij} ($\tilde{A} = T_{ij}^H A$) zmienia w niej tylko elementy dwóch wierszy: i oraz j, według wzorów:

$$\tilde{a}_{ik} = c^* a_{ik} + s^* a_{jk}, \qquad k = 1,\ldots,n$$

$$\tilde{a}_{jk} = -s a_{ik} + c a_{jk}, \qquad k = 1,\ldots,n.$$

W celu wyzerowania elementu $\tilde{a}_{ji}$, dla c i s przyjmuje się wartości:

$$c = \frac{a_{ii}}{S}, \qquad s = \frac{a_{ji}}{S}, \qquad \text{gdzie } S = \sqrt{|a_{ii}|^2 + |a_{ji}|^2}.$$

Prawostronne mnożenie macierzy A przez macierz obrotu T_{ij} ($\tilde{A} = AT_{ij}$) zmienia w niej tylko elementy dwóch kolumn: i oraz j, według wzorów:

$$\tilde{a}_{ki} = ca_{ki} + sa_{kj} \quad , \qquad k = 1,\ldots,n \quad ,$$

$$\tilde{a}_{kj} = -s^{*}a_{ki} + c^{*}a_{kj} \quad , \qquad k = 1,\ldots,n \quad ,$$

gdzie w celu wyzerowania elementu a_{ji}, dla c i s przyjmuje się wartości:

$$c = \frac{a_{jj}}{S} \quad , \qquad s = -\frac{a_{ji}}{S} \quad , \qquad \text{gdzie} \quad S = \sqrt{|a_{jj}|^2 + |a_{ji}|^2} \quad .$$

Za pomocą macierzy obrotu T_{ij} można wyzerować dowolny element j-tego wiersza (lub j-tej kolumny) $\tilde{a}_{jm}$ (lub $\tilde{a}_{mj}$), t.j. dla $m \neq i,j$. W tym celu w powyższych formułach należy przyjąć, dla mnożenia lewostronnego:

$$c = \frac{a_{im}}{S} \quad , \qquad s = \frac{a_{jm}}{S} \quad , \qquad \text{gdzie} \quad S = \sqrt{|a_{im}|^2 + |a_{jm}|^2} \quad ,$$

oraz, analogicznie, dla mnożenia prawostronnego:

$$c = \frac{a_{mj}}{S} \quad , \qquad s = -\frac{a_{mi}}{S} \quad , \qquad \text{gdzie} \quad S = \sqrt{|a_{mj}|^2 + |a_{mi}|^2} \quad .$$

Lewo- i prawostronne przekształcenia obrotu zostały zastosowane dla wyzerowania pojedynczych elementów poddiagonalnych macierzy A lub B.

Przekształcenia unitarne zachowują normę euklidesową wektora i macierzy. Dzięki temu wielkość zaburzeń przekształcanych elementów może być oceniana na podstawie tych samych kryteriów, co błędy danych wejściowych i pierwotnego zapisu do pamięci komputera. Upraszcza to również problem zbieżności; przekształcany element a_{ij} może być uznany za równy zeru, jeśli jest w granicach zakłóceń tolerowanych przez wyjściową macierz A, t.j. spełnia warunek:

$$|a_{ij}| \leq eps \cdot \|A\| \quad ,$$

gdzie $\|\cdot\|$ - norma $\|\cdot\|_1$ lub $\|\cdot\|_E$.

III. REDUKCJA PARY MACIERZY (A,B) DO UOGÓLNIONEJ POSTACI HESSENBERGA

Pierwszym krokiem algorytmu QZ jest sprowadzenie macierzy A i B do uogólnionej postaci Hessenberga, t.j. takiej, w ktorej macierz A jest macierzą górną Hessenberga, macierz B - macierzą trójkątną górną. Postępowanie składa się z dwóch etapów:

1. sprowadzenie macierzy B do postaci trójkątnej górnej;
2. sprowadzenie macierzy A do postaci Hessenberga z jednoczesnym zachowaniem trójkątnej postaci B.

Etap 1 realizowany jest przy użyciu n-1 lewostronnych przekształceń Householdera Q_k:

$$B_{k+1} = Q_k B_k , \qquad \text{dla} \quad k = 1,\ldots,n-1 ,$$

gdzie $B_1 = B$, natomiast macierz B_k ma budowę (k = 3, n = 6):

$$\begin{bmatrix} b & b & b & b & b & b \\ 0 & b & b & b & b & b \\ 0 & 0 & b & b & b & b \\ 0 & 0 & b' & b & b & b \\ 0 & 0 & b' & b & b & b \\ 0 & 0 & b' & b & b & b \end{bmatrix} \begin{matrix} \\ \\ k \\ \\ \\ n \end{matrix} .$$

W użytej notacji b oznacza element niezerowy, b' - element, który ma być wyzerowany w kolejnym, k-tym kroku algorytmu. Macierz Q_k dobiera się na podstawie k-tej kolumny macierzy B_k (według wzorów (2)) tak, aby po przekształceniu wyzerowane zostały jej elementy k+1,...,n .

Ostatecznie otrzymujemy:

$$B := Q_{n-1}Q_{n-2}\cdot\ldots\cdot Q_1 B.$$

Jednocześnie dokonuje się lewostronnego przemnożenia macierzy A przez wyznaczone macierze Q_i:

$$A := Q_{n-1}Q_{n-2}\cdot\ldots\cdot Q_1 A.$$

Oznaczając $Q = Q_1\cdot\ldots\cdot Q_{n-2}Q_{n-1}$ otrzymujemy macierze $Q^H A$ oraz

$Q^H B$ w następującej postaci:

$$\begin{bmatrix} a & a & a & a & a & a \\ a & a & a & a & a & a \\ a' & a & a & a & a & a \\ a' & a' & a & a & a & a \\ a' & a' & a' & a & a & a \\ a' & a' & a' & a' & a & a \end{bmatrix} \qquad \begin{bmatrix} b & b & b & b & b & b \\ 0 & b & b & b & b & b \\ 0 & 0 & b & b & b & b \\ 0 & 0 & 0 & b & b & b \\ 0 & 0 & 0 & 0 & b & b \\ 0 & 0 & 0 & 0 & 0 & b \end{bmatrix} .$$

Etap 2 realizowany jest przy użyciu sekwencji $(n^2-3n+2)/2$ obrotów lewostronnych Q_{kl} oraz prawostronnych Z_{kl}. Procedura rozpoczyna się od wyzerowania elementu a_{n1} za pomocą obrotu $Q_{n-1,n}$; dla n = 6 otrzymane macierze $Q_{56}A$ oraz $Q_{56}B$ mają postać:

$$\begin{bmatrix} a & a & a & a & a & a \\ a & a & a & a & a & a \\ a' & a & a & a & a & a \\ a' & a' & a & a & a & a \\ a' & a' & a' & a & a & a \\ 0 & a' & a' & a' & a & a \end{bmatrix} \qquad \begin{bmatrix} b & b & b & b & b & b \\ 0 & b & b & b & b & b \\ 0 & 0 & b & b & b & b \\ 0 & 0 & 0 & b & b & b \\ 0 & 0 & 0 & 0 & b & b \\ 0 & 0 & 0 & 0 & b' & b \end{bmatrix} .$$

Obrót lewostronny Q_{56} spowodował pojawienie się niezerowego elementu poddiagonalnego b_{65}; element ten usuwany jest następnie za pomocą obrotu prawostronnego Z_{56}, który zachowuje zero w pozycji a_{61}:

$$A := Q_{56}AZ_{56}, \qquad B := Q_{56}BZ_{56}$$

W analogiczny sposób zerowane są następne elementy poniżej codiagonali macierzy A, w następującej kolejności:

$$\begin{bmatrix} a & a & a & a & a & a \\ a & a & a & a & a & a \\ a_4 & a & a & a & a & a \\ a_3 & a_7 & a & a & a & a \\ a_2 & a_6 & a_9 & a & a & a \\ a_1 & a_5 & a_8 & a_{10} & a & a \end{bmatrix} ,$$

która gwarantuje zachowanie zerowych wartości elementów macierzy A na pozycjach wyzerowanych uprzednio.

Ostatecznie otrzymujemy:

$$A := Q_{n-1,n} \cdot \cdot \cdot \cdot \cdot \left(Q_{n-2,n-1} \left(Q_{n-1,n} A Z_{n-1,n} \right) Z_{n-2,n-1} \right) \cdot \cdot \cdot Z_{n-1,n}$$

$$B := Q_{n-1,n} \cdot \cdot \cdot \cdot \cdot \left(Q_{n-2,n-1} \left(Q_{n-1,n} B Z_{n-1,n} \right) Z_{n-2,n-1} \right) \cdot \cdot \cdot Z_{n-1,n}$$

gdzie nawiasy oznaczają kolejność wykonywania przekształceń. Oznaczając:

$$Q = Q_{n-1,n} Q_{n-2,n-1} \cdot \cdot \cdot \cdot \cdot Q_{n-1,n} ,$$

$$Z = Z_{n-1,n} Z_{n-2,n-1} \cdot \cdot \cdot \cdot \cdot Z_{n-1,n} ,$$

otrzymujemy macierze $Q^H AZ$ oraz $Q^H BZ$ w następującej postaci:

$$\begin{bmatrix} a & a & a & a & a & a \\ a & a & a & a & a & a \\ 0 & a & a & a & a & a \\ 0 & 0 & a & a & a & a \\ 0 & 0 & 0 & a & a & a \\ 0 & 0 & 0 & 0 & a & a \end{bmatrix} \qquad \begin{bmatrix} b & b & b & b & b & b \\ 0 & b & b & b & b & b \\ 0 & 0 & b & b & b & b \\ 0 & 0 & 0 & b & b & b \\ 0 & 0 & 0 & 0 & b & b \\ 0 & 0 & 0 & 0 & 0 & b \end{bmatrix} ,$$

która jest tzw. uogólnioną postacią **Hessenberga pary macierzy (A,B)**. Jest to postać wyjściowa do następnego etapu algorytmu - iteracji QZ.

Podsumowując, algorytm uzyskania uogólnionej postaci Hessenberga jest następujący:

1. Dla $k = 1,2,\ldots\ldots,n-1$:

   1.1. wybór macierzy Householdera Q_k takiej, by wyzerować elementy $b_{k+1,k}, \ldots\ldots, b_{n,k}$;

   1.2. $B := Q_k B$, $\quad A := Q_k A$.

2. Dla $k = 1,2,\ldots\ldots,n-2$:

   2.1. dla $l = n-1, n-2, \ldots\ldots, k+1$:

      (a) wybór lewostronnej macierzy obrotu Q_{kl} takiej, by wyzerować element $a_{l+1,k}$;

(b) $A := Q_{kl} A, \qquad B := Q_{kl} B;$

(c) wybór prawostronnej macierzy obrotu Z_{kl} takiej, by wyzerować element $b_{l+1,l}$;

(d) $B := BZ_{kl}, \qquad A := AZ_{kl}.$

IV. ITERACJE QZ

Para macierzy (A,B) mająca uogólnioną postać Hessenberga poddawana jest dalszej redukcji metodą iteracyjną. Celem tego procesu jest sprowadzenie macierzy A do postaci quasi-trójkątnej górnej, przy zachowaniu trójkątnej postaci macierzy B. Stosowane techniki iteracji QZ polegają na wyznaczaniu (przy użyciu różnych strategii iteracyjnych) kolejnych, unitarnie równoważnych macierzy $A_{k+1} = Q_k A_k Z_k$ oraz $B_{k+1} = Q_k B_k Z_k$, mających uogólnioną postać Hessenberga; macierze transformacji Q_k muszą być przy tym budowane w ten sposób, by w przypadku, gdy B jest macierzą nieosobliwą, uzyskiwane macierze $C_k = A_k B_k^{-1}$ były takie same, jak macierze otrzymywane w procesie iteracyjnym QR z macierzą wyjściową $C_1 = A_1 B_1^{-1}$.

IV.1. Iteracje QZ dla macierzy zespolonych

IV.1.1. *Metoda pojedynczego kroku z niejawnym przesunieciem*

Uogólniona metoda pojedynczego kroku z niejawnym przesunięciem [10] jest połączeniem stosowanej w zwykłym algorytmie QR strategii iteracyjnej z pojedynczym przesunięciem, ze schematem wyznaczania unitarnie-równoważnych macierzy QAZ i QBZ, sformułowanym przez Molera i Stewarta [7] dla przypadku podwójnego kroku iteracyjnego.

Procedura rozpoczyna się przy założeniu, że macierz B jest nieosobliwa, po czym wykonuje się standardowy pojedynczy krok iteracyjny QR z przesunięciem σ, odniesiony do macierzy $C = AB^{-1}$. Polega on na wyznaczeniu macierzy unitarnej Q,

przekształcającej wyjściową macierz Hessenberga $C_1 = C$ w unitarnie-podobną macierz Hessenberga C_2, według wzorów:

$$Q^H(C_1 - \sigma I) = R \quad ,$$

$$C_2 = RQ + \sigma I = Q^H C_1 Q \quad ,$$

gdzie R jest macierzą trójkątną górną.

Macierz Q spełniająca powyższe zależności jest jednoznacznie określona (z dokładnością do znaków kolumn) przez swoją **pierwszą kolumnę** (wynika to z faktu, że macierz C_2 ma postać Hessenberga oraz z ortogonalności kolumn macierzy Q; dowód tego twierdzenia zamieszczony jest np. w [8]). Ponieważ realizuje ona rozkład trójkątny macierzy $(C - \sigma I)$, może być wyrażona iloczynem (n-1) macierzy Householdera:

$$Q = H_1 H_2 \ldots . H_{n-1} \quad ,$$

gdzie wszystkie macierze H_i $(i = 2,\ldots,n-1)$ mają pierwszą kolumnę jednostkową. A zatem pierwsza kolumna macierzy Q (równa pierwszej kolumnie macierzy H_1) jest w pełni określona przez elementy pierwszej kolumny macierzy $(C - \sigma I)$, a mianowicie:

$$H_1 = I - (\alpha u) u^H \quad ,$$

gdzie, zgodnie z (4):

$$u_1 = 1 \quad ,$$

$$u_i = \frac{c_{i1}}{(c_{11} - \sigma) \pm S} \quad , \qquad i = 2,\ldots,n \quad ,$$

$$\alpha = \frac{(c_{11} - \sigma) \pm S}{\pm S} \quad , \qquad S = \left(|c_{11} - \sigma|^2 + \sum_{i=2}^{n} |c_{i1}|^2 \right)^{1/2} \quad .$$

Ponieważ $C = AB^{-1}$ jest macierzą Hessenberga, pierwsza kolumna macierzy $(C - \sigma I)$ ma w rzeczywistości tylko dwa elementy niezerowe:

$$a_{10} = \frac{a_{11}}{b_{11}} - \sigma \quad ,$$

(5)

$$a_{20} = \frac{a_{21}}{b_{11}}$$

(kolumna ta traktowana jest jako tzw. "zerowa, fikcyjna kolumna" macierzy A). A zatem wektor u definiujący macierz Householdera H_1 ma również tylko 2 elementy niezerowe: u_1, u_2 i jest dobierany tak, by wyzerować element a_{20}.

Dalsza strategia polega na wykonaniu sekwencji (n-1) lewostronnych (Q) i (n-1) prawostronnych (Z) przekształceń macierzy A i B takich, by macierz QAZ była macierzą Hessenberga, a macierz QBZ - macierzą trójkątną górną. Mnożąc lewostronnie macierze A i B przez macierz Householdera $Q_1 = H_1$ otrzymujemy macierze Q_1A i Q_1B w postaci:

$$\begin{bmatrix} a & a & a & a & a & a \\ a & a & a & a & a & a \\ 0 & a & a & a & a & a \\ 0 & 0 & a & a & a & a \\ 0 & 0 & 0 & a & a & a \\ 0 & 0 & 0 & 0 & a & a \end{bmatrix} \qquad \begin{bmatrix} b & b & b & b & b & b \\ b' & b & b & b & b & b \\ 0 & 0 & b & b & b & b \\ 0 & 0 & 0 & b & b & b \\ 0 & 0 & 0 & 0 & b & b \\ 0 & 0 & 0 & 0 & 0 & b \end{bmatrix} .$$

Przekształcenie spowodowało pojawienie się niezerowego elementu poddiagonalnego macierzy B w pozycji (2,1). Element ten usuwany jest następnie za pomocą prawostronnego obrotu Z_1; po wykonaniu mnożenia macierz Q_1AZ_1 ma postać:

$$\begin{bmatrix} a & a & a & a & a & a \\ a & a & a & a & a & a \\ a' & a & a & a & a & a \\ 0 & 0 & a & a & a & a \\ 0 & 0 & 0 & a & a & a \\ 0 & 0 & 0 & 0 & a & a \end{bmatrix} .$$

Następnie zerowany jest element a_{31} za pomocą kolejnego przekształcenia Householdera Q_2; uzyskane macierze $Q_2Q_1AZ_1$ i $Q_2Q_1BZ_1$ są postaci:

$$\begin{bmatrix} a & a & a & a & a & a \\ a & a & a & a & a & a \\ 0 & a & a & a & a & a \\ 0 & 0 & a & a & a & a \\ 0 & 0 & 0 & a & a & a \\ 0 & 0 & 0 & 0 & a & a \end{bmatrix} \qquad \begin{bmatrix} b & b & b & b & b & b \\ 0 & b & b & b & b & b \\ 0 & b' & b & b & b & b \\ 0 & 0 & 0 & b & b & b \\ 0 & 0 & 0 & 0 & b & b \\ 0 & 0 & 0 & 0 & 0 & b \end{bmatrix}$$

W ten sposób niezerowy element poddiagonalny macierzy B "przesunął się" o jedno miejsce niżej, w dół codiagonali.

W analogiczny sposób zerowane są następne elementy poddiagonalne macierzy A i B; ostatecznie otrzymujemy:

$$A := Q_{n-1} \cdot \ldots \cdot \left(Q_2 \left(Q_1 A Z_1 \right) Z_2 \right) \ldots Z_{n-1} \quad ,$$

$$B := Q_{n-1} \cdot \ldots \cdot \left(Q_2 \left(Q_1 B Z_1 \right) Z_2 \right) \ldots Z_{n-1} \quad ,$$

gdzie, oznaczając:

$$Q = Q_1 Q_2 \cdot \ldots \cdot Q_{n-1} \quad ,$$

$$Z = Z_1 Z_2 \cdot \ldots \cdot Z_{n-1} \quad ,$$

macierze $A = Q^H A Z$ oraz $B = Q^H B Z$ mają ponownie uogólnioną postać Hessenberga.

Opisana procedura ilustruje technikę wykonania jednego kroku iteracyjnego z (niejawnym) przesunięciem σ. Aby go zainicjować, należy wyznaczyć elementy a_{10} i a_{20} "zerowej, fikcyjnej kolumny" macierzy A, określone wzorami (5). Dla ich wyznaczenia wymagany jest jedynie warunek:

$$b_{11} \neq 0.$$

Jeśli $b_{11} = 0$, wówczas dokonuje się deflacji problemu "od góry" za pomocą przekształcenia obrotu Q_{12} zerującego element a_{21} (nie narusza ono zer w pozycjach (1,1) oraz (2,1) macierzy B), a następnie kontynuuje procedurę dla macierzy o zmniejszonym wymiarze (od kolumny 2 i wiersza 2). Może się przy tym zdarzyć, że w wyniku zastosowanego przekształcenia nastąpi wyzerowanie

elementu b_{22} (oznacza to wydzielenie się kolejnej "nieskończonej" wartości własnej); proces deflacyjny należy zatem kontynuować tak długo, aż element b_{11} macierzy o zmniejszonym wymiarze spełni warunek $b_{11} \neq 0$:

$$\begin{bmatrix} 0 & b & b & b & .. \\ 0 & b & b & b & .. \\ 0 & 0 & b & b & .. \\ 0 & 0 & 0 & b & .. \\ : & : & : & : & \end{bmatrix} \xrightarrow{Q_{12}} \left[\begin{array}{c|ccc..} 0 & b & b & b & .. \\ \hline 0 & 0 & b & b & .. \\ 0 & 0 & b & b & .. \\ 0 & 0 & 0 & b & .. \\ : & : & : & : & \end{array}\right] \xrightarrow{Q_{23}} \left[\begin{array}{cc|cc..} 0 & b & b & b & .. \\ 0 & 0 & b & b & .. \\ \hline 0 & 0 & b & b & .. \\ 0 & 0 & 0 & b & .. \\ : & : & : & : & \end{array}\right] \dots ,$$

$$\begin{bmatrix} a & a & a & a & .. \\ a' & a & a & a & .. \\ 0 & a & a & a & .. \\ 0 & 0 & a & a & .. \\ : & : & : & : & \end{bmatrix} \xrightarrow{Q_{12}} \left[\begin{array}{c|ccc..} a & a & a & a & .. \\ \hline 0 & a & a & a & .. \\ 0 & a' & a & a & .. \\ 0 & 0 & a & a & .. \\ : & : & : & : & \end{array}\right] \xrightarrow{Q_{23}} \left[\begin{array}{cc|cc..} a & a & a & a & .. \\ 0 & a & a & a & .. \\ \hline 0 & 0 & a & a & .. \\ 0 & 0 & a & a & .. \\ : & : & : & : & \end{array}\right] \dots .$$

Analogicznie do techniki stosowanej w metodzie QR, jako przesunięcie σ wybiera się tę wartość własną dolnego bloku diagonalnego 2 x 2 macierzy AB^{-1}, czyli ten pierwiastek równania kwadratowego

$$\det(\tilde{A} - \sigma\tilde{B}) = 0 \quad , \tag{6}$$

gdzie:

$$\tilde{A} = \begin{bmatrix} a_{n-1,n-1} & a_{n-1,n} \\ a_{n,n-1} & a_{nn} \end{bmatrix}, \qquad \tilde{B} = \begin{bmatrix} b_{n-1,n-1} & b_{n-1,n} \\ 0 & b_{nn} \end{bmatrix},$$

którego wartość jest bliższa n-tej wartości własnej, określonej stosunkiem ostatnich elementów diagonalnych a_{nn}/b_{nn}:

$$\sigma = \sigma_1 \qquad \text{gdy} \quad \left|\sigma_1 - \frac{a_{nn}}{b_{nn}}\right| < \left|\sigma_2 - \frac{a_{nn}}{b_{nn}}\right| ,$$

w przeciwnym wypadku $\sigma = \sigma_2$.

Taki sposób wyboru przesunięć zapewnia najszybsze malenie elementów codiagonali macierzy A, czyli wydzielanie się wartości własnych od dołu w górę [2,11].

Jeśli macierz $\tilde{B}$ jest osobliwa ($b_{nn} = 0$ lub/i $b_{n-1,n-1} = 0$), przesunięcie σ wyznacza się jako pierwiastek równania liniowego (6), według jednego z następujących wzorów:

1. jeśli $b_{n-1,n-1} = 0$:

$$\sigma = \frac{\det(\tilde{A})}{a_{n,n-1}b_{n-1,n} - a_{n-1,n-1}b_{nn}} ; \tag{7}$$

2. jeśli $b_{nn} = 0$:

$$\sigma = \frac{\det(\tilde{A})}{a_{n,n-1}b_{n-1,n} - a_{nn}b_{n-1,n-1}} ; \tag{8}$$

3. jeśli $b_{n-1,n-1} = 0$ oraz $b_{nn} = 0$:

$$\sigma = \frac{\det(\tilde{A})}{a_{n,n-1}b_{n-1,n}} . \tag{9}$$

Jeśli mianownik w aktualnym wyrażeniu (7), (8) lub (9) jest równy zeru, wówczas wykonuje się krok iteracyjny z przesunięciem $\sigma = 0$ [10].

Podsumowując, algorytm wykonania pojedynczego kroku iteracyjnego QZ z niejawnym przesunięciem jest następujący:

1. Wyznaczenie elementów a_{10}, a_{20} według (5)
   (poprzedzone deflacją problemu "od góry", jeśli $b_{11} = 0$);

2. Dla $j = 1,2,.....,n-2$:

   2.1. wyznaczenie macierzy Householdera Q_j takiej, by wyzerować element $a_{j+1,j-1}$;

   $A := Q_j A$, $B := Q_j B$.

   2.2. wyznaczenie macierzy obrotu Z_j takiej, by wyzerować

element $b_{j+1,j}$;

$B := BZ_j$, $A := AZ_j$.

Stosując opisaną procedurę iteracyjnie (z użyciem w każdym k-tym kroku iteracyjnym przesunięcia σ_k) otrzymuje się ciąg unitarnie - równoważnych macierzy:

$$A_{k+1} = Q_k A_k Z_k \ , \qquad B_{k+1} = Q_k B_k Z_k \ ,$$

zachowujących w trakcie procesu iteracyjnego uogólnioną postać Hessenberga. Elementy macierzy oznaczone a_{ij} oraz b_{ij} we wzorach opisujących technikę pojedynczego kroku iteracyjnego odnoszą się wówczas do kolejnych macierzy A_k oraz B_k.

Przebieg procesu iteracyjnego charakteryzuje się tymi samymi własnościami, co w zwykłym algorytmie QR. Zbieżność iteracji określona jest warunkiem zmniejszania się wartości iloczynu dwóch sąsiadujących ze sobą elementów poddiagonalnych macierzy A_k:

$$a^k_{i,i-1}\, a^k_{i+1,i} \longrightarrow 0 \qquad k \longrightarrow \infty \ ,$$

i przy prawidłowym przebiegu procesu elementy diagonalne określające wartości własne wydzielają się od dołu w górę.

Zazwyczaj liczba iteracji niezbędnych do wydzielenia się kolejnej wartości własnej nie przekracza kilku. Mogą jednak istnieć macierze niezmiennicze względem zadanego sposobu wyboru przesunięć; wówczas, aby uzyskać zbieżność, należy ten proces zakłócić wyborem pewnych niestandardowych wartości przesunięć. Wzorując się na strategii stosowanej w algorytmie QR z pojedynczym przesunięciem [11], dla dziesiątej oraz dwudziestej iteracji zastosowano niestandardową wartość przesunięcia σ_k, wybraną następująco:

$$\mathrm{Re}(\sigma_k) = \left| \mathrm{Re}\left(\frac{a^k_{n,n-1}}{b^k_{n-1,n-1}} \right) \right| + \left| \mathrm{Re}\left(\frac{a^k_{n-1,n-2}}{b^k_{n-2,n-2}} \right) \right| ,$$

$$Im(\sigma_k) = \left| Im\left(\frac{a^k_{n,n-1}}{b^k_{n-1,n-1}}\right)\right| + \left| Im\left(\frac{a^k_{n-1,n-2}}{b^k_{n-2,n-2}}\right)\right|$$

(jeśli $b^k_{n-2,n-2} = 0$ lub $b^k_{n-1,n-1} = 0$, wówczas element ten zastępuje się wartością $eps \cdot \|B\|_1$).

IV.1.2. *Testowanie elementów codiagonali macierzy A*

Po zakończeniu każdego k-tego kroku iteracyjnego sprawdzana jest wielkość elementów poddiagonalnych macierzy A_k. Jeśli element $a^k_{n,n-1}$ jest dostatecznie mały według przyjętego kryterium, t.j. spełnia nierówność:

$$|a^k_{n,n-1}| \leq eps \cdot \|A\|_1 ,$$

wówczas traktuje się go jako równy zeru. Elementy diagonalne a^k_{nn} oraz b^k_{nn} wyznaczają wydzieloną wartość własną $\lambda_n = a^k_{nn}/b^k_{nn}$ i dalsze iteracje prowadzone są dla bloków diagonalnych macierzy A i B, obejmujących wiersze i kolumny 1,...,n-1:

$$\begin{bmatrix} a & a & a & a & a \\ a & a & a & a & a \\ & a & a & a & a \\ & & a & a & a \\ & & & \varepsilon & a \end{bmatrix} \longrightarrow \left[\begin{array}{ccccc} a & a & a & a & a \\ a & a & a & a & a \\ & a & a & a & a \\ & & a & a & a \\ \hdashline & & & 0 & a \end{array}\right], \quad \left[\begin{array}{ccccc} b & b & b & b & b \\ & b & b & b & b \\ & & b & b & b \\ & & & b & b \\ \hdashline & & & & b \end{array}\right] \begin{matrix} 1 \\ \\ \\ \\ n-1 \end{matrix}$$

(ε oznacza element spełniający kryterium zaniedbywalności).

Jeśli zamiast elementu a^k_{nn} zaniedbywalnie mały staje się element $a^k_{n-1,n-2}$:

$$|a^k_{n-1,n-2}| \leq eps \cdot \|A\|_1 ,$$

wówczas (traktując go jako równy zeru) dalsze iteracje prowadzi się dla bloków diagonalnych obejmujących wiersze i kolumny 1,...,n-2:

$$\begin{bmatrix} a & a & a & a & a \\ a & a & a & a & a \\ & a & a & a & a \\ & & \varepsilon & a & a \\ & & & a & a \end{bmatrix} \longrightarrow \left[\begin{array}{ccc:cc} a & a & a & a & a \\ a & a & a & a & a \\ & a & a & a & a \\ \hdashline & & 0 & a & a \\ & & & a & a \end{array}\right], \quad \left[\begin{array}{ccc:cc} b & b & b & b & b \\ & b & b & b & b \\ & & b & b & b \\ \hdashline & & & b & b \\ & & & 0 & b \end{array}\right] \begin{matrix} 1 \\ \\ n-2 \end{matrix}$$

Wartości własne λ_n oraz λ_{n-1} są w tym przypadku pierwiastkami równania charakterystycznego:

$$\det(\tilde{A}_k - \lambda \tilde{B}_k) = 0 \quad ,$$

gdzie:

$$\tilde{A}_k = \begin{bmatrix} a^k_{n-1,n-1} & a^k_{n-1,n} \\ a^k_{n,n-1} & a^k_{nn} \end{bmatrix}, \qquad \tilde{B}_k = \begin{bmatrix} b^k_{n-1,n-1} & b^k_{n-1,n} \\ 0 & b^k_{nn} \end{bmatrix}.$$

Przedstawiona technika obliczeniowa pozwala na dokonanie deflacji macierzy również w przypadku, gdy dwa kolejne elementy poddiagonalne nie są zaniedbywalne każdy z osobna, lecz są "małe" w sensie spełnienia odpowiedniego kryterium. W pracy [10] dokonano uogólnienia metody QR dla wykrywania dwóch sąsiadujących ze sobą "małych" elementów poddiagonalnych na przypadek iteracji QZ z pojedynczym przesunięciem.

Jeśli w trakcie procesu iteracyjnego dwa kolejne elementy poddiagonalne $a^k_{r,r-1}$ oraz $a^k_{r+1,r}$ (oznaczone jako ε_1 i ε_2)

$$\begin{bmatrix} & & & r & & \\ a & a & a & a & a & a \\ a & a & a & a & a & a \\ & \varepsilon_1 & a & a & a & a \\ & & \varepsilon_2 & a & a & a \\ & & & a & a & a \\ & & & & a & a \end{bmatrix} \quad r$$

spełniają warunek:

$$(10) \qquad |a^k_{r,r-1} \cdot a^k_{r+1,r}| \le |a^k_{rr} - \sigma_k \cdot b^k_{rr}| \cdot eps \cdot \|A\|_1$$

(gdzie σ_k oznacza przesunięcie zastosowane w k-tym kroku),

wówczas można dokonać podziału macierzy A_k i B_k na niezależne bloki, obejmujące kolumny i wiersze $1,\ldots,r-1$ oraz $r,\ldots,n$. Dalsze iteracje prowadzi się najpierw dla dolnych bloków diagonalnych (od kolumny r i wiersza r), wyznaczając elementy "zerowej, fikcyjnej kolumny" a^k_{10} oraz a^k_{20} (5) jako:

$$a^k_{10} = \frac{a^k_{rr}}{b_{rr}} - \sigma_k ,$$

(11)

$$a^k_{20} = \frac{a^k_{r+1,r}}{b_{rr}} .$$

W tym przypadku, po wykonaniu przekształcenia Householdera Q^k_1 zerującego element a^k_{20} (zmienia ono jedynie wartości elementów wierszy r oraz r+1) otrzymuje się macierz $Q^k_1 A_k$ w postaci:

$$\begin{array}{c} \begin{array}{cccccc} & & r & & & n \end{array} \\ \left[\begin{array}{cc:cccc} a & a & a & a & a & a \\ a & a & a & a & a & a \\ \hdashline & \eta_1 & a & a & a & a \\ & \eta_2 & a & a & a & a \\ & & & a & a & a \\ & & & & a & a \end{array}\right] \begin{array}{c} \\ \\ r \\ \\ \\ n \end{array} \end{array}$$

gdzie, na podstawie (4):

$$\eta_1 = a^k_{r,r-1}\left(1 - \frac{a^k_{10} \pm S}{\pm S}\right),$$

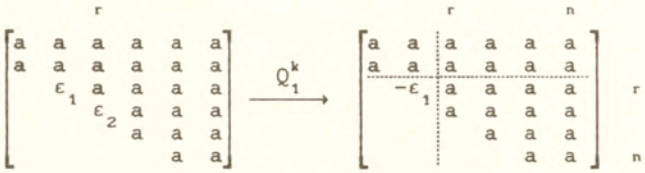
$$\eta_2 = \frac{a^k_{r+1,r}\, a^k_{20}}{\pm S} , \qquad S^2 = (a^k_{10})^2 + (a^k_{20})^2 .$$

Jeśli $a^k_{10} \neq 0$, wówczas, uwzględniając (11), otrzymuje się następujące oszacowanie elementu η_2:

$$|\eta_2| \leq \frac{|a^k_{r,r-1} \cdot a^k_{r+1,r}|}{|a^k_{rr} - \sigma_k b^k_{rr}|} .$$

Spełnienie warunku (10) oznacza zatem, że wartość elementu η_2

jest zaniedbywalnie mała, a zatem rozpoczęcie procedury iteracyjnej od kolumny r nie narusza uzyskanej uprzednio postaci Hessenberga macierzy A w obszarze oznaczonym indeksami 1,...,r-1. Uwzględniając zaniedbywalność η_2 można ponadto wykazać, że element η_1 przyjmuje wówczas wartość $-a_{r,r-1}(= -\varepsilon_1)$:



Przekształcenie Q_1^k nie narusza zer macierzy B w kolumnach 1,...,r-1.

Spełnienie warunku (10) umożliwia także dokonanie dalszej deflacji problemu "od góry", jeśli $b_{rr}^k = 0$ (lub jest zaniedbywalnie małe), bez naruszenia struktury Hessenberga macierzy A. W tym przypadku wykonuje się lewostronne przekształcenie Householdera Q_r^k zerujące element $a_{r+1,r}^k(= \varepsilon_2)$; nie narusza ono zer w pozycjach (r,r) oraz (r+1,r) macierzy B, wprowadza jednak niezerowy element x_2 w pozycji (r+1,r-1) macierzy A:

W analogiczny jak uprzednio sposób można wykazać, że jeśli $a_{rr} \neq 0$, element x_2 spełnia nierówność:

$$|x_2| \leq \frac{|\varepsilon_1 \varepsilon_2|}{|a_{rr}^k|},$$

a zatem jest on zaniedbywalnie mały, jeśli:

$$|\varepsilon_1\varepsilon_2| \le |a^k_{rr}| \cdot eps \cdot \|A\|_1 \quad ,$$

co oznacza spełnienie kryterium (10), w którym $b_{rr} = 0$. Ponadto, również w tym przypadku $x_1 = -\varepsilon_1$:

$$\begin{bmatrix} a & a & a & a & a & a \\ a & a & a & a & a & a \\ & \varepsilon_1 & a & a & a & a \\ & & \varepsilon_2 & a & a & a \\ & & & a & a & a \\ & & & & a & a \end{bmatrix} \xrightarrow{Q^k_r} \begin{bmatrix} a & a & a & a & a & a \\ a & a & a & a & a & a \\ & -\varepsilon_1 & a & a & a & a \\ & & 0 & a & a & a \\ & & & a & a & a \\ & & & & a & a \end{bmatrix} \begin{matrix} \\ \\ \\ r+1 \\ \\ n \end{matrix}$$

$\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad r+1 \qquad n$

Dalsze iteracje rozpoczyna się od kolumny i wiersza r+1, sprawdzając na wstępie warunek $b_{r+1,r+1} \neq 0$ itd.

Podsumowując, testowanie sąsiadujących elementów codiagonali macierzy A przebiega następująco:

1. Dla $r = 1,\ldots,n$ sprawdzenie warunku (10);
2. Jeśli warunek (10) jest spełniony dla $r = m$, wówczas:

2.1. jeśli $b^k_{mm} \neq 0$, rozpoczęcie iteracji od kolumny i wiersza m;

$a^k_{m,m-1} := -a^k_{m,m-1}$;

2.2. jeśli $b^k_{mm} = 0$, deflacja problemu w kolumnie m za pomocą lewostronnego przekształcenia Householdera Q^k_m zerującego element $a^k_{m+1,m}$, a następnie rozpoczęcie iteracji od kolumny i wiersza m+1;

$a^k_{m,m-1} := -a^k_{m,m-1}$.

Jeśli w trakcie procesu iteracyjnego przekształcane macierze ulegają podziałowi na bloki:

$$\left[\begin{array}{c|c} A_{11} & A_{12} \\ \hline 0 & A_{22} \end{array}\right], \qquad \left[\begin{array}{c|c} B_{11} & B_{12} \\ \hline 0 & B_{22} \end{array}\right],$$

to dla wyznaczenia wartości własnych ignoruje się bloki A_{12} oraz B_{12}, ograniczając przekształcenia do głównych bloków diagonalnych A_{11}, B_{11} oraz A_{22}, B_{22}; jeśli celem obliczeń jest również wyznaczenie wektorów własnych, wówczas transformacjom Q i Z muszą podlegać także bloki A_{12} oraz B_{12}.

IV.2. Iteracje QZ dla macierzy rzeczywistych

IV.2.1. *Metoda podwójnego kroku z niejawnymi przesunięciami*

W przypadku macierzy rzeczywistych strategia pojedynczego kroku iteracyjnego jest utrudniona, gdyż wartości własne tych macierzy mogą być zarówno liczbami rzeczywistymi, jak i zespolonymi. Jednak zespolone wartości własne występują tu zawsze parami, jako liczby sprzężone, co umożliwia zastosowanie techniki podwójnego kroku, pozwalającej na prowadzenie procesu iteracyjnego wyłącznie w arytmetyce rzeczywistej. Metoda **podwójnego kroku iteracyjnego z niejawnymi przesunięciami** została wprowadzona przez Francisa [13] i zastosowana w iteracjach QR macierzy rzeczywistej [1,2]. Moler i Stewart [7] rozszerzyli ją na przypadek uogólnionego zagadnienia własnego, stosując tę technikę w opracowanym przez siebie algorytmie QZ.

Procedura rozpoczyna się wykonaniem standardowego podwójnego kroku iteracyjnego QR, odniesionego do macierzy $C = AB^{-1}$. Polega on na zbudowaniu macierzy ortogonalnej $Q = Q_1Q_2$, przekształcającej wyjściową macierz Hessenberga C_1 ($C_1 \equiv C$) w ortogonalnie-podobną macierz Hessenberga C_3, według wzorów:

$$Q_1^T(C_1 - \sigma_1 I) = R_1,$$
$$C_2 = R_1Q_1 + \sigma_1 I,$$
$$Q_2^T(C_2 - \sigma_2 I) = R_2,$$
$$C_3 = R_2Q_2 + \sigma_2 I,$$

$$C_3 = (Q_1Q_2)^T C_1 (Q_1Q_2) = Q^T C_1 Q .$$

Tak określona macierz Q spełnia równanie:

$$M = QR \ ,$$

gdzie:

$$M = (C - \sigma_1 I)(C - \sigma_2 I),$$

$R = R_1 R_2$ jest macierzą trójkątną górną,

σ_1, σ_2 są liczbami określającymi zastosowane przesunięcia.

Macierz Q jest przy tym jednoznacznie określona (z dokładnością do znaków kolumn) przez swoją pierwszą kolumnę [8]; ponieważ realizuje ona rozkład trójkątny macierzy M, może być wyrażona w postaci iloczynu (n-1) macierzy Householdera:

$$Q = H_1 H_2 \ldots . H_{n-1} \ ,$$

gdzie wszystkie macierze H_i $(i = 2,\ldots,n-1)$ mają pierwszą kolumnę jednostkową. A zatem pierwsza kolumna macierzy Q (równa pierwszej kolumnie macierzy H_1) jest w pełni określona przez elementy pierwszej kolumny macierzy M, a mianowicie:

$$H_1 = I - (\alpha u) u^T \ ,$$

gdzie, zgodnie z (4):

$$u_1 = 1 \ ,$$

$$u_i = \frac{m_{i1}}{m_{11} \pm S} \ , \qquad i = 2,\ldots,n \ ,$$

$$\alpha = \frac{m_{11} \pm S}{\pm S} \ , \qquad S = \left(\sum_{i=1}^{n} |c_{i1}|^2 \right)^{1/2} \ .$$

Ponieważ $C = AB^{-1}$ jest macierzą Hessenberga, pierwsza kolumna macierzy M ("zerowa, fikcyjna kolumna" macierzy A) ma w rzeczywistości tylko trzy elementy niezerowe:

$$a_{10} \equiv m_{11}, \qquad a_{20} \equiv m_{21}, \qquad a_{30} \equiv m_{31} \ .$$

A zatem wektor Householdera określający macierz H_1 ma również tylko 3 elementy niezerowe: u_1, u_2, u_3; dobierane są one tak, by wyzerować elementy a_{20} oraz a_{30}.

Przy użyciu tak określonej macierzy Q buduje się następnie macierze QAZ i QBZ, z których pierwsza jest macierzą Hessenberga, druga - macierzą trójkątną górną. Mnożąc lewostronnie macierze A i B przez macierz $Q_1 = H_1$ otrzymujemy macierze Q_1A oraz Q_1B w postaci:

(12)
$$\begin{bmatrix} a & a & a & a & a & a \\ a & a & a & a & a & a \\ a & a & a & a & a & a \\ 0 & 0 & a & a & a & a \\ 0 & 0 & 0 & a & a & a \\ 0 & 0 & 0 & 0 & a & a \end{bmatrix} \qquad \begin{bmatrix} b & b & b & b & b & b \\ b'' & b & b & b & b & b \\ b' & b' & b & b & b & b \\ 0 & 0 & 0 & b & b & b \\ 0 & 0 & 0 & 0 & b & b \\ 0 & 0 & 0 & 0 & 0 & b \end{bmatrix} .$$

Przekształcenie spowodowało pojawienie się trzech niezerowych elementów poddiagonalnych macierzy B w pozycjach (2,1), (3,1) oraz (3,2). Elementy te usuwane są następnie za pomocą dwóch kolejnych przekształceń prawostronnych:

Z_1' - macierzy Householdera zerującej elementy b_{31} oraz b_{32};
Z_1'' - macierzy obrotu zerującej element b_{21}.

Po wykonaniu mnożenia macierz $Q_1AZ_1'Z_1''$ ma postać:

$$\begin{bmatrix} a & a & a & a & a & a \\ a & a & a & a & a & a \\ a' & a & a & a & a & a \\ a' & a & a & a & a & a \\ 0 & 0 & 0 & a & a & a \\ 0 & 0 & 0 & 0 & a & a \end{bmatrix} .$$

Następnie zerowane są elementy a_{31} i a_{41} za pomocą lewostronnego przekształcenia Householdera Q_2; uzyskane macierze $Q_2Q_1AZ_1'Z_1''$ i $Q_2Q_1BZ_1'Z_1''$ mają postać:

$$\begin{bmatrix} a & a & a & a & a & a \\ a & a & a & a & a & a \\ 0 & a & a & a & a & a \\ 0 & a & a & a & a & a \\ 0 & 0 & 0 & a & a & a \\ 0 & 0 & 0 & 0 & a & a \end{bmatrix} \qquad \begin{bmatrix} b & b & b & b & b & b \\ 0 & b & b & b & b & b \\ 0 & b'' & b & b & b & b \\ 0 & b' & b' & b & b & b \\ 0 & 0 & 0 & 0 & b & b \\ 0 & 0 & 0 & 0 & 0 & b \end{bmatrix} .$$

W ten sposób wyjściowy układ niezerowych elementów poddiagonalnych (12) w obu macierzach "przesunął" się niżej, do prawego bloku diagonalnego 5 x 5.

W analogiczny sposób zerowane są kolejne elementy poddiagonalne obu macierzy; proces kończy się nieco prostszym krokiem zerującym pojedynczy element $a_{n,n-2}$ za pomocą macierzy Q_{n-1}, a następnie element $b_{n,n-1}$ za pomocą macierzy Z'_{n-1}.

Ostatecznie otrzymujemy:

$$A := Q_{n-1}\cdots\left(Q_2\left(Q_1AZ'_1Z''_1\right)Z'_2Z''_2\right)\cdots Z'_{n-1} \quad ,$$

$$B := Q_{n-1}\cdots\left(Q_2\left(Q_1BZ'_1Z''_1\right)Z'_2Z''_2\right)\cdots Z'_{n-1} \quad ,$$

gdzie, oznaczając:

$$Q = Q_1Q_2\cdots\cdots Q_{n-1} \quad ,$$

$$Z = Z'_1Z''_1Z'_2Z''_2\cdots Z'_{n-1} \quad ,$$

macierze $A = Q^TAZ$ oraz $B = Q^TBZ$ mają ponownie uogólnioną postać Hessenberga.

Opisana procedura ilustruje technikę wykonania jednego (podwójnego) kroku iteracyjnego. Aby go zainicjować, należy wyznaczyć elementy "zerowej, fikcyjnej kolumny" macierzy A. Ponieważ są to trzy niezerowe elementy pierwszej kolumny macierzy $M = (C - \sigma_1 I)(C - \sigma_2 I)$, gdzie $C = AB^{-1}$, otrzymujemy:

$$a_{10} = c_{11}^2 - c_{11}(\sigma_1 + \sigma_2) + \sigma_1\sigma_2 + c_{12}c_{21} \; ;$$

$$a_{20} = c_{21}(c_{11} + c_{22} - \sigma_1 - \sigma_2) \; ;$$

$$a_{30} = c_{32}c_{21} \; .$$

Analogicznie do techniki stosowanej w standardowym algorytmie QR, jako przesunięcia σ_1 i σ_2 wybiera się wartości własne dolnego bloku diagonalnego 2 x 2 macierzy C, czyli pierwiastki równania charakterystycznego:

$$\det(\tilde{C} - \sigma I) = 0 \quad ,$$

gdzie

$$\tilde{C} = \begin{bmatrix} c_{n-1,n-1} & c_{n-1,n} \\ c_{n,n-1} & c_{nn} \end{bmatrix} \quad .$$

Zgodnie z definicją, wartości te spełniają zależności:

$$\sigma_1 + \sigma_2 = c_{n-1,n-1} + c_{nn} \quad ,$$

$$\sigma_1\sigma_2 = c_{n-1,n-1}c_{nn} - c_{n,n-1}c_{n-1,n} \quad .$$

Zastępując w powyższych wzorach elementy macierzy C odpowiednimi elementami macierzy AB^{-1} oraz dokonując niezbędnych przekształceń, otrzymamy $(m = n-1)$:

$$a_{10} = \left[\left(\frac{a_{mm}}{b_{mm}} - \frac{a_{11}}{b_{11}}\right)\left(\frac{a_{nn}}{b_{nn}} - \frac{a_{11}}{b_{11}}\right) - \left(\frac{a_{mn}}{b_{nn}}\right)\left(\frac{a_{nm}}{b_{mm}}\right) + \left(\frac{a_{nm}}{b_{mm}}\right)\left(\frac{b_{mn}}{b_{nn}}\right)\left(\frac{a_{11}}{b_{11}}\right)\right]\left(\frac{b_{11}}{a_{21}}\right) + \left(\frac{a_{12}}{b_{22}}\right) - \left(\frac{a_{11}}{b_{11}}\right)\left(\frac{b_{12}}{b_{22}}\right) \quad ,$$

$$a_{20} = \left(\frac{a_{22}}{b_{22}} - \frac{a_{11}}{b_{11}}\right) - \left(\frac{a_{21}}{b_{11}}\right)\left(\frac{b_{12}}{b_{22}}\right) - \left(\frac{a_{mm}}{b_{mm}} - \frac{a_{11}}{b_{11}}\right) + - \left(\frac{a_{nn}}{b_{nn}} - \frac{a_{11}}{b_{11}}\right) - \left(\frac{a_{nm}}{b_{mm}}\right)\left(\frac{b_{mn}}{b_{nn}}\right) \quad ,$$

$$a_{30} = \frac{a_{32}}{b_{22}} \quad . \qquad (13)$$

Dla wyznaczenia elementów a_{10}, a_{20}, a_{30} wymagane jest zatem spełnienie warunków:

$$b_{11} \neq 0, \quad b_{22} \neq 0 \quad \text{oraz} \quad b_{n-1,n-1} \neq 0, \quad b_{nn} \neq 0.$$

Jednak również w przypadku, gdy wartość któregoś z tych elementów nie jest dokładnie równa zeru, lecz jest zaniedbywal-

nie mała (w sensie występujących błędów zaokrągleń), sytuacja jest niekorzystna dla zbieżności procesu iteracyjnego, gdyż powoduje osłabienie efektu zastosowanych przesunięć. Jeśli np. element b_{11} lub b_{22} jest "mały", wówczas człony definiujące przesunięcia (zawierające elementy b_{nn}, $b_{n-1,n-1}$) stają się zaniedbywalne w porównaniu z pozostałymi członami. Podobnie negatywny wpływ ma stosowanie nieskończonych wartości przesunięć ($b_{n-1,n-1} \sim 0$ lub $b_{nn} \sim 0$).

Jeśli $b_{11} = 0$ (lub jest zaniedbywalnie małe), dokonuje się deflacji problemu "od góry" za pomocą lewostronnego przekształcenia obrotu Q_{12} zerującego element a_{21} (według schematu przedstawionego na str. 22), a następnie kontynuuje procedurę dla macierzy o zmniejszonym wymiarze.

Dla większości regularnych zagadnień własnych metoda QZ umożliwia wyeliminowanie przypadków $b_{nn} = 0$ oraz $b_{n-1,n-1} = 0$ już w trakcie wstępnego etapu redukcji macierzy A i B do uogólnionej postaci Hessenberga (opisanego w rozdziale III). Technika obliczeniowa zastosowana podczas tej redukcji powoduje, że wartości zerowe na diagonali B (jeśli istnieją) przemieszczają się wzdłuż diagonali w kierunku od dołu w górę [7,10]. W ten sposób "duże" wartości własne mają tendencję do gromadzenia się w lewym górnym rogu. Nie wyklucza to jednak możliwości zerowania się elementów b_{nn} lub $b_{n-1,n-1}$ w przypadkach szczególnych.

Istotnym problemem w algorytmie QZ z podwójnym przesunięciem jest przypadek $b_{22} = 0$, gdyż element ten nie może być wyeliminowany z zagadnienia poprzez deflację. Przypadek ten występuje przy tym bardzo często, gdy macierz B jest osobliwa, co wynika ze wspomnianej wyżej własności przesuwania się zer wzdłuż diagonali B na etapie wstępnej redukcji (wartość elementu b_{11} nie ulega zmianie podczas tej redukcji). Wprawdzie w praktyce numerycznej problem ten nie powoduje istotnych ograniczeń stosowalności algorytmu; ze względu na skończoną dokładność arytmetyki zmiennoprzecinkowej i błędy zaokrągleń, wartość elementu b_{22}

(jako wynik sekwencji poprzednich przekształceń) będzie w rzeczywistości zaniedbywalnie mała, lecz różna od zera. Obliczenia przebiegną więc na ogół bez zakłóceń, choć spowodowane osłabienie (lub wręcz anulowanie) efektu przesunięć może znacznie wydłużyć proces iteracyjny.

Strategia podwójnego kroku zastosowana w iteracjach QZ może być więc źródłem problemów numerycznych w przypadku, gdy B jest macierzą osobliwą, co znacznie ogranicza praktyczną użyteczność algorytmu.

Podsumowując omówioną do tej pory technikę obliczeniową, algorytm wykonania jednego podwójnego kroku iteracyjnego QZ jest następujący:

1. Wyznaczenie elementów a_{10}, a_{20}, a_{30} według (13) (poprzedzone deflacją problemu "od góry", jeśli $b_{11} = 0$);

2. Dla $j = 1,2,.....,n-2$:

   2.1. wyznaczenie macierzy Householdera Q_j takiej, by wyzerować elementy $a_{j+1,j-1}$ oraz $a_{j+2,j-1}$;

   $A := Q_j A$, $\quad B := Q_j B$.

   2.2. wyznaczenie macierzy Householdera Z'_j takiej, by wyzerować elementy $b_{j+2,j+1}$ oraz $b_{j+2,j}$;

   $B := BZ'_j$, $\quad A := AZ'_j$.

   2.3. wyznaczenie macierzy obrotu Z''_j takiej, by wyzerować element $b_{j+1,j}$;

   $B := BZ''_j$, $\quad A := AZ''_j$.

3. Wyznaczenie macierzy Householdera Q_{n-1} takiej, by wyzerować element $a_{n,n-1}$;

   $A := Q_{n-1}A$, $\quad B := Q_{n-1}B$.

4. Wyznaczenie macierzy obrotu Z_{n-1} takiej, by wyzerować element $b_{n,n-1}$;

$$B := BZ_{n-1} \quad , \qquad A := AZ_{n-1} \quad .$$

Prawidłowy przebieg procesu iteracyjnego charakteryzuje się tymi samymi własnościami, co w metodzie pojedynczego kroku, opisanej w rozdziale IV.1. W jego wyniku maleją wartości elementów codiagonali macierzy A, a elementy diagonalne określające wartości własne wydzielają się od dołu w górę. Po zakończeniu każdego k-tego kroku iteracyjnego sprawdzana jest wielkość elementów codiagonali macierzy A_k. Jeśli element $a^k_{n,n-1}$ jest dostatecznie mały według przyjętego kryterium, t.j. spełnia nierówność:

$$|a^k_{n,n-1}| \leq eps \cdot \|A\|_1 \quad ,$$

wówczas traktuje się go jako równy zeru. Elementy diagonalne a^k_{nn} oraz b^k_{nn} wyznaczają wydzieloną wartość własną $\lambda_n = a^k_{nn}/b^k_{nn}$ i dalsze iteracje prowadzone są dla bloków diagonalnych macierzy A i B, obejmujących wiersze i kolumny 1,...,n-1.

Jeśli zamiast elementu a^k_{nn} zaniedbywalnie mały staje się element $a^k_{n-1,n-2}$, wówczas (traktując go jako równy zeru) dalsze iteracje prowadzi się dla bloków diagonalnych obejmujących wiersze i kolumny 1,...,n-2; wartości własne λ_n oraz λ_{n-1} (rzeczywiste lub stanowiące sprzężoną parę zespoloną) są w tym przypadku pierwiastkami równania charakterystycznego:

$$\det(\tilde{A}_k - \lambda \tilde{B}_k) = 0 \quad ,$$

gdzie $\tilde{A}_k$, $\tilde{B}_k$ są dolnymi blokami diagonalnymi 2 x 2 macierzy A_k, B_k, obejmującymi wiersze i kolumny: n-1, n.

Jeśli dwa kolejne elementy poddiagonalne $a^k_{r,r-1}$ oraz $a^k_{r+1,r}$ (oznaczone jako ε_1 oraz ε_2) nie są zaniedbywalne każdy z osobna, lecz są "małe" w sensie spełnienia odpowiedniego kryterium, można dokonać deflacji macierzy, kontynuując proces iteracyjny od kolumny i wiersza r. W przypadku techniki podwójnego kroku kryterium to określone jest przez warunek [1]:

(14) $$|\varepsilon_1| \cdot (|\bar{a}_{20}| + |\bar{a}_{30}|) \leq |\bar{a}_{10}| \cdot eps \cdot \|A\|_1 \quad ,$$

gdzie:

$\bar{a}_{10}$, $\bar{a}_{20}$, $\bar{a}_{30}$ są trzema niezerowymi elementami "fikcyjnej zerowej kolumny" macierzy $\bar{A}_k$; wyznacza się je na podstawie wzorów (13), w których elementy a,b z indeksami 1,2 zastępowane są odpowiednio elementami a,b z indeksami r,r+1;

$\bar{A}_k$ jest dolnym blokiem diagonalnym macierzy A_k obejmującym wiersze i kolumny r,...,n:

$$\begin{array}{c} \\ \\ \\ \\ \\ \\ \end{array} \begin{array}{c} \begin{array}{cc} & r \end{array} \qquad\qquad \begin{array}{c} n \end{array} \\ \left[\begin{array}{cc:cccc} a & a & a & a & a & a \\ a & a & a & a & a & a \\ \hdashline & \varepsilon_1 & a & a & a & a \\ & & \varepsilon_2 & a & a & a \\ & & & a & a & a \\ & & & & a & a \end{array}\right] \end{array} \begin{array}{l} \\ \\ r \\ \\ \\ n \end{array} \left.\begin{array}{l} \\ \\ \\ \\ \\ \\ \end{array}\right\} \bar{A}_k$$

Po wykonaniu przekształcenia Householdera Q_1^k zerującego elementy $\bar{a}_{20}$ oraz $\bar{a}_{30}$(zmienia ono jedynie wartości elementów wierszy r, r+1 oraz r+2) otrzymuje się macierz $Q_1^k A_k$ w postaci:

$$\begin{array}{c} \begin{array}{cc} & r \end{array} \qquad\qquad \begin{array}{c} n \end{array} \\ \left[\begin{array}{cc:cccc} a & a & a & a & a & a \\ a & a & a & a & a & a \\ \hdashline & \eta_1 & a & a & a & a \\ & \eta_2 & a & a & a & a \\ & \eta_3 & a & a & a & a \\ & & & & a & a \end{array}\right] \end{array} \begin{array}{l} \\ \\ r \\ \\ \\ n \end{array}$$

gdzie, na podstawie (4):

$$\eta_1 = \varepsilon_1 \left(1 - \frac{\bar{a}_{10} \pm S}{\pm S}\right),$$

$$\eta_2 = \frac{\varepsilon_1 \bar{a}_{20}}{\pm S} \quad ,$$

$$\eta_3 = \frac{\varepsilon_1 \bar{a}_{30}}{\pm S} \quad , \qquad S^2 = (\bar{a}_{10})^2 + (\bar{a}_{20})^2 + (\bar{a}_{30})^2 \quad .$$

Jeśli $\bar{a}_{10} \neq 0$, wówczas otrzymuje się następujące oszacowanie:

$$|\eta_2| + |\eta_3| \leq \frac{|\varepsilon_1| \cdot (|\bar{a}_{20}| + |\bar{a}_{30}|)}{|\bar{a}_{10}|}$$

Spełnienie warunku (14) oznacza, że wartości elementów η_2 i η_3 są zaniedbywalnie małe, a zatem rozpoczęcie procedury iteracyjnej od kolumny r nie narusza uzyskanej uprzednio postaci Hessenberga macierzy A w kolumnach 1,...,r-1. Uwzględniając zaniedbywalność η_2 i η_3 można ponadto wykazać, że element η_1 przyjmuje wówczas wartość $-\varepsilon_1$:

$$\begin{bmatrix} a & a & a & a & a & a \\ a & a & a & a & a & a \\ & \varepsilon_1 & a & a & a & a \\ & & \varepsilon_2 & a & a & a \\ & & & a & a & a \\ & & & & a & a \end{bmatrix} \xrightarrow{Q_1^k} \begin{bmatrix} a & a & a & a & a & a \\ a & a & a & a & a & a \\ & -\varepsilon_1 & a & a & a & a \\ & & a & a & a & a \\ & & a & a & a & a \\ & & & & a & a \end{bmatrix} \begin{matrix} \\ \\ r \\ \\ \\ n \end{matrix} \quad \begin{matrix} \\ \\ \\ \\ \\ \end{matrix} \quad \begin{matrix} \\ \\ r \\ \\ \\ n \end{matrix} .$$

Wzorując się na strategii stosowanej w algorytmie QR z podwójnym przesunięciem [2,11], dla dziesiątej oraz dwudziestej iteracji zastosowano niestandardowe wartości przesunięć σ_k i σ_{k+1}, spełniające zależności:

$$\sigma_k + \sigma_{k+1} = 1.5 \cdot S \,, \qquad \sigma_k \sigma_{k+1} = S^2 \,,$$

gdzie:

$$S = \left| \frac{a^k_{n,n-1}}{b^k_{n-1,n-1}} \right| + \left| \frac{a^k_{n-1,n-2}}{b^k_{n-2,n-2}} \right|$$

(jeśli $b^k_{n-1,n-1} = 0$ lub $b^k_{n-2,n-2} = 0$, wówczas element ten zastępuje się wartością $eps \cdot \|B\|_1$).

Postępowanie to ma na celu uzyskanie zbieżności (poprzez zaburzenie procesu iteracyjnego) w przypadku, gdy redukowane macierze okażą się niezmiennicze względem zadanej strategii wyboru przesunięć.

IV.2.2. *Iteracje QZ z kombinowanym przesunięciem*

W rozdziale IV.2.1. wykazano, że przy zastosowaniu techniki podwójnego kroku iteracyjnego może nastąpić zakłócenie prawidłowego funkcjonowania algorytmu w przypadkach, gdy:

1) $b_{22} = 0$;

2) $b_{nn} = 0$ lub/oraz $b_{n-1,n-1} = 0$;

Sytuacje te mogą powodować problemy numeryczne (nieokreśloność formuł (13)) lub też (nawet przy niezakłóconym przebiegu obliczeń ze względu na występujące błędy zaokrągleń) znaczne osłabienie efektu przesunięć, co jest niekorzystne dla zbieżności procesu iteracyjnego.

Powyższe przypadki nie powodują natomiast trudności obliczeniowych w algorytmie QZ z pojedynczym przesunięciem; przypadek $b_{22} = 0$ nie ma tam w ogóle wpływu na przebieg obliczeń, natomiast przypadki $b_{nn} = 0$, $b_{n-1,n-1} = 0$ implikują jedynie inny sposób wyznaczenia aktualnej wartości przesunięcia (wzory (7), (8),(9)), bez naruszenia jednolitości schematu iteracyjnego.

Ponadto, metoda podwójnego kroku iteracyjnego jest bardzo czasochłonna, przede wszystkim ze względu na procedurę wykrywania dwóch "małych" sąsiadujących ze sobą elementów poddiagonalnych macierzy A za pomocą kryterium (14). Obliczaniu (w każdym kroku iteracyjnym) trzech elementów "zerowej, fikcyjnej kolumny" według (13) musi towarzyszyć testowanie odpowiednich elementów diagonali macierzy B w kolumnach 2,...,n-1, co wymaga znacznej liczby operacji. W większości przypadków, czas pochłonięty przez te operacje niweluje korzyści wynikające z faktu jednoczesnego wykonania dwóch kroków iteracyjnych [10].

Wiadomo również, że użycie metody podwójnego kroku jest uzasadnione jedynie w przypadku, gdy wyznaczone wartości przesunięć są liczbami zespolonymi, co nie zawsze ma miejsce.

Znacznie efektywniejszą techniką iteracyjną w przypadku zagadnienia z macierzami rzeczywistymi jest metoda iteracji QZ

z kombinowanym przesunięciem, opracowana przez Warda [10]. Umożliwia ona wybór strategii pojedynczego lub podwójnego kroku na początku każdej kolejnej iteracji, w zależności od typu aktualnych wartości przesunięć (rzeczywiste lub zespolone), jak również aktualnych wartości elementów diagonalnych b_{22} oraz b_{nn} i $b_{n-1,n-1}$. Postępowanie to pozwala wyeliminować trudności związane z techniką podwójnego kroku, zmniejszając jednocześnie czasochłonność obliczeń.

Schemat wykonania każdego kolejnego kroku iteracyjnego jest tu następujący:

1. Jeśli $b_{nn} \neq 0$ oraz $b_{n-1,n-1} \neq 0$:

   1.1. sprawdzenie, czy wartości własne dolnego bloku diagonalnego 2 x 2 macierzy AB^{-1} (określające przesunięcia σ_1, σ_2) są rzeczywiste, czy zespolone;

   1.2. jeśli σ_1, σ_2 są zespolone:

      (a) jeśli $b_{22} \neq 0$, wykonanie podwójnego kroku iteracyjnego z przesunięciami σ_1, σ_2;

      (b) jeśli $b_{22} = 0$, wykonanie pojedynczego kroku iteracyjnego z przesunięciem σ równym części rzeczywistej pary wartości sprzężonych σ_1, σ_2 ($\sigma = \mathrm{Re}(\sigma_1) = \mathrm{Re}(\sigma_2)$);

   1.3. jeśli σ_1, σ_2 są rzeczywiste, wykonanie pojedynczego kroku iteracyjnego z przesunięciem σ równym tej wartości spośród σ_1, σ_2, która jest bliższa wartości a_{nn}/b_{nn};

2. Jeśli $b_{nn} = 0$ lub/oraz $b_{n-1,n-1} = 0$, wykonanie pojedynczego kroku iteracyjnego z przesunięciem σ, wyznaczonym według odpowiedniego wzoru (7), (8), (9), lub też przesunięciem $\sigma = 0$, jeśli mianownik w aktualnym wzorze jest równy zeru.

V. OSTATECZNA REDUKCJA MACIERZY A DO POSTACI TRÓJKĄTNEJ

W wyniku opisanych do tej pory etapów algorytmu otrzymujemy: macierz B w postaci trójkątnej górnej oraz macierz A w postaci quasi-trójkątnej górnej, w której przynajmniej co drugi element dolnej codiagonali jest równy zeru. Postać ta umożliwia już wyznaczenie wszystkich wartości własnych na podstawie bloków diagonalnych 1 x 1 oraz 2 x 2 obu macierzy. Wartości własne bloków 1 x 1 są po prostu ilorazami odpowiadających sobie elementów diagonalnych a_{ii}/b_{ii}. Wartości własne bloków 2x2 (oznaczonych jako $\tilde{A},\tilde{B}$) są pierwiastkami równania kwadratowego

$$\det(\tilde{A} - \lambda\tilde{B}) = 0 \tag{15}$$

i w przypadku macierzy rzeczywistych mogą być zarówno rzeczywiste, jak i zespolone.

Zamiast rozwiązywać równanie (15), można jednak dokonać dalszej redukcji zagadnienia 2 x 2 do dwóch zagadnień 1 x 1, w celu wyodrębnienia elementów diagonalnych. Metoda ta jest korzystniejsza co najmniej z dwóch powodów:

- elementy diagonalne zawierają bogatszą informację o wartości własnej niż sama wartość; na przykład, jeśli oba elementy a_{ii} i b_{ii} są małe, wyznaczona przez nie wartość własna λ_i jest źle uwarunkowana, niezależnie od jej wartości bezwzględnej;
- wyodrębnienie wartości własnych w blokach 1 x 1 znacznie upraszcza procedurę wyznaczania wektorów własnych.

W opracowanych algorytmach zastosowano metodę redukcji poszczególnych bloków diagonalnych 2 x 2 za pomocą przekształceń unitarnych, omówioną w pracy Molera i Stewarta [7]. Zostanie ona zilustrowana na przykładzie bloków:

$$\tilde{A} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}, \qquad \tilde{B} = \begin{bmatrix} b_{11} & b_{12} \\ 0 & b_{22} \end{bmatrix}.$$

W szczególnych przypadkach, gdy $b_{11} = 0$ lub $b_{22} = 0$, redukcja polega na wyzerowaniu elementu a_{21} za pomocą przekształcenia obrotu nie naruszającego zer w macierzy $\tilde{B}$. Uzyskuje się to przez zastosowanie:

- obrotu lewostronnego Q, jeśli $b_{11} = 0$:
  $\tilde{A} \longrightarrow Q\tilde{A}, \quad \tilde{B} \longrightarrow Q\tilde{B};$
- obrotu prawostronnego Z, jeśli $b_{22} = 0$:
  $\tilde{A} \longrightarrow \tilde{A}Z, \quad \tilde{B} \longrightarrow \tilde{B}Z.$

W ogólnym przypadku procedura anihilacji elementu a_{21} z jednoczesnym zachowaniem zerowej wartości elementu b_{21} jest nieco bardziej złożona i polega na zastosowaniu następujących operacji:

1. wyznaczenie jednej z wartości własnych λ (tej, dla której nie zachodzi odejmowanie się cyfr znaczących) jako pierwiastka równania kwadratowego (15);

2. wyznaczenie macierzy obrotu Z takiej, by wyzerować pierwszy element tego wiersza macierzy $E = (\tilde{A} - \lambda\tilde{B})$, który ma większą normę, a zatem:

element e_{11} jeśli $\|e_1\| > \|e_2\|$,

element e_{21} jeśli $\|e_1\| < \|e_2\|$,

gdzie e_1 i e_2 oznaczają pierwszy i drugi wiersz macierzy E;

3. wyznaczenie macierzy obrotu Q takiej, by wyzerować element (2,1) tej z dwóch macierzy $\tilde{A}Z$ i $\lambda\tilde{B}Z$, która ma mniejszą normę, a zatem:

element a_{21} jeśli $\|\tilde{A}Z\|_1 < |\lambda| \cdot \|\tilde{B}Z\|_1$,

element b_{21} jeśli $\|\tilde{A}Z\|_1 > |\lambda| \cdot \|\tilde{B}Z\|_1$.

Analiza powyższej procedury przeprowadzona w pracy [7] na podstawie oszacowań błędów zaokrągleń prowadzi do wniosku, że w obu uzyskanych macierzach $Q\tilde{A}Z$ oraz $Q\tilde{B}Z$ elementy znajdujące się

w pozycji (2,1) są zaniedbywalnie małe, a zatem macierze te są trójkątne górne. Istotnym warunkiem stabilności procedury jest, by element a_{21} wyjściowej macierzy $\tilde{A}$ nie był zaniedbywalnie mały, tzn. by spełniony był warunek:

$$|a_{21}| > eps \cdot \|\tilde{A}\|_1 ,$$

co jest zapewnione w wyniku testowania elementów codiagonali w trakcie procedury iteracyjnej.

W praktyce przekształcenia Q i Z macierzy A i B na etapie końcowej redukcji do postaci trójkątnej wykonywane są w przypadku zespolonym oraz w tym przypadku rzeczywistym, gdy blokowi diagonalnemu 2 x 2 odpowiadają rzeczywiste wartości własne. Jeśli rzeczywistemu blokowi 2 x 2 odpowiada para wartości zespolonych sprzężonych, wówczas powyższa procedura pozwala wyznaczyć (w arytmetyce zespolonej) odpowiadające elementy diagonalne, lecz transformacje macierzy nie są wykonywane. Zatem końcową postacią macierzy A w przypadku rzeczywistym jest postać quasi-trójkątna, w której każdemu blokowi diagonalnemu 2 x 2 odpowiada para zespolonych wartości własnych.

VI. WYZNACZANIE WEKTORÓW WŁASNYCH

Wektory własne zredukowanego zagadnienia z macierzami $\hat{A} = QAZ$ oraz $\hat{B} = QBZ$ obliczane są uogólnioną metodą "podstawienia wstecznego"[1]. Metoda ta, zastosowana w procedurze *fi* standardowego algorytmu QR [2,11] jest bardzo efektywną techniką obliczania wektorów własnych w przypadku, gdy celem obliczeń jest wyznaczenie pełnego zestawu tych wektorów.

W przypadku uogólnionego zagadnienia własnego, wektory własne obliczane są jako rozwiązania szczególne układów równań jednorodnych:

---

[1] *ang.* "backsubstitution".

$(\hat{A} - \lambda_i \hat{B})y_i = 0 \qquad i = 1,\ldots,n \qquad$ (wektory prawe),

$v_i^T(\hat{A} - \lambda_i \hat{B}) = 0 \qquad i = 1,\ldots,n \qquad$ (wektory lewe).

Elementy wektorów y_i oraz v_i będą dalej oznaczane jako y_{ji} oraz v_{ji} $(j = 1,\ldots,n)$.
Ponadto, dla uproszczenia notacji, elementy macierzy $\hat{A}$ i $\hat{B}$ oznaczane będą jako a_{ij} oraz b_{ij}.

W przypadku zespolonym, gdy obie macierze $\hat{A}$ i $\hat{B}$ są trójkątne, składowe prawego wektora własnego y_i odpowiadającego wartości własnej λ_i są następujące:

$$y_{ji} = \frac{-\sum_{k=j+1}^{i} (a_{jk} - \lambda_i b_{jk}) \cdot y_{ki}}{(a_{jj} - \lambda_i b_{jj})}, \qquad j = 1,\ldots,i-1,$$

$$y_{ii} = 1,$$

(16)

$$y_{ji} = 0, \qquad j = i+1,\ldots,n.$$

Analogicznie, składowe lewego wektora własnego v_i określone są jako:

$$v_{ji} = 0, \qquad j = 1,\ldots,i-1,$$

$$v_{ii} = 1,$$

(17)

$$v_{ji} = \frac{-\sum_{k=i}^{j-1} (a_{kj} - \lambda_i b_{kj}) \cdot v_{ki}}{(a_{jj} - \lambda_i b_{jj})}, \qquad j = i+1,\ldots,n.$$

Elementy $y_{ii} = 1$ oraz $v_{ii} = 1$ zadaje się z góry wybierając tym samym rozwiązania szczególne układów (16) i (17).

Jeśli $a_{jj} - \lambda_i b_{jj} = 0$, wówczas mianowniki w powyższych wzorach zastępuje się wartością równą $\max(eps \cdot \|A\|_1, eps|\lambda_i| \cdot \|B\|_1)$.

Bardziej skomplikowane są obliczenia w przypadku rzeczywistym, gdy nie wszystkie wartości własne są rzeczywiste. Wówczas zredukowana macierz $\hat{A}$ zawiera bloki diagonalne 2 x 2 odpowiadające parom wartości zespolonych sprzężonych. Wektory własne odpowiadające tym wartościom są również zespolone i parami sprzężone. Jednak i w tym przypadku technika "podstawienia wstecznego" umożliwia wyznaczenie pełnego zestawu wektorów własnych.

Jeśli wektor y_i odpowiada rzeczywistej wartości własnej λ_i (tzn. jest wektorem rzeczywistym), a blok diagonalny 2 x 2 określający parę zespolonych wartości własnych znajduje się w wierszach i kolumnach j, j+1, wówczas składowe y_{ji} oraz $y_{j+1,i}$ oblicza się z układu dwóch równań, utworzonych przez rozpisanie układu (16):

$$(a_{jj} - \lambda_i b_{jj})y_{ji} + (a_{j,j+1} - \lambda_i b_{j,j+1})y_{j+1,i} =$$

$$= - \sum_{k=j+2}^{i} (a_{jk} - \lambda_i b_{jk}) \cdot y_{ki} \quad , \tag{18a}$$

$$a_{j+1,j}\, y_{ji} + (a_{j+1,j+1} - \lambda_i b_{j+1,j+1})y_{j+1,i} =$$

$$= - \sum_{k=j+2}^{i} (a_{j+1,k} - \lambda_i b_{j+1,k}) \cdot y_{ki} \tag{18b}$$

Analogiczny układ równań dla wyznaczenia składowych v_{ji} oraz $v_{j+1,i}$ lewego wektora własnego ma postać:

$$(a_{jj} - \lambda_i b_{jj})v_{ji} + a_{j+1,j}\, v_{j+1,i} = - \sum_{k=i}^{j-1} (a_{kj} - \lambda_i b_{kj}) \cdot v_{ki} \quad ,$$

(19a)

$$(a_{j,j+1} - \lambda_i b_{j,j+1})\, v_{ji} + (a_{j+1,j+1} - \lambda_i b_{j+1,j+1}) v_{j+1,i} =$$

$$= - \sum_{k=i}^{j-1} (a_{k,j+1} - \lambda_i b_{k,j+1}) \cdot v_{ki} \qquad (19b)$$

Pozostałe składowe wektorów y_i oraz v_i obliczane są ze wzorów (16) i (17).

Jeśli wektory y_i oraz y_{i+1} odpowiadają parze zespolonych wartości własnych $\lambda_i = \lambda^*_{i+1}$, wówczas wyznacza się ten wektor własny, który odpowiada wartości własnej z dodatnią częścią urojoną (drugi wektor jest po prostu wektorem sprzężonym).

Przy założeniu, że $y_{i+1,i} = i$ (jedność urojona) druga składowa y_{ii} spełnia równania:

$$\begin{aligned} &(a_{ii} - \lambda_i b_{ii}) y_{ii} + i(a_{i,i+1} - \lambda_i b_{i,i+1}) = 0 \quad , \\ &a_{i+1,i} y_{ii} + i(a_{i+1,i+1} - \lambda_i b_{i+1,i+1}) = 0 \end{aligned} \qquad (20)$$

i jest określana z tego równania, dla którego współczynnik stojący przy y_{ii} jest większy (zapewnia to większą stabilność rozwiązania).

Analogiczne równania dla wyznaczenia składowej $v_{i+1,i}$ lewego wektora własnego, przy założeniu $v_{ii} = 1$, mają postać:

$$\begin{aligned} &(a_{ii} - \lambda_i b_{ii}) + a_{i+1,i} v_{i+1,i} = 0 \quad , \\ &(a_{i,i+1} - \lambda_i b_{i,i+1}) + (a_{i+1,i+1} - \lambda_i b_{i+1,i+1}) v_{i+1,i} = 0 \end{aligned} \qquad (21)$$

i składowa ta wyznaczana jest również z równania o większym współczynniku stojącym przy $v_{i+1,i}$.

Pozostałe składowe zespolonych wektorów własnych y_i oraz v_i wyznaczane są ze wzorów (16) i (18) oraz (17) i (19) z uwzględnieniem, że λ_i ma w tym przypadku wartość zespoloną.

Opisana metoda wyznaczania prawych i lewych wektorów własnych dotyczy tych wektorów, które odpowiadają skończonym wartościom własnym. Aby opis uogólnionego zagadnienia własnego był pełny, wektory własne przypisuje się również wartościom własnym równym "nieskończoności", przyjmując, że są to niezerowe wektory z podprzestrzeni zerowej macierzy $\hat{B}$, t.j. rozwiązania szczególne równań:

$$\hat{B}y = 0 \qquad \text{oraz} \qquad v^T\hat{B} = 0 \,. \tag{21}$$

Składowe wektorów prawych określa się w tym przypadku według wzorów:

$$y_{ji} = \frac{-\sum_{k=j+1}^{i} b_{jk}y_{ki}}{b_{jj}} \,, \qquad j = 1,\ldots,i-1,$$

$$y_{ii} = 1 \,,$$

$$y_{ji} = 0 \,, \qquad j = i+1,\ldots,n.$$

natomiast wektorów lewych:

$$v_{ji} = 0 \,, \qquad j = 1,\ldots,i-1,$$

$$v_{ii} = 1 \,,$$

$$v_{ji} = \frac{-\sum_{k=i}^{j-1} b_{kj}v_{jk}}{b_{jj}} \,, \qquad j = i+1,\ldots,n.$$

Natomiast dla wartości własnych równych zeru ($\lambda_i = 0$) wzory (16) i (17) opisują wektory własne z podprzestrzeni zerowej macierzy $\hat{A}$, t.j. spełniające równania: $\hat{A}y = 0$ oraz $v^T\hat{A} = 0$.

Przekształcenie wyznaczonych wektorów własnych (y,v) zagadnienia z macierzami trójkątnymi (lub quasi-trójkątnymi) (AZ i

QBZ w wektory własne zagadnienia wyjściowego (x,u) dokonywane jest za pomocą macierzy transformacji Q i Z, według zależności:

$$
\begin{aligned}
&x = Zy \qquad \text{(wektory prawe)},\\
&u^T = v^T Q \qquad \text{(wektory lewe)}.
\end{aligned}
\tag{22}
$$

Jeśli uogólnione zagadnienie własne posiada pełny zestaw wartości i wektorów własnych (co oznacza, że ma ono tylko liniowe dzielniki elementarne), lewe i prawe wektory spełniają następujące warunki biortogonalności:

$$
\begin{aligned}
&y_i^T A x_j = \bar{\alpha}_i \delta_{ij} ,\\
&y_i^T B x_j = \bar{\beta}_i \delta_{ij} ,
\end{aligned}
\qquad i = 1,..,n ,
$$

gdzie para wartości $(\bar{\alpha}_i, \bar{\beta}_i)$ określa i-tą wartość własną $\lambda_i = \bar{\alpha}_i / \bar{\beta}_i$ (uzyskane przy powyższej diagonalizacji wartości elementów $\bar{\alpha}_i, \bar{\beta}_i$ są określone sposobem normowania wektorów x,y).

VII. ORGANIZACJA PROCEDUR NUMERYCZNYCH

Opracowane procedury numeryczne realizujące algorytm QZ zostały zgrupowane w postaci sześciu niezależnych pakietów obliczeniowych (units), po trzy dla przypadku rzeczywistego (QZVALR, QZVECR, QZEIGR) oraz zespolonego (QZVALC, QZVECC, QZEIGC). Pakiety QZVALR i QZVALC służą do wyznaczania tylko wartości własnych, pakiety QZVECR i QZVECC - wartości i prawych wektorów własnych, natomiast pakiety QZEIGR i QZEIGC - wartości oraz prawych i lewych wektorów własnych. Podział ten wynika z innej (w każdym z trzech powyższych przypadków) gospodarki pamięcią komputerową, niezbędną do obliczeń oraz zapamiętania wyników.

Inicjalizacja obliczeń następuje poprzez wywołanie w programie głównym funkcji qz (typu logicznego), zawartej w części

INTERFACE wszystkich pakietów. Procedury realizujące poszczególne fragmenty algorytmu zawarte są w części IMPLEMENTATION.

Funkcja logiczna check0 wywoływana jest na samym początku obliczeń i służy do kontroli postaci wyjściowej pary macierzy (A,B).

Etap wstępnej redukcji, czyli przekształcenie pary macierzy do uogólnionej postaci Hessenberga, realizują procedury orttr i orthes (przypadek rzeczywisty) oraz unittr i unithes (przypadek zespolony); procedury te realizowane są opcyjnie w zależności od uzyskanej wartości funkcji check0.

We wszystkich pakietach programów iteracje QZ realizuje procedura iter (ogólny schemat budowy tej procedury wzorowany jest na analogicznych fragmentach funkcji *hqr*, zawartej w pakietach REALEIGEN i COMPEIGEN, opisanych w [11]). W pakietach dla zagadnienia z macierzami zespolonymi (QZVALC,QZVECC,QZEIGC) iteracje przebiegają według strategii pojedynczego kroku; sprawdzanie wielkości elementów poddiagonalnych macierzy A_k oraz wyznaczanie aktualnych dla danego kroku iteracyjnego elementów "fikcyjnej, zerowej kolumny" (poprzedzone ewentualną deflacją problemu) odbywa się w procedurze contsub, natomiast każdy pojedynczy krok iteracyjny wykonywany jest procedurą step. W pakietach dla zagadnienia z macierzami rzeczywistymi (QZVALR,QZVECR,QZEIGR) iteracje przebiegają według algorytmu z kombinowanym przesunięciem; zadania analogiczne do realizowanych w procedurze contsub spełniają tu procedury contsub1 (krok pojedynczy) oraz contsub2 (krok podwójny). Każdy pojedynczy krok iteracyjny wykonuje procedura step1, natomiast krok podwójny - procedura step2.

W procedurze iter wykonywana jest również na bieżąco ostateczna redukcja macierzy A do postaci trójkątnej, w miarę wyodrębniania się poszczególnych bloków diagonalnych 2 x 2 w trakcie procesu iteracyjnego.

Programy przewidują bezwzględne przerwanie procesu iteracyjnego (sygnalizowane jest to wartością funkcji qz = false), jeśli dana wartość własna nie wydzieliła się do momentu przekroczenia zadanej liczby iteracji (itm).

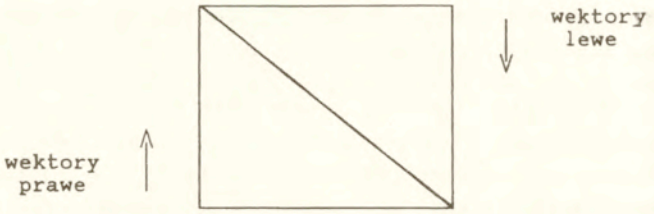
W procedurze iter wartości własne wydzielają się w postaci par wartości (α_i,β_i) odpowiadających sobie elementów diagonalnych macierzy trójkątnych QAZ i QBZ. Jeśli $\beta_i = 0$, element ten zastępowany jest wielkością $eps \cdot \|B\|_1$, w celu umożliwienia numerycznej reprezentacji także "nieskończonych" wartości własnych. W pakietach QZVALR i QZVALC wykonanie dzielenia α_i/β_i dla uzyskania wartości własnych pozostawione jest użytkownikowi programu. W pakietach QZVECR, QZVECC oraz QZEIGR, QZEIGC wartości własne α_i/β_i są obliczane podczas wyznaczania wektorów własnych; dlatego też zostały one wyprowadzone na wyjściu funkcji qz, niezależnie od wartości elementów diagonalnych α_i,β_i.

W pakietach przeznaczonych do wyznaczania wektorów własnych procedury orttr, orthes (unittr, unithes) oraz iter zapamiętują kolejne macierze transformacji, budując systematycznie pełne macierze przekształcenia Z (wektory prawe) oraz Q (wektory lewe).

W pakietach QZVECR, QZVECC, QZEIGR, QZEIGC obliczanie wektorów własnych oraz ich przekształcenie do wektorów zagadnienia wyjściowego realizuje procedura fin (jest ona wzorowana na odpowiednim fragmencie procedury *hqr2* z algorytmu II.15 w [2] oraz jego pascalowskiej wersji, rozbudowanej dla uwzględnienia lewych wektorów własnych, zawartej w pakietach REALEIGEN i COMPEIGEN [11]). Wynikiem działania procedury fin są *nieunormowane* wektory własne; sposób ich normowania pozostawiony jest użytkownikowi.

Ponieważ każdy i-ty wektor własny zagadnienia z macierzami trójkątnymi posiada (n-i) składowych rownych zeru (wzory (16), (17)), pozwala to na oszczędny zapis obliczanych wektorów w pamięci komputera. Poniższy rysunek przedstawia kolejność wyznaczania wektorów własnych oraz sposób ich zapamiętywania w

tablicy dwuwymiarowej:



Przy zachowaniu tej kolejności obliczeń do zapisu może być przeznaczona jedna z tablic zawierających elementy przekształconych macierzy A lub B (kolejne wektory zapisywane są na miejscu tych wierszy macierzy, które nie są wykorzystane przy obliczaniu następnych wektorów).

Jeśli w przypadku zagadnienia rzeczywistego wektory y_i (prawe) oraz v_i (lewe) są zespolone (tzn. odpowiadają zespolonej wartości własnej $\lambda_i = \lambda^*_{i+1}$), wówczas w wierszu i - tym zostają zapamiętane części rzeczywiste tych wektorów: $Re(y_i)$ i $Re(v_i)$, natomiast w wierszu i+1 - części urojone: $Im(y_i)$ i $Im(v_i)$. Wektory y_i oraz v_i mają wówczas (zgodnie z (16) i (17)) elementy zerowe:

$$y_{ji} = 0 \quad \text{dla} \quad j > i+1 \quad \text{oraz} \quad v_{ji} = 0 \quad \text{dla} \quad j < i.$$

W tym przypadku zapis wektorów według przedstawionego wyżej schematu wymaga oddzielnego zapamiętania elementów zawartych w bloku diagonalnym 2x2 o indeksach i, i+1.

Procedura przekształcenia wektorów własnych zagadnienia z macierzami trójkątnymi do wyjściowego układu współrzędnych, według zależności (22), wykorzystuje (w odpowiedniej kolejności) obszary pamięci zawarte w tablicach B, A, Z, Q i nie wymaga rezerwowania dodatkowej pamięci roboczej. Prawe wektory własne umieszczane są w kolumnach 1,..,n macierzy A, natomiast lewe wektory - w wierszach 1,..,n macierzy B.

VIII. WYNIKI TESTÓW NUMERYCZNYCH

W tablicach 1 - 3 podane są przykładowe czasy rozwiązywania poszczególnych fragmentów uogólnionego zagadnienia własnego, uzyskane podczas testowania opracowanych pakietów procedur na niehermitowskich macierzach Ortegi stopnia n = 20, 50, 90.

Macierze te tworzy się za pomocą przekształcenia przez podobieństwo macierzy diagonalnej $D = diag(d_1,\ldots,d_n)$, zadając przy tym w określony sposób wzajemnie ortogonalne prawe i lewe wektory własne **x**,**y**. Dla parzystych stopni n, elementy macierzy wyznacza się na podstawie zależności [11]:

$$a_{ij} = d_i\delta_{ij} \pm c(d_i - d_j + \sigma) ,$$

gdzie $\sigma = \mathbf{y}^T\mathbf{D}\mathbf{x}$, c jest współczynnikiem dobieranym tak, aby uzyskać zadane, jednakowe dla wszystkich wartości własnych wskaźniki uwarunkowania $s_i = |\mathbf{y}_i^T\mathbf{x}|$. Jako macierze o znanych, zadanych z góry własnościach macierze Ortegi są często wykorzystywane dla testowania algorytmów algebry liniowej.

Podane czasy odpowiadają realizacji programów w Turbo Pascalu 4.0 na mikrokomputerze IBM 486 (25 MHz). W każdej rubryce tablic podano 2 czasy: górny (R) dotyczy pakietów dla macierzy rzeczywistych, dolny (C) - dla zespolonych.

Tabela 1

Czasy obliczania wartości własnych pakietami procedur QZVALR / QZVALC

n →		20	50	90
ortr + orthes	R	0.2"	1.7"	10.2"
unittr + unithes	C	0.4"	5.2"	31.0"
iter	R	0.2"	2.3"	11.0"
	C	0.9"	9.6"	46.6"
qz	R	0.4"	4.0"	21.2"
	C	1.3"	14.8"	1'17.8"

Tabela 2

Czasy obliczania wartości i prawych wektorów własnych pakietami procedur QZVECR / QZVECC

n ⇒		20	50	90
ortr + orthes	R	0.2"	2.2"	13.2"
unittr + unithes	C	0.4"	6.7"	39.8"
iter	R	0.4"	4.0"	20.8"
	C	1.6"	18.8"	1'36.9"
fin	R	0.1"	0.5"	3.0"
	C	0.1"	1.2"	6.9"
qz	R	0.6"	6.8"	37.1"
	C	2.2"	26.8"	2'24.0"

Tabela 3

Czasy rozwiązania pełnego zagadnienia własnego (wartości oraz prawe i lewe wektory własne) pakietami procedur QZEIGR / QZEIGC

n ⇒		20	50	90
ortr + orthes	R	0.2"	3.0"	18.1"
unittr + unithes	C	0.6"	9.1"	53.5"
iter	R	0.5"	6.0"	31.3"
	C	2.4"	29.9"	2'33.0"
fin	R	0.1"	1.1"	6.0"
	C	0.2"	2.4"	14.0"
qz	R	0.8"	10.3"	55.6"
	C	3.2"	41.6"	3'40.9"

(czasy mierzono przy użyciu procedur Time0 oraz Time z pakietu CZAS, opisanych w [12]).

Z praktycznego punktu widzenia najbardziej miarodajne są czasy realizacji funkcji qz, obejmujące całość obliczeń.

Dla porównania, w Tabeli 4 zamieszczono całkowity czas rozwiązania pełnego zagadnienia własnego $(B^{-1}A)x = \lambda x$ (z zespolonymi macierzami Ortegi) zwykłą metodą QR; obejmuje on obliczanie macierzy $B^{-1}A$ oraz wykonywanie procedury *qr2* z pakietu COMPEIG [12]:

Tabela 4

n →		20	50	90
$B^{-1}A$ + *qr2*	C	1.0"	12.5"	1'08.5"

Analiza powyższych wyników prowadzi do następujących istotnych wniosków:

1. najbardziej czasochłonnym fragmentem obliczeń jest procedura iter, realizująca proces iteracyjny QZ; w pakietach przeznaczonych do wyznaczania również wektorów własnych ma ona dominujący wpływ na całkowity czas obliczeń, ponieważ wówczas w procesie iteracyjnym:

- zapamiętywane są kolejne macierze transformacji Z (wektory prawe) lub Q i Z (wektory lewe i prawe);
- przekształceniom podlegają całe macierze A,B, a nie tylko bloki diagonalne;

2. rozszerzenie obliczeń o prawe, a następne lewe wektory własne powoduje każdorazowo zwiększenie całkowitego czasu obliczeń o ok. 50%;

3. rozwiązanie pełnego zagadnienia własnego $Ax = \lambda Bx$ opracowaną procedurą QZ jest ok. 3-krotnie bardziej czasochłonne niż rozwiązanie (jeśli B jest macierzą nieosobliwą) równoważnego zagadnienia $B^{-1}Ax = \lambda x$ zwykłą procedurą QR.

Działanie algorytmów zilustrowano na kilku poniższych przykładach testów numerycznych.

Przykład 1. Macierze A,B są niesymetrycznymi rzeczywistymi macierzami Ortegi stopnia 6.

Elementy diagonalne d_i macierzy A i B zadano w postaci ciągów:

$$d_{Ai} = 2i - 1 , \qquad d_{Bi} = i , \qquad i = 1,\ldots,n ,$$

generując obie macierze ze współczynnikiem $s_i = 10$. Jest to dobrze uwarunkowany problem własny, stanowiący bezpośredni test algorytmu.

Dokładne wartosci własne $\lambda_i = d_{Ai}/d_{Bi}$ wynoszą:

1.00000000000000E+0000
1.50000000000000E+0000
1.66666666666667E+0000
1.75000000000000E+0000
1.80000000000000E+0000
1.83333333333333E+0000

W wyniku obliczeń otrzymano:

α_i:	β_i:
3.00000000000002E+0000	2.00000000000001E+0000
1.00000000000000E+0000	9.99999999999999E-0001
5.00000000000002E+0000	3.00000000000001E+0000
6.99999999999998E+0000	3.99999999999999E+0000
1.10000000000000E+0001	5.99999999999998E+0000
8.99999999999996E+0000	4.99999999999998E+0000

α_i/β_i:	ilość iteracji:
1.50000000000000E+0000	0
1.00000000000000E+0000	0
1.66666666666667E+0000	2
1.75000000000000E+0000	2
1.83333333333333E+0000	3
1.80000000000000E+0000	5
	----
	razem 12

PRAWE wektory własne:

x1	x2
3.48139673601820E-0001	1.61966743598451E+0000
6.27689284855250E-0001	8.98327414713116E-0001
3.48139673601819E-0001	8.98327414713118E-0001
3.48139673601818E-0001	8.98327414713118E-0001
3.48139673601818E-0001	8.98327414713118E-0001
3.48139673601819E-0001	8.98327414713118E-0001

x3	x4
1.02379357595084E+0000	1.51487518030284E+0000
1.02379357595086E+0000	1.51487518030285E+0000
1.84588056534734E+0000	1.51487518030284E+0000
1.02379357595082E+0000	2.98458897235150E-0001
1.02379357595083E+0000	1.51487518030282E+0000
1.02379357595083E+0000	1.51487518030285E+0000

x5	x6
1.72645235195953E+0000	-4.45487715294171E+0000
1.72645235195954E+0000	-4.45487715294173E+0000
1.72645235195953E+0000	-4.45487715294171E+0000
1.72645235195956E+0000	-4.45487715294164E+0000
1.72645235195957E+0000	-8.77694571587834E-0001
3.40143578688594E-0001	-4.45487715294177E+0000

LEWE wektory własne:

y1	y2
-4.45487715294161E+0000	-3.40143578688678E-0001
-8.77694571587810E-0001	-1.72645235195964E+0000
-4.45487715294169E+0000	-1.72645235195964E+0000
4.45487715294165E+0000	1.72645235195964E+0000
4.45487715294164E+0000	1.72645235195964E+0000
4.45487715294165E+0000	1.72645235195964E+0000

y3	y4
-1.51487518030282E+0000	1.02379357595081E+0000
-1.51487518030282E+0000	1.02379357595081E+0000
-2.98458897235131E-0001	1.02379357595079E+0000
1.51487518030283E+0000	-1.84588056534732E+0000
1.51487518030282E+0000	-1.02379357595080E+0000
1.51487518030283E+0000	-1.02379357595083E+0000

y5	y6
8.98327414713163E-0001	-3.48139673601820E-0001
8.98327414713162E-0001	-3.48139673601820E-0001
8.98327414713159E-0001	-3.48139673601817E-0001
-8.98327414713169E-0001	3.48139673601816E-0001
-8.98327414713172E-0001	6.27689284855244E-0001
-1.61966743598458E+0000	3.48139673601831E-0001

Przykład 2. Elementy macierzy rzeczywistych A,B stopnia 5 były generowane losowo. W wyniku obliczeń otrzymano 3 rzeczywiste wartości własne oraz jedną parę wartości zespolonych (sprzężonych):

α_i:

```
-1.06961427943514E+0000
 1.27941955516918E+0000
 3.07174692065718E-0003
 5.57878284308690E-0001 + 3.84738847312089E-0001 i
 3.23511667763120E-0001 - 2.23108713223050E-0001 i
```

β_i:

```
9.87027005431770E-0001
1.88124362181972E-0001
1.41742057213318E+0000
1.05351458108010E+0000
6.10929424435885E-0001
```

α_i/β_i:	iter
-1.08367276026784E+0000	0
6.80092434775463E+0000	0
2.16713866092284E-0003	3
5.29540164253574E-0001 + 3.65195559911133E-0001 i	6
5.29540164253574E-0001 - 3.65195559911133E-0001 i	-6
	razem 9

Ta sama liczba iteracji podana z przeciwnym znakiem dla dwóch sąsiadujących wartości własnych (6,-6) oznacza, że wartości te wydzieliły się jednocześnie jako para.

Unormowane prawe wektory własne ($\|\| x \|\| = 1$):

x1	x2	x3
-6.20665442217566E-0001	-1.39529658893651E-0001	-1.98772000436069E-0001
5.87323716100432E-0001	6.93522065739112E-0002	4.97694311325392E-0001
7.38553549305699E-0002	-1.90379741957077E-0001	-5.76196217087885E-0001
-2.53688602146169E-0001	-4.54067823509943E-0001	5.11983610964060E-0001
-4.47227840179985E-0001	8.56329207278165E-0001	3.44471719934743E-0001

x4

2.37715381342180E-0001 - 1.65154543356793E-0001 i
6.62267556701453E-0001 + 0.00000000000000E+0000 i
1.17316787161497E-0001 + 5.76886509419147E-0001 i
1.94666270504316E-0001 - 3.06650315506008E-0002 i
2.26778798164573E-0001 - 2.01969948244556E-0001 i

x5

2.37715381342180E-0001 + 1.65154543356793E-0001 i
6.62267556701453E-0001 + 0.00000000000000E+0000 i
1.17316787161497E-0001 - 5.76886509419147E-0001 i
1.94666270504316E-0001 + 3.06650315506008E-0002 i
2.26778798164573E-0001 + 2.01969948244556E-0001 i

Przykład 3. Macierze rzeczywiste A,B stopnia 6:

$$A = \begin{bmatrix} 0 & 0 & 0 & 5 & 5 & 5 \\ 1 & 0 & 0 & 6 & 6 & 6 \\ 0 & 2 & 3 & 7 & 7 & 7 \\ 0 & 0 & 4 & 8 & 8 & 8 \\ 0 & 0 & 0 & 9 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}, \qquad B = \begin{bmatrix} 1 & 0 & 0 & 3 & 3 & 3 \\ 0 & 0 & 0 & 0 & 4 & 4 \\ 0 & 0 & 2 & 5 & 5 & 5 \\ 0 & 0 & 0 & 6 & 6 & 6 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

zaczerpnięto z pracy [10]; macierze te mają już uogólnioną postać Hessenberga, a zatem etap wstępnej redukcji jest pomijany w obliczeniach. Macierz B jest macierzą osobliwą. Zagadnienie posiada trzy nieskończone wartości własne, jedną skończoną rzeczywistą oraz jedną parę wartości zespolonych.

W wyniku obliczeń otrzymano:

α_i:

-2.00000000000000E+0000
-4.00000000000000E+0000
3.39116499156263E+0000
9.47178442001488E+0000
-1.75696878590961E+0000 + 2.14426795901340E-0001 i
-7.85681964272457E-0001 - 9.58874554559565E-0002 i

β_i:

1.75155045602460E-0015
1.75155045602460E-0015
1.75155045602460E-0015
5.66568049440244E+0000
1.95981510720501E+0000
8.76390858726889E-0001

α_i/β_i:

```
-1.14184549644050E+0015
-2.28369099288100E+0015
 1.93609323665124E+0015
 1.67178230918118E+0000
-8.96497215196650E-0001 + 1.09411747625083E-0001 i
-8.96497215196650E-0001 - 1.09411747625083E-0001 i
```

Uzyskane numerycznie nieskończone wartości własne są wynikiem dzielenia elementów diagonalnych α_i przez odpowiadające im zerowe elementy diagonalne β_i, przyjęte jako $eps \cdot \|B\|_1$.

Przykład 4. Macierze A,B stopnia 8 zaczerpnięto z pracy [5]:

$$A = \begin{bmatrix} 6 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 5 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 4 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \qquad B = \mathrm{diag}\{1,1,1,1,\delta,\delta,\delta,\delta\} .$$

Zagadnienie to posiada 2 wartości własne niezależne od wielkości δ:

$$\lambda = 4 \quad \text{oraz} \quad \lambda = 3,$$

natomiast pozostałe wartości własne zależą od δ i niektóre z nich dążą do nieskończoności, gdy $\delta \longrightarrow 0$.

W celu zbadania wpływu błędów zaokrągleń na wyniki obliczeń macierze A,B przekształcono przez podobieństwo za pomocą macierzy Householdera:

$$H = I - ee^T ,$$

gdzie wszystkie elementy wektora e są równe $n^{-1/2}$

$$\tilde{A} = H^T A H, \qquad \tilde{B} = H^T B H .$$

Otrzymano następujące wyniki obliczeń dla zagadnienia : macierzami testowymi $\tilde{A},\tilde{B}$:

1. $\delta = 10^{-5}$:

$\tilde{\alpha}_i$	$\tilde{\beta}_i$
1.00000000000000E+0000	9.99999999998845E-0006
2.00000000000000E+0000	9.99999999998200E-0006
-9.81205666421172E-0001	3.13242055369110E-0003
-9.84313273237136E-0001	3.13737697599903E-0003
1.01915432637902E+0000	3.19241935371757E-0003
1.01593672176189E+0000	3.18737597582404E-0003
3.00000000000000E+0000	1.00000000000000E+0000
4.00000000000000E+0000	1.00000000000000E+0000

$\tilde{\alpha}_i/\tilde{\beta}_i$:

1.00000000000016E+0005
2.00000000000360E+0005
-3.13241995958992E+0002
-3.13737647967440E+0002
3.19241995946497E+0002
3.18737647979930E+0002
3.00000000000000E+0000
4.00000000000000E+0000

2. $\delta = 10^{-15}$:

$\tilde{\alpha}_i$	$\tilde{\beta}_i$
1.00023855932758E+0000	1.00979647688736E-0015
1.99908299500685E+0000	1.02495330931483E-0015
1.03508291720901E+0000	3.46449170794017E-0001
-1.27419753125446E+0000	-3.81389798215226E-0015
-7.84796164876550E-0001	2.47832564153379E-0008
1.00000017368575E+0000	3.15792267675972E-0008
3.00000000000000E+0000	1.00000000000000E+0000
4.00000000000000E+0000	1.00000000000000E+0000

$\tilde{\alpha}_i/\tilde{\beta}_i$:

9.90534807975133E+0014
1.95041371820460E+0015
2.98769056031144E+0000
3.34093239309826E+0014
-3.16663860359712E+0007
3.16663919938605E+0007
3.00000000000000E+0000
4.00000000000000E+0000

Wyniki wskazują, że niezależnie od wielkości δ (determinującej wystąpienie nieskończonych wartości własnych) wartości $\lambda = 4, 3$

obliczane są z dokładnością maszynową, co świadczy o stabilności numerycznej algorytmu QZ.

Przykład 5. Macierze rzeczywiste A,B stopnia 6 zaczerpnięte są z pracy [7]:

$$A = \begin{bmatrix} 50 & -60 & 50 & -27 & 6 & 6 \\ 38 & -28 & 27 & -17 & 5 & 5 \\ 27 & -17 & 27 & -17 & 5 & 5 \\ 27 & -28 & 38 & -17 & 5 & 5 \\ 27 & -28 & 27 & -17 & 16 & 5 \\ 27 & -28 & 27 & -17 & 5 & 16 \end{bmatrix}, \qquad B = \begin{bmatrix} 16 & 5 & 5 & 5 & -6 & 5 \\ 5 & 16 & 5 & 5 & -6 & 5 \\ 5 & 5 & 16 & 5 & -6 & 5 \\ 5 & 5 & 5 & 16 & -6 & 5 \\ 5 & 5 & 5 & 5 & -6 & 16 \\ 6 & 6 & 6 & 6 & -5 & 6 \end{bmatrix}.$$

Zagadnienie to posiada 3 podwójne wartości własne:

$$\lambda_1 = \lambda_2 = \infty \ ,$$

$$\lambda_3 = \lambda_4 = 0.5 + \frac{\sqrt{3}}{2} i \ ,$$

$$\lambda_3 = \lambda_4 = 0.5 - \frac{\sqrt{3}}{2} i \ ,$$

którym odpowiadają kwadratowe dzielniki elementarne, t.j. każdej wartości odpowiada tylko jeden wektor własny.

W wyniku obliczeń numerycznych otrzymano:

α_i:

```
-2.57686707116200E+0001
 1.28218411367655E+0001
 5.69162895864406E+0000 + 9.85819024224948E+0000 i
 5.92602907109173E+0000 - 1.02641831342851E+0001 i
 5.52858776929517E+0000 + 9.57579519411233E+0000 i
 5.71813643353103E+0000 - 9.90410312079857E+0000 i
```

β_i:

```
 3.37322400914553E-0016
-1.56381531843480E-0015
 1.13832573398901E+0001
 1.18520575410062E+0001
 1.10571760994484E+0001
 1.14362734471493E+0001
```

α_i/β_i:	iter
-7.63918157873763E+0016	0
-8.19907631394654E+0015	0
5.00000025361723E-0001 + 8.66025422064707E-0001 i	0
5.00000025361723E-0001 - 8.66025422064707E-0001 i	0
4.99999974638277E-0001 + 8.66025385504172E-0001 i	28
4.99999974638277E-0001 - 8.66025385504172E-0001 i	-28
	----
	razem 28

Uzyskana dokładność skończonych wartości własnych jest rzędu $(eps)^{1/2}$, co w przypadku kwadratowych dzielników elementarnych jest prawidłowym wynikiem.

Unormowane PRAWE wektory własne ($\|\| x \|\| = 1$):

x1	x2
-1.56173761888606E-0001	-1.56173761888606E-0001
-1.56173761888606E-0001	-1.56173761888606E-0001
-1.56173761888606E-0001	-1.56173761888606E-0001
-1.56173761888606E-0001	-1.56173761888606E-0001
-9.37042571331636E-0001	-9.37042571331636E-0001
-1.56173761888606E-0001	-1.56173761888606E-0001

x3

2.88312041820577E-0001 + 3.25676798442111E-0001 i
5.39018134520052E-0001 + 0.00000000000000E+0000 i
3.82326809335795E-0001 - 3.79956235558383E-0001 i
-2.50705932644001E-0002 - 4.34235688852109E-0001 i
1.31620710268002E-0001 - 5.42794584409322E-0002 i
1.31620710268002E-0001 - 5.42794584409314E-0002 i

x4

2.88312041820577E-0001 - 3.25676798442111E-0001 i
5.39018134520052E-0001 + 0.00000000000000E+0000 i
3.82326809335795E-0001 + 3.79956235558383E-0001 i
-2.50705932644001E-0002 + 4.34235688852109E-0001 i
1.31620710268002E-0001 + 5.42794584409322E-0002 i
1.31620710268002E-0001 + 5.42794584409314E-0002 i

x5

-2.50706215997058E-0002 + 4.34235686935944E-0001 i
3.82326799098667E-0001 + 3.79956248047812E-0001 i
5.39018136062687E-0001 + 0.00000000000000E+0000 i
2.88312030084126E-0001 - 3.25676808458409E-0001 i
1.31620704849531E-0001 + 5.42794585028150E-0002 i
1.31620704849530E-0001 + 5.42794585028155E-0002 i

x6

-2.50706215997058E-0002 - 4.34235686935944E-0001 i
3.82326799098667E-0001 - 3.79956248047812E-0001 i
5.39018136062687E-0001 + 0.00000000000000E+0000 i
2.88312030084126E-0001 + 3.25676808458409E-0001 i
1.31620704849531E-0001 - 5.42794585028150E-0002 i
1.31620704849530E-0001 - 5.42794585028155E-0002 i

Unormowane LEWE wektory własne (|| x || = 1):

y1	y2
-1.56173761888607E-0001	1.56173761888607E-0001
-1.56173761888606E-0001	1.56173761888606E-0001
-1.56173761888606E-0001	1.56173761888606E-0001
-1.56173761888606E-0001	1.56173761888606E-0001
-1.56173761888606E-0001	1.56173761888606E-0001
9.37042571331636E-0001	-9.37042571331636E-0001

y3

-3.18799132641198E-0001 - 1.01716691015129E-0001 i
6.20819356588478E-0001 + 0.00000000000000E+0000 i
-6.12429893596673E-0001 + 1.01716676666502E-0001 i
2.97825500555229E-0001 - 1.52575014687410E-0001 i
4.19472303138781E-0003 + 5.08583430120124E-0002 i
4.19472303138785E-0003 + 5.08583430120126E-0002 i

y4

-3.18799132641198E-0001 + 1.01716691015129E-0001 i
6.20819356588478E-0001 + 0.00000000000000E+0000 i
-6.12429893596673E-0001 - 1.01716676666502E-0001 i
2.97825500555229E-0001 + 1.52575014687410E-0001 i
4.19472303138781E-0003 - 5.08583430120124E-0002 i
4.19472303138785E-0003 - 5.08583430120126E-0002 i

y5

2.97825492858333E-0001 + 1.52575020672379E-0001 i
-6.12429893954719E-0001 - 1.01716688940474E-0001 i
6.20819358952686E-0001 + 0.00000000000000E+0000 i
-3.18799128003120E-0001 + 1.01716691992747E-0001 i
4.19472338227340E-0003 - 5.08583412415506E-0002 i
4.19472338227338E-0003 - 5.08583412415508E-0002 i

y6

2.97825492858333E-0001 - 1.52575020672379E-0001 i
-6.12429893954719E-0001 + 1.01716688940474E-0001 i
6.20819358952686E-0001 + 0.00000000000000E+0000 i
-3.18799128003120E-0001 - 1.01716691992747E-0001 i
4.19472338227340E-0003 + 5.08583412415506E-0002 i
4.19472338227338E-0003 + 5.08583412415508E-0002 i

Wektory x1, x2 oraz y1, y2 (odpowiadające nieskończonym wartościom własnym) należą do podprzestrzeni zerowej macierzy B.

W przypadku każdej podwójnej wartości własnej $\lambda_i = \lambda_j$ $(i \neq j)$ unormowane prawe i lewe wektory spełniają zależności: $|x_i^H x_j| = 1$ oraz $|y_i^H y_j| = 1$, a zatem w rzeczywistości odpowiada jej tylko jeden prawy i jeden lewy wektor własny.

Przykład 6. Macierze A,B są zespolonymi niehermitowskimi macierzami Ortegi stopnia 6.

Elementy diagonalne (zespolone) d_{Ak} i d_{Bk} macierzy A i B zadano w postaci ciągów:

$$d_{Ak} = k^2 + k \cdot i, \qquad d_{Bk} = k + 2k \cdot i, \qquad k = 1,\ldots,n,$$

generując obie macierze ze wskaźnikiem uwarunkowania (jednakowym dla wszystkich wartości własnych) $|s_i| = 10$.

Dokładne wartosci własne $\lambda_i = d_{Ai}/d_{Bi}$ wynoszą:

```
6.00000000000000E-0001 + 2.00000000000000E-0001 i
8.00000000000000E-0001 + 6.00000000000000E-0001 i
1.00000000000000E+0000 + 1.00000000000000E+0000 i
1.20000000000000E+0000 + 1.40000000000000E+0000 i
1.40000000000000E+0000 + 1.80000000000000E+0000 i
1.60000000000000E+0000 + 2.20000000000000E+0000 i
```

W wyniku obliczeń numerycznych otrzymano:

α_i:

```
1.34164078649985E+0000 + 4.47213595499964E-0001 i
3.04820786692772E+0001 + 2.00709461660383E+0001 i
3.57770876399970E+0000 + 2.68328157299980E+0000 i
1.56524758424985E+0001 + 2.01246117974981E+0001 i
6.70820393249939E+0000 + 6.70820393249940E+0000 i
1.07331262919989E+0001 + 1.25219806739988E+0001 i
```

β_i:

```
2.23606797749976E+0000 + 0.00000000000000E+0000 i
1.25577577616389E+0001 + 4.72257556847955E+0000 i
4.47213595499961E+0000 + 0.00000000000000E+0000 i
1.11803398874989E+0001 + 0.00000000000000E+0000 i
6.70820393249938E+0000 + 0.00000000000000E+0000 i
8.94427190999913E+0000 + 0.00000000000000E+0000 i
```

α_i/β_i:	iter
5.99999999999998E-0001 + 2.00000000000006E-0001 i	0
1.60000000000000E+0000 + 2.20000000000000E+0000 i	0
8.00000000000002E-0001 + 6.00000000000008E-0001 i	3
1.40000000000000E+0000 + 1.80000000000000E+0000 i	3
1.00000000000000E+0000 + 1.00000000000000E+0000 i	3
1.20000000000000E+0000 + 1.40000000000000E+0000 i	6
razem	15

PRAWE wektory własne:

x1

-3.93287666218100E-0001 + 4.52062785483371E-0001 i
-3.13236688198004E-0001 + 1.73423144600404E-0001 i
-3.13236688198005E-0001 + 1.73423144600407E-0001 i
-3.13236688198004E-0001 + 1.73423144600407E-0001 i
-3.13236688198004E-0001 + 1.73423144600407E-0001 i
-3.13236688198004E-0001 + 1.73423144600407E-0001 i

x2

1.15990245524555E+0000 + 1.70920548005620E-0001 i
1.15990245524555E+0000 + 1.70920548005623E-0001 i
1.15990245524555E+0000 + 1.70920548005624E-0001 i
1.15990245524555E+0000 + 1.70920548005626E-0001 i
1.15990245524555E+0000 + 1.70920548005624E-0001 i
3.97932761604752E-0001 + 7.37167328407620E-0001 i

x3

1.02030266089733E+0000 + 2.78767114318387E-0001 i
1.76409187721001E+0000 - 1.45802953880073E-0001 i
1.02030266089730E+0000 + 2.78767114318409E-0001 i
1.02030266089730E+0000 + 2.78767114318407E-0001 i
1.02030266089730E+0000 + 2.78767114318409E-0001 i
1.02030266089730E+0000 + 2.78767114318409E-0001 i

x4

-8.64886289236241E-0001 + 1.15386663125379E+0000 i
-8.64886289236240E-0001 + 1.15386663125379E+0000 i
-8.64886289236236E-0001 + 1.15386663125379E+0000 i
-8.64886289236242E-0001 + 1.15386663125379E+0000 i
-1.03034350040361E+0000 - 1.98094622599282E-0003 i
-8.64886289236233E-0001 + 1.15386663125377E+0000 i

x5

1.26973462467644E+0000 + 2.83115647859701E-0001 i
1.26973462467645E+0000 + 2.83115647859694E-0001 i
2.15882719637358E+0000 - 2.81778224329643E-0001 i
1.26973462467642E+0000 + 2.83115647859705E-0001 i
1.26973462467642E+0000 + 2.83115647859707E-0001 i
1.26973462467642E+0000 + 2.83115647859707E-0001 i

x6

-3.05833030849209E+0000 + 2.96541655668089E+0000 i
-3.05833030849211E+0000 + 2.96541655668088E+0000 i
-3.05833030849211E+0000 + 2.96541655668086E+0000 i
-3.00513205235308E+0000 - 4.83511318936214E-0001 i
-3.05833030849211E+0000 + 2.96541655668082E+0000 i
-3.05833030849211E+0000 + 2.96541655668082E+0000 i

LEWE wektory własne:

y1

-9.80538851629520E-0001 + 2.88151790000738E+0000 i
 2.41064297015642E+0000 + 3.51224715828920E+0000 i
 2.41064297015638E+0000 + 3.51224715828920E+0000 i
-2.41064297015637E+0000 - 3.51224715828920E+0000 i
-2.41064297015636E+0000 - 3.51224715828920E+0000 i
-2.41064297015636E+0000 - 3.51224715828920E+0000 i

y2

 1.01396506649329E+0000 + 8.15018607822123E-0001 i
 1.01396506649329E+0000 + 8.15018607822130E-0001 i
 1.01396506649329E+0000 + 8.15018607822128E-0001 i
-1.01396506649329E+0000 - 8.15018607822131E-0001 i
-1.01396506649329E+0000 - 8.15018607822125E-0001 i
-2.06115926631367E+0000 - 7.01110768992877E-0001 i

y3

-1.41415152233077E+0000 - 2.82155928378868E-0001 i
-4.42921790518091E-0001 - 9.30285945532982E-0001 i
-1.41415152233082E+0000 - 2.82155928378846E-0001 i
 1.41415152233080E+0000 + 2.82155928378852E-0001 i
 1.41415152233080E+0000 + 2.82155928378855E-0001 i
 1.41415152233080E+0000 + 2.82155928378856E-0001 i

y4

-1.88910503390884E-0001 - 1.04069277193996E+0000 i
-1.88910503390884E-0001 - 1.04069277193996E+0000 i
-1.88910503390881E-0001 - 1.04069277193996E+0000 i
 1.88910503390878E-0001 + 1.04069277193996E+0000 i
 8.92926310688095E-0001 + 1.52838517929197E+0000 i
 1.88910503390886E-0001 + 1.04069277193997E+0000 i

y5

-1.13762521105266E+0000 - 2.83542622179907E-0001 i
-1.13762521105265E+0000 - 2.83542622179904E-0001 i
-3.23928110946270E-0001 - 7.72552090003543E-0001 i
 1.13762521105267E+0000 + 2.83542622179903E-0001 i
 1.13762521105266E+0000 + 2.83542622179903E-0001 i
 1.13762521105266E+0000 + 2.83542622179905E-0001 i

y6

```
-1.18447390977365E-0001 - 3.37880193984500E-0001 i
-1.18447390977365E-0001 - 3.37880193984500E-0001 i
-1.18447390977364E-0001 - 3.37880193984499E-0001 i
 3.79720144201344E-0001 + 4.63517596759784E-0001 i
 1.18447390977366E-0001 + 3.37880193984504E-0001 i
 1.18447390977365E-0001 + 3.37880193984503E-0001 i
```

Przykład 7. Macierze zespolone A,B stopnia 7:

$$A = \mathrm{diag}\left[-1,\ 0,\ -2,\ i,\ 1,\ 1,\ 5+i\right],$$

$$B = \mathrm{diag}\left[1,\ 1,\ i,\ i,\ 0,\ 0,\ 0.5\right]$$

są osobliwe, przy czym macierz B ma podwójne zero na diagonali.

Macierze testowe (pełne) zbudowano za pomocą unitarnego przekształcenia przez podobieństwo:

$$\tilde{A} = S^H A S, \qquad \tilde{B} = S^H B S,$$

gdzie S jest macierzą Householdera:

$$S = I - ee^H,$$

z wektorem e o elementach generowanych losowo (a następnie unormowanym).

Dokładne wartości własne a_{ii}/b_{ii} zagadnienia z macierzami A,B wynoszą:

$$-1,\ 0,\ 2i,\ 1,\ \infty,\ \infty,\ 1+2i.$$

W wyniku obliczeń numerycznych otrzymano:

$\tilde{\alpha}_i$:

```
 8.64480263327631E-0001 + 5.02666762693726E-0001 i
-8.64480263327629E-0001 - 5.02666762693728E-0001 i
 3.63581679001565E-0017 + 2.00000000000000E+0000 i
-5.00000000000000E+0000 - 1.00000000000001E+0000 i
 1.00000000000000E+0000 + 8.01882356596580E-0017 i
-1.00000000000000E+0000 - 1.08687061992354E-0016 i
 1.27114719987921E-0017 + 2.38887207710259E-0017 i
```

$\tilde{\beta}_i$:

```
 2.35424337359860E-0017 + 1.00129458760825E-0016 i
-9.45968174289150E-0017 - 4.95794981804527E-0018 i
 9.99999999999999E-0001 + 0.00000000000000E+0000 i
-5.00000000000000E-0001 - 1.30463699495605E-0015 i
 1.00000000000000E+0000 + 0.00000000000000E+0000 i
 1.00000000000000E+0000 + 0.00000000000000E+0000 i
 1.00000000000000E+0000 + 0.00000000000000E+0000 i
```

$\tilde{\alpha}_i/\tilde{\beta}_i$	iter
6.68078324173644E+0015 + 7.06284020014346E+0015 i	0
8.83580248468067E+0015 + 5.77687751943224E+0015 i	0
3.63581679001566E-0017 + 2.00000000000000E+0000 i	0
1.00000000000000E+0001 + 2.00000000000000E+0000 i	0
1.00000000000000E+0000 + 8.01882356596580E-0017 i	2
-1.00000000000000E+0000 - 1.08687061992354E-0016 i	3
1.27114719987921E-0017 + 2.38887207710259E-0017 i	4
	---
razem	9

Uzyskane wyniki wskazują, że wszystkie skończone wartości własne obliczane są z dokładnością maszynową ($eps \sim 10^{-16}$).

Przykład 8. Macierz zespolona A jest sumą prostą bloków diagonalnych mających postać klatek Jordana o wymiarach 1,2,3, natomiast macierz B jest macierzą diagonalną (n = 6):

$$A = \mathrm{diag}\left\{10, \begin{bmatrix} 6+i & 1 \\ 0 & 6+i \end{bmatrix}, \begin{bmatrix} 4 & 1 & 0 \\ 0 & 4 & 1 \\ 0 & 0 & 4 \end{bmatrix}\right\}, \qquad B = \mathrm{diag}\{2,3,3,1,1,1\} .$$

Zagadnienie to ma następujące wartości własne:

$$\lambda_1 = 5 ,$$

$$\lambda_2 = \lambda_3 = 2 + 1/3\, i ,$$

$$\lambda_3 = \lambda_4 = \lambda_5 = 4 .$$

Podwójna wartość własna $\lambda_2 = \lambda_3$ ma kwadratowy dzielnik elementarny, natomiast wartość potrójna $\lambda_3 = \lambda_4 = \lambda_5$ - dzielnik elementarny stopnia 3. Oznacza to istnienie jedynie trzech niezależnych wektorów własnych.

W celu zbadania wpływu błędów zaokrągleń na wyniki obliczeń

macierze A,B przekształcono przez podobieństwo za pomocą macierzy unitarnej:

$$S = I - ee^H ,$$

gdzie elementy wektora e były generowane losowo (następnie wektor e był normowany).

$$\tilde{A} = S^H A S, \qquad \tilde{B} = S^H B S.$$

Otrzymano następujące wyniki obliczeń dla zagadnienia z macierzami $\tilde{A},\tilde{B}$:

$\tilde{\alpha}_i$:

```
 5.99999999977077E+0000 + 1.00000003830964E+0000 i
-6.00000000022925E+0000 - 9.99999961690354E-0001 i
 3.99999943699277E+0000 + 1.03156356213904E-0005 i
 3.99999134797808E+0000 - 5.64556301974982E-0006 i
 4.00000921502914E+0000 - 4.67007260053934E-0006 i
 1.00000000000000E+0001 - 6.06357725565623E-0016 i
```

$\tilde{\beta}_i$:

```
 3.00000000000000E+0000 + 0.00000000000000E+0000 i
-3.00000000000000E+0000 + 2.48568255822174E-0015 i
 1.00000000000000E+0000 + 0.00000000000000E+0000 i
 1.00000000000000E+0000 + 0.00000000000000E+0000 i
 1.00000000000000E+0000 + 0.00000000000000E+0000 i
 2.00000000000000E+0000 + 0.00000000000000E+0000 i
```

$\tilde{\alpha}_i/\tilde{\beta}_i$:

```
 1.99999999992359E+0000 + 3.33333346103214E-0001 i
 2.00000000007641E+0000 + 3.33333320563453E-0001 i
 3.99999943699277E+0000 + 1.03156356213904E-0005 i
 3.99999134797808E+0000 - 5.64556301974982E-0006 i
 4.00000921502914E+0000 - 4.67007260053934E-0006 i
 5.00000000000000E+0000 - 3.03178862782812E-0016 i
```

Uzyskana dokładność obliczeń pojedynczej, podwójnej i potrójnej wartości własnej jest w przybliżeniu równa *eps*, $(eps)^{1/2}$ oraz $(eps)^{1/3}$, co jest rzeczą normalną w przypadku występowania dzielników elementarnych stopni 2 oraz 3.

Otrzymano następujące unormowane ($\|\ \| \times \|\ \| = 1$) PRAWE wektory własne:

x1

1.80977018776671E-0001 + 6.24182076197963E-0002 i
9.68897443442726E-0001 + 0.00000000000000E+0000 i
2.10056302565168E-0002 + 6.34105008287326E-0002 i
-3.91089536943879E-0002 - 5.40570572207469E-0002 i
-8.27480042780329E-0002 + 7.33533247632140E-0003 i
-7.67653523338711E-0002 - 5.36774410404467E-0002 i

x2

1.80977043596871E-0001 + 6.24182271907573E-0002 i
9.68897438593892E-0001 + 0.00000000000000E+0000 i
2.10056305419845E-0002 + 6.34104352371470E-0002 i
-3.91089568522178E-0002 - 5.40570677745769E-0002 i
-8.27480175771376E-0002 + 7.33532912071710E-0003 i
-7.67653613829078E-0002 - 5.36774535825467E-0002 i

x3

3.36046313248961E-0001 - 2.36051923450979E-0001 i
-3.91088869848075E-0002 + 5.40561739687041E-0002 i
1.36620080658183E-0001 + 4.32256656939788E-0002 i
8.56872687202873E-0001 + 0.00000000000000E+0000 i
-9.13005326275342E-0002 + 1.53049732753865E-0001 i
-1.89817342764775E-0001 + 6.59230729347425E-0002 i

x4

3.36045126998371E-0001 - 2.36062119098914E-0001 i
-3.91081647027574E-0002 + 5.40576770720617E-0002 i
1.36622657311045E-0001 + 4.32231779926781E-0002 i
8.56870982979373E-0001 + 0.00000000000000E+0000 i
-9.13063455238232E-0002 + 1.53037600218666E-0001 i
-1.89818154012337E-0001 + 6.59280294375865E-0002 i

x5

3.36054040922267E-0001 - 2.36056826919338E-0001 i
-3.91098141243268E-0002 + 5.40573364411380E-0002 i
1.36623076812654E-0001 + 4.32267708083384E-0002 i
8.56869500992118E-0001 + 0.00000000000000E+0000 i
-9.12929470584858E-0002 + 1.53038058461383E-0001 i
-1.89821637121593E-0001 + 6.59243419239869E-0002 i

x6

-1.71467469638603E-0001 + 4.89528675521245E-0002 i
1.56888532231138E-0001 - 1.09702817354840E-0001 i
-3.29614342053515E-0001 - 2.45767508299398E-0001 i
3.87941026040491E-0001 + 1.34733949613798E-0001 i
3.91527511669261E-0001 - 3.28864233251547E-0001 i
5.76549710033687E-0001 + 0.00000000000000E+0000 i

oraz LEWE wektory własne (|| y || = 1):

y1

-2.49481238779122E-0001 - 3.26812466724926E-0001 i
2.10056302377918E-0002 + 6.34105051309612E-0002 i
8.56535152241089E-0001 + 0.00000000000000E+0000 i
1.36621936249366E-0001 + 4.32252031933015E-0002 i
4.09303102665337E-0002 + 1.73656876681451E-0001 i
1.61279871331129E-0001 + 1.20253722327435E-0001 i

y2

-2.49481242853411E-0001 - 3.26812481097118E-0001 i
2.10056305607078E-0002 + 6.34104309349175E-0002 i
8.56535147392255E-0001 + 0.00000000000000E+0000 i
1.36621940290561E-0001 + 4.32252064759364E-0002 i
4.09303093343710E-0002 + 1.73656883096479E-0001 i
1.61279875178622E-0001 + 1.20253728542272E-0001 i

y3

5.54400092742636E-0001 + 1.58283804912086E-0001 i
-7.67652313931768E-0002 - 5.36785260355199E-0002 i
1.61281987772827E-0001 - 1.20252741107937E-0001 i
-1.89820761840219E-0001 + 6.59235736215191E-0002 i
-1.91573882275426E-0001 - 1.60905370692774E-0001 i
7.17893416995098E-0001 + 0.00000000000000E+0000 i

y4

5.54393372453698E-0001 + 1.58276687210867E-0001 i
-7.67645762700170E-0002 - 5.36770768442050E-0002 i
1.61278590779847E-0001 - 1.20252384208395E-0001 i
-1.89817419194211E-0001 + 6.59242573506124E-0002 i
-1.91580570092212E-0001 - 1.60917321829803E-0001 i
7.17897534977147E-0001 + 0.00000000000000E+0000 i

y5

5.54402508533408E-0001 + 1.58271336834583E-0001 i
-7.67662628951401E-0002 - 5.36767390429939E-0002 i
1.61279041176584E-0001 - 1.20256050961858E-0001 i
-1.89818952861569E-0001 + 6.59276133232438E-0002 i
-1.91567269974537E-0001 - 1.60915845429076E-0001 i
7.17893955135360E-0001 + 0.00000000000000E+0000 i

y6

-1.71467469638605E-0001 - 4.89528675521236E-0002 i
1.56888532231138E-0001 + 1.09702817354839E-0001 i
-3.29614342053516E-0001 + 2.45767508299399E-0001 i
3.87941026040490E-0001 - 1.34733949613800E-0001 i
3.91527511669261E-0001 + 3.28864233251547E-0001 i
5.76549710033686E-0001 + 0.00000000000000E+0000 i

Łatwo zauważyć, że w granicach uzyskanej dokładności obliczeń są to tylko 3 niezależne prawe i lewe wektory.

Przykład 9. Macierze rzeczywiste A i B stopnia 6 mają postać:

$$A = \begin{bmatrix} -9 & 21 & -15 & 4 & 2 & 0 \\ -10 & 21 & -14 & 4 & 2 & 0 \\ -8 & 16 & -11 & 4 & 2 & 0 \\ -6 & 12 & -9 & 3 & 3 & 0 \\ -4 & 8 & -6 & 0 & 5 & 0 \\ -2 & 4 & -3 & 0 & 1 & 3 \end{bmatrix}, \qquad B = I.$$

Macierz A zaczerpnięto z pracy [15]. Posiada ona następujące wartości własne:

$\lambda_1 = \lambda_2 = 3$ (liniowe dzielniki elementarne)

$\lambda_3 = 2 + i$

$\lambda_4 = 2 - i$

$\lambda_5 = \lambda_6 = 1$ (kwadratowy dzielnik elementarny).

W wyniku obliczeń otrzymano:

α_i:

```
2.00000000000000E+0000 + 1.00000000000001E+0000 i
2.00000000000000E+0000 - 1.00000000000001E+0000 i
3.00000000000000E+0000
1.00000000000000E+0000 + 3.95918571019636E-0008 i
1.00000000000000E+0000 - 3.95918571019636E-0008 i
3.00000000000000E+0000
```

β_i:

```
1.00000000000000E+0000
1.00000000000000E+0000
1.00000000000000E+0000
1.00000000000000E+0000
1.00000000000000E+0000
1.00000000000000E+0000
```

α_i/β_i:

```
2.00000000000000E+0000 + 1.00000000000001E+0000 i
2.00000000000000E+0000 - 1.00000000000001E+0000 i
3.00000000000000E+0000
1.00000000000000E+0000 + 3.95918571019636E-0008 i
1.00000000000000E+0000 - 3.95918571019636E-0008 i
3.00000000000000E+0000
```

Przykład ten wskazuje, że jeśli B jest macierzą jednostkową, algorytm QZ daje te same wyniki, co algorytm QR, wyprowadzając wartości własne macierzy A.

LITERATURA

1. J.H.WILKINSON, The Algebraic Eigenvalue Problem, Oxford University Press, Oxford, 1965.

2. J.H.WILKINSON, C.REINSCH, Handbook for Automatic Computation, Vol. II, Linear Algebra, Springer Verlag, Berlin Heidelberg New York, 1971.

3. E.TYRKIEL, Wyznaczanie wskaźnika uwarunkowania macierzy - porównanie metod przybliżonych, Raport IPPT PAN 6/1992.

4. G.PETERS, J.H.WILKINSON, Ax = λBx and the Generalized Eigenproblem, SIAM J. Numer. Anal., vol.7, No 4, 1970.

5. G.FIX, R.HEIBERGER, An Algorithm for the ill-conditioned Generalized Eigenvalue Problem, Ibid., vol.9, 1972.

6. R.S.MARTIN, J.H.WILKINSON, Reduction of the Symmetric Eigenproblem Ax = λBx and Related Problems to Standard Form, Numer. Math., vol.11, 1968.

7. C.B.MOLER, G.W.STEWART, An Algorithm for Generalized Matrix Eigenvalue Problems, SIAM J.Numer.Anal.,vol.10, No 2, 1973.

8. H.D.IKRAMOW, Czislennoje reszenije matricznych urawnienij. Ortogonalnyje metody, Moskwa, "Nauka", 1984.

9. G.W.STEWART, On the Sensitivity of the Eigenvalue Problem Ax = λBx, SIAM J. Numer. Anal., vol.9, No 4, 1972.

10. R.C.WARD, The Combination Shift QZ Algorithm, SIAM J. Numer. Anal., vol.12, No 6, 1975.

11. M.NOWAK, Niektóre algorytmy algebry liniowej w Języku Pascal, Raport IPPT PAN 22/1986.

12. M.NOWAK, Usprawnienie programów numerycznych w języku Pascal dla mikrokomputerów IBM XT/AT, Raport IPPT PAN 10/1989.

13. J.G.F.FRANCIS, The QR Transformation - a Unitary Analogue to the LR Transformation, Comput. J., vol.4, 1961-62.

14. KING-WAH E.CHU, Exclusion Theorems and the Perturbation Analysis of the Generalized Eigenvalue Problem, SIAM J. Numer. Anal., vol.24, No 5, 1987.

15. R.T.GREGORY, D.L.KARNEY, A Collection of Matrices for Testing Computational Algorithms, Wiley Interscience, New York, London, Sydney, Toronto, 1969.

ANEKS: PAKIETY PROCEDUR NUMERYCZNYCH W JĘZYKU TURBO PASCAL 4.0

WARTOŚCI WŁASNE ZAGADNIENIA Ax = λBx Z MACIERZAMI ZESPOLONYMI

Plik QZVALC.PAS

Opis części zewnętrznej

```
INTERFACE
uses CompUnit,NewType,VecComp;
function qz(n,itm:Integer;A,B:CompPtr2;CNT:IntPtr1):Boolean;
end;
```

Opis parametrów formalnych:

```
qz(n,itm,A,B,CNT);
```

Parametry wejściowe:

n - stopień macierzy;

itm - maksymalna liczba iteracji dla wydzielenia się jednej wartości własnej (w przypadku zerowej wartości tego parametru ustawiana jest wartość domniemana itm = 30);

A - tablica dwuwymiarowa zawierająca w wierszach 1,...,n zadaną macierz A;

B - tablica dwuwymiarowa zawierająca w wierszach 1,...,n zadaną macierz B;

Parametry wyjściowe:

A - tablica dwuwymiarowa zawierająca w wierszach 1,...,n macierz trójkątną górną QAZ;

B - tablica dwuwymiarowa zawierająca w wierszach 1,...,n macierz trójkątną górną QBZ;

UWAGA: w celu uzyskania wartości własnych należy wykonać dzielenie (zespolone) elementów diagonalnych:

$$\lambda_i = A^{\wedge}[i]^{\wedge}[i]/B^{\wedge}[i]^{\wedge}[i];$$

CNT - tablica jednowymiarowa zawierająca liczby iteracji wykonanych do chwili wydzielenia się poszczególnych wartości własnych; jeżeli dwie wartości własne zostały wydzielone jednocześnie jako para, to liczba iteracji podana jest z dodatnim znakiem dla pierwszej wartości własnej, z ujemnym - dla drugiej.

qz - przyjmuje wartość TRUE, jeżeli podczas obliczania każdej wartości własnej nie została przekroczona maksymalna liczba iteracji (itm); w przeciwnym razie proces iteracyjny zostaje przerwany, co sygnalizowane jest wartością qz = FALSE.

WARTOŚCI I PRAWE WEKTORY WŁASNE ZAGADNIENIA Ax = λBx Z MACIERZAMI ZESPOLONYMI

Plik QZVECC.PAS

Opis części zewnętrznej

```
INTERFACE

uses CompUnit,NewType,VecComp;
function qz(n,itm:Integer;A,B:CompPtr2;WW:CompPtr1):Boolean;

end;
```

Opis parametrów formalnych:

```
qz(n,itm,A,B,WW);
```

Parametry wejściowe:

n - stopień macierzy;

itm - maksymalna liczba iteracji dla wydzielenia się jednej wartości własnej (w przypadku zerowej wartości tego parametru ustawiana jest wartość domniemana itm = 30);

A - tablica dwuwymiarowa zawierająca w wierszach 1,...,n zadaną macierz A;

B - tablica dwuwymiarowa zawierająca w wierszach 1,...,n zadaną macierz B;

Parametry wyjściowe:

B - tablica dwuwymiarowa zawierająca:

w pierwszym wierszu - elementy diagonalne macierzy trójkątnej górnej QAZ;

w drugim wierszu - elementy diagonalne macierzy trójkątnej górnej QBZ;

A - tablica dwuwymiarowa zawierająca n prawych wektorów własnych (nie unormowanych), zapisanych kolumnami; dostęp do j-tego elementu i-tego wektora - A^[j]^[i];

WW - tablica jednowymiarowa zawierająca wartości własne;

qz - przyjmuje wartość TRUE, jeżeli podczas obliczania każdej wartości własnej nie została przekroczona maksymalna liczba iteracji (itm); w przeciwnym razie proces iteracyjny zostaje przerwany, co sygnalizowane jest wartością qz = FALSE.

PEŁNE UOGÓLNIONE ZAGADNIENIE WŁASNE Ax = λBx Z MACIERZAMI ZESPOLONYMI

Plik QZEIGC.PAS

Opis części zewnętrznej

```
INTERFACE

uses CompUnit,NewType,VecComp;
function qz(n,itm:Integer;A,B:CompPtr2;diagA,diagB,WW:
           CompPtr1):Boolean;
end;
```

Opis parametrów formalnych:

```
qz(n,itm,A,B,diagA,diagB,WW);
```

Parametry wejściowe:

n - stopień macierzy;

itm - maksymalna liczba iteracji dla wydzielenia się jednej wartości własnej (w przypadku zerowej wartości tego parametru ustawiana jest wartość domniemana itm = 30);

A - tablica dwuwymiarowa zawierająca w wierszach 1,...,n zadaną macierz A;

B - tablica dwuwymiarowa zawierająca w wierszach 1,...,n zadaną macierz B;

Parametry wyjściowe:

diagA - tablica jednowymiarowa zawierająca elementy diagonalne macierzy trójkątnej górnej QAZ;

diagB - tablica jednowymiarowa zawierająca elementy diagonalne macierzy trójkątnej górnej QBZ;

WW - tablica jednowymiarowa zawierająca wartości własne;

A - tablica dwuwymiarowa zawierająca n prawych wektorów własnych (nie unormowanych), zapisanych kolumnami; dostęp do j-tego elementu i-tego wektora - A^[j]^[i];

B - tablica dwuwymiarowa zawierająca n lewych wektorów własnych (nie unormowanych), zapisanych wierszami; dostęp do j-tego elementu i-tego wektora - B^[i]^[j];

qz - przyjmuje wartość TRUE, jeżeli podczas obliczania każdej wartości własnej nie została przekroczona maksymalna liczba iteracji (itm); w przeciwnym razie proces iteracyjny zostaje przerwany, co sygnalizowane jest wartością qz = FALSE.

```
unit QZEIGC;

INTERFACE

uses CompUnit,NewType,VecComp;
function qz(n,itm:Integer;A,B:CompPtr2;diagA,diagB,WW:
            CompPtr1):Boolean;

IMPLEMENTATION

const
 tol:Double=1E-280;
 eps:Double=1.11E-16;

function SAbs(X:Complex):Double;
begin
 with X do SAbs:=Abs(Re)+Abs(Im)
end;

procedure unittr(n:Integer;A,B,QU:CompPtr2);
var
 i,j:Integer;
 H,Z,Y:Double;
 F,G:Complex;
begin
 for i:=1 to n-1 do
 begin
  H:=NormE(B^[i]^[i],n,n-i+1);
  if H<tol then G:=Zero else
  begin
   F:=B^[i]^[i];   Y:=Sqrt(H);
   if (F.Re=0.0) and (F.Im=0.0) then
   begin
    RMult(Y,One,B^[i]^[i]);  RMult(-Y,One,G)
   end
    else
   begin
    Z:=CAbs(F);  H:=H+Y*Z;
    DivR(F,-Z,G); RMult(Y,G,G);  Sub(F,G,B^[i]^[i])
   end;
   for j:=i+1 to n do Househ(B^[i]^[i],n,H,B^[i]^[j],n,n-i+1);
   for j:=1 to n do Househ(B^[i]^[i],n,H,A^[i]^[j],n,n-i+1);
   for j:=1 to n do Househ(B^[i]^[i],n,H,QU^[i]^[j],n,n-i+1);
  end;
  B^[i]^[i]:=G;
 end;
 for i:=1 to n-1 do FillChar(B^[i+1]^[1],i shl 4,#0)
end;
```

```
procedure unithes(n:Integer;A,B,QU,ZET:CompPtr2);
var
 i,j:Integer;
 S,C,V:Complex;
 Z:Double;
begin
 for j:=1 to n-2 do
 for i:=n downto j+2 do
 begin
  C:=A^[i-1]^[j];
  S:=A^[i]^[j];
  Z:=CSqr(C)+CSqr(S);
  Z:=Sqrt(Z);
  if Z>tol then
  begin
   DivR(C,Z,C);  DivR(S,Z,S);
   RMult(Z,One,A^[i-1]^[j]);
   A^[i]^[j]:=Zero;
   Rotate(S,C,A^[i-1]^[j+1],A^[i]^[j+1],1,n-j,True);
   Rotate(S,C,B^[i-1]^[i],B^[i]^[i],1,n-i+1,True);
   Rotate(S,C,QU^[i-1]^[1],QU^[i]^[1],1,n,True);
   V:=B^[i-1]^[i-1];
   with C do Im:=-Im;
   CMult(C,V,B^[i-1]^[i-1]);
   Neg(S);
   CMult(S,V,B^[i]^[i-1]);
   C:=B^[i]^[i];
   S:=B^[i]^[i-1];
   Z:=CSqr(C)+CSqr(S);
   Z:=Sqrt(Z);
   if Z>tol then
   begin
    DivR(C,Z,C);  DivR(S,-Z,S);
    RMult(Z,One,B^[i]^[i]);
    B^[i]^[i-1]:=Zero;
    Rotate(S,C,B^[1]^[i-1],B^[1]^[i],n,i-1,False);
    Rotate(S,C,A^[1]^[i-1],A^[1]^[i],n,n,False);
    Rotate(S,C,ZET^[1]^[i-1],ZET^[1]^[i],n,n,False)
   end
  end
 end
end;

function iter(n:Integer;var itm:Integer;NormA,NormB:Double;A,B,
              QU,ZET:CompPtr2;diagA,diagB,WW:CompPtr1):Boolean;
var
 en,na,l,it,i,j:Integer;
 R,X,Y,S,C:Complex;
 NmA,NmB,F,G,H:Double;
 WA,NL,NM:Boolean;

procedure Housn(Q,X,Y:Complex;var A,B:Complex);
var
 P,S:Complex;
```

```
begin
 CMult(Q,B,P);  Add(A,P,P);
 CMult(P,Y,S);  Sub(B,S,B);
 CMult(P,X,S);  Sub(A,S,A)
end;

procedure contsub(en,na,l:Integer;var m:Integer;var T:Complex;
                  var P,Q:Complex);
var
 i,j,k:Integer;
 X,Y,Z:Complex;
 F,G,H:Double;
 NS:Boolean;
begin
 i:=na;   NS:=False;
 if na>=3 then
 repeat
  i:=i-1;
  CMult(A^[i]^[i-1],A^[i+1]^[i],X);
  CMult(T,B^[i]^[i],Y);  Sub(A^[i]^[i],Y,Y);
  if (Y.Re<>0.0) and (Y.Im<>0.0) then
  NS:=SAbs(X)<=eps*SAbs(Y)*NormA
 until (NS) or (i=l+1);
 if not NS then i:=l;
 NS:=True;
 k:=0;
 repeat
  k:=k+1;
  NS:=SAbs(B^[i]^[i])<=eps*NormB;
  if not NS then m:=i else
  begin
   P:=A^[i]^[i];
   Q:=A^[i+1]^[i];
   F:=SAbs(P)+SAbs(Q);
   DivR(P,F,P);  DivR(Q,F,Q);
   H:=Sqrt(CSqr(P)+CSqr(Q));
   if(P.Re=0.0) and (P.Im=0.0) then RMult(H,One,Z) else
   begin
    G:=CAbs(P);   DivR(P,G,Z);   RMult(H,Z,Z)
   end;
   RMult(-F,Z,A^[i]^[i]);
   A^[i+1]^[i]:=Zero;
   if (i<>1) and (k=1) then Neg(A^[i]^[i-1]);
   Add(P,Z,P);  DivC(P,Z,X);  DivC(Q,Z,Y);
   DivC(Q,P,Q);
   with Q do Im:=-Im;
   for j:=i+1 to n do Housn(Q,X,Y,A^[i]^[j],A^[i+1]^[j]);
   for j:=i+1 to n do Housn(Q,X,Y,B^[i]^[j],B^[i+1]^[j]);
   for j:=1 to n do Housn(Q,X,Y,QU^[i]^[j],QU^[i+1]^[j]);
   i:=i+1
  end
 until (not NS) or (i=en);
 if i=en then begin m:=en; exit end;
 DivC(A^[m]^[m],B^[m]^[m],P);   Sub(P,T,P);
```

```
 DivC(A^[m+1]^[m],B^[m]^[m],Q);
 F:=SAbs(P)+SAbs(Q);
 DivR(P,F,P);   DivR(Q,F,Q)
end;

procedure step(en,na,l,m,k:Integer;var P,Q:Complex);
var
 j:Integer;
 X,Y,Z,S,C:Complex;
begin
 if k<>m then
 begin
  P:=A^[k]^[k-1];
  Q:=A^[k+1]^[k-1];
  F:=SAbs(P)+SAbs(Q);
  if F<tol then exit;
  DivR(P,F,P);  DivR(Q,F,Q);
 end;
 H:=Sqrt(CSqr(P)+CSqr(Q));
 if(P.Re=0.0) and (P.Im=0.0) then RMult(H,One,Z) else
 begin
  G:=CAbs(P);   DivR(P,G,Z);   RMult(H,Z,Z)
 end;
 if k<>m then
 begin
  RMult(-F,Z,A^[k]^[k-1]);
  A^[k+1]^[k-1]:=Zero;
 end
  else
 if l<>m then
 begin
  Neg(A^[k]^[k-1]);
  A^[k+1]^[k-1]:=Zero;
 end;
 Add(P,Z,P);  DivC(P,Z,X);  DivC(Q,Z,Y);
 DivC(Q,P,Q);
 with Q do Im:=-Im;
 for j:=k to n do Housn(Q,X,Y,A^[k]^[j],A^[k+1]^[j]);
 for j:=k to n do Housn(Q,X,Y,B^[k]^[j],B^[k+1]^[j]);
 for j:=1 to n do Housn(Q,X,Y,QU^[k]^[j],QU^[k+1]^[j]);
 C:=B^[k+1]^[k+1];
 S:=B^[k+1]^[k];
 F:=CSqr(C)+CSqr(S);   F:=Sqrt(F);
 if F>tol then
 begin
  DivR(C,F,C);    DivR(S,-F,S);
  RMult(F,One,B^[k+1]^[k+1]);
  B^[k+1]^[k]:=Zero;
  Rotate(S,C,B^[1]^[k],B^[1]^[k+1],n,k,False);
  if k<>na then Rotate(S,C,A^[1]^[k],A^[1]^[k+1],n,k+2,False)
  else Rotate(S,C,A^[1]^[k],A^[1]^[k+1],n,en,False);
  Rotate(S,C,ZET^[1]^[k],ZET^[1]^[k+1],n,n,False)
 end
end;
```

```
procedure nextw(en,na,itm:Integer;NormA,NormB:Double;
                var l,it:Integer;var R:Complex;
                var WA,NL,NM:Boolean);
var
 m,k:Integer;
 T,P,Q,V,Z,W,Dm,Dn,Amn,Anm,Det:Complex;
begin
 repeat
  l:=en+1;   T:=Zero;
  NL:=False;
  if en>=2 then
  repeat
   l:=l-1;
   NL:=SAbs(A^[l]^[l-1])<=eps*NormA
  until (l=2) OR (NL);
  if NL then A^[l]^[l-1]:=Zero else l:=1;
  if l=en then exit;
  if (((it=10) or (it=20)) and (it<itm)) and (l<>na) then
  begin
   X:=B^[en-2]^[en-2];
   if SAbs(X)<=eps*NormB then RMult(eps*NormB,One,X);
   DivC(A^[na]^[en-2],X,V);
   X:=B^[na]^[na];
   if SAbs(X)<=eps*NormB then RMult(eps*NormB,One,X);
   DivC(A^[en]^[na],X,Z);
   T.Re:=Abs(Z.Re)+Abs(V.Re);
   T.Im:=Abs(Z.Im)+Abs(V.Im);
  end
   else
  begin
   if it=itm then
   begin
    WA:=True;  exit
   end;
   NM:=SAbs(B^[en]^[en])<=eps*NormB;
   NL:=SAbs(B^[na]^[na])<=eps*NormB;
   if (not NL) and (not NM) then
   begin
    DivC(A^[en]^[en],B^[en]^[en],Dn);
    DivC(A^[na]^[na],B^[na]^[na],Dm);
    DivC(A^[na]^[en],B^[en]^[en],V);
    DivC(A^[en]^[na],B^[na]^[na],Z);
    DivC(B^[na]^[en],B^[en]^[en],X);
    CMult(V,Z,Amn);   CMult(Z,X,Anm);
    CMult(Dm,Anm,Y);  Sub(Amn,Y,W);
    Sub(Dn,Dm,Y);  Sub(Y,Anm,Y);   RMult(0.5,Y,P);
    CMult(P,P,Y);  Add(Y,W,Q);
    CSqrt(Q,Q);
    Sub(P,Q,Y);   Add(P,Q,C);
    if SAbs(C)>SAbs(Y) then Y:=C;
    Add(Dm,Y,R);
    if l=na then exit;
    if (Y.Re=0.0) and (Y.Im=0.0) then C:=Dm else
    begin
```

```
     DivC(W,Y,C);  Sub(Dm,C,C)
    end;
    Sub(R,Dn,Y);  Sub(C,Dn,W);
    if SAbs(Y)<SAbs(W) then T:=R else T:=C
   end
    else
   begin
    if l=na then exit;
    CMult(A^[na]^[na],A^[en]^[en],V);
    CMult(A^[en]^[na],A^[na]^[en],Z);
    Sub(V,Z,Det);
    Neg(Det);
    CMult(A^[en]^[na],B^[na]^[en],V);
    if (NL) and (NM) then
    begin
     if (V.Re<>0.0) and (V.Im<>0.0) then
     DivC(Det,V,T);
    end
     else
    if NL then
    begin
     CMult(A^[na]^[na],B^[en]^[en],Z);
     Sub(V,Z,Z);
     if (Z.Re<>0.0) and (Z.Im<>0.0) then
     DivC(Det,Z,T);
    end
     else
    begin
     CMult(A^[en]^[en],B^[na]^[na],Z);
     Sub(V,Z,Z);
     if (Z.Re<>0.0) and (Z.Im<>0.0) then
     DivC(Det,Z,T);
    end;
    CMult(B^[1]^[1],T,X);  Sub(A^[1]^[1],X,X);
    if SAbs(X)>SAbs(A^[1+1]^[1]) then T:=Zero
   end
  end;
  it:=it+1;
  contsub(en,na,l,m,T,P,Q);
  for k:=m to na do
  step(en,na,l,m,k,P,Q);
 until WA;
end;

begin                    {body of iter}
 en:=n;   WA:=False;
 if itm=0 then itm:=30;
 repeat
  it:=0;   na:=en-1;
  nextw(en,na,itmax,NormA,NormB,l,it,R,WA,NL,NM);
  if WA then
  begin
   iter:=False; exit
  end;
```

```
if l=en then
begin
 diagA^[en]:=A^[en]^[en];
 diagB^[en]:=B^[en]^[en];
 X:=diagB^[en];
 if (X.Re=0.0) and (X.Im=0.0) then
 RMult(eps*NormB,One,diagB^[en]);
 DivC(diagA^[en],diagB^[en],WW^[en]);
 en:=na
end
 else
begin
 if (not NL) and (not NM) then
 begin
  NmA:=SAbs(A^[na]^[na])+SAbs(A^[na]^[en]);
  F:=SAbs(A^[en]^[na])+SAbs(A^[en]^[en]);
  if F>NmA then NmA:=F;
  NmB:=SAbs(B^[na]^[na])+SAbs(B^[na]^[en]);
  F:=SAbs(B^[en]^[en]);
  if F>NmB then NmB:=F;
  NmB:=NmB*CAbs(R);
  CMult(R,B^[na]^[na],Y);  Sub(A^[na]^[na],Y,Y);
  CMult(R,B^[na]^[en],X);  Sub(A^[na]^[en],X,X);
  G:=SAbs(Y)+SAbs(X);
  CMult(R,B^[en]^[en],C);  Sub(A^[en]^[en],C,C);
  F:=SAbs(A^[en]^[na])+SAbs(C);
  if G>F then
  begin
   C:=X;  S:=Y
  end
  else S:=A^[en]^[na];
  G:=CSqr(C)+CSqr(S);   G:=Sqrt(G);
  if G>tol then
  begin
   DivR(C,G,C);  DivR(S,-G,S);
   Rotate(S,C,A^[1]^[na],A^[1]^[en],n,en,False);
   Rotate(S,C,B^[1]^[na],B^[1]^[en],n,en,False);
   Rotate(S,C,ZET^[1]^[na],ZET^[1]^[en],n,n,False);
  end;
  if NmB<NmA then
  begin
   C:=B^[na]^[na];   S:=B^[en]^[na];
   G:=CSqr(C)+CSqr(S);    G:=Sqrt(G);
   if G>tol then
   begin
    DivR(C,G,C);  DivR(S,G,S);
    RMult(G,One,B^[na]^[na]);
    Rotate(S,C,A^[na]^[en],A^[en]^[en],1,n-na,True);
    Rotate(S,C,B^[na]^[en],B^[en]^[en],1,n-na,True);
    Rotate(S,C,QU^[na]^[1],QU^[en]^[1],1,n,True);
    with C do Im:=-Im;
    with S do Im:=-Im;
    X:=A^[na]^[na];  Y:=A^[en]^[na];
    CMult(C,X,X);  CMult(S,Y,Y);
```

```
   Add(X,Y,A^[na]^[na])
  end
 end
  else
 begin
  C:=A^[na]^[na];    S:=A^[en]^[na];
  G:=CSqr(C)+CSqr(S);     G:=Sqrt(G);
  if G>tol then
  begin
   DivR(C,G,C);  DivR(S,G,S);
   RMult(G,One,A^[na]^[na]);
   Rotate(S,C,A^[na]^[en],A^[en]^[en],1,n-na,True);
   Rotate(S,C,B^[na]^[en],B^[en]^[en],1,n-na,True);
   Rotate(S,C,QU^[na]^[1],QU^[en]^[1],1,n,True);
   with C do Im:=-Im;
   with S do Im:=-Im;
   X:=B^[na]^[na];  Y:=B^[en]^[na];
   CMult(C,X,X);  CMult(S,Y,Y);
   Add(X,Y,B^[na]^[na])
  end
 end;
 A^[en]^[na]:=Zero;
 B^[en]^[na]:=Zero
end
 else
if NL then
begin
 C:=A^[na]^[na];
 S:=A^[en]^[na];
 F:=CSqr(C)+CSqr(S);    F:=Sqrt(F);
 if F>tol then
 begin
  DivR(C,F,C);     DivR(S,F,S);
  RMult(F,One,A^[na]^[na]);
  A^[en]^[na]:=Zero;
  Rotate(S,C,A^[na]^[en],A^[en]^[en],1,n-na,True);
  Rotate(S,C,B^[na]^[en],B^[en]^[en],1,n-na,True);
  Rotate(S,C,QU^[na]^[1],QU^[en]^[1],1,n,True)
 end
end
 else
begin
 C:=A^[en]^[en];
 S:=A^[en]^[na];
 F:=CSqr(C)+CSqr(S);    F:=Sqrt(F);
 if F>tol then
 begin
  DivR(C,F,C);     DivR(S,-F,S);
  RMult(F,One,A^[en]^[en]);
  A^[en]^[na]:=Zero;
  Rotate(S,C,A^[1]^[na],A^[1]^[en],n,na,False);
  Rotate(S,C,B^[1]^[na],B^[1]^[en],n,na,False);
  Rotate(S,C,ZET^[1]^[na],ZET^[1]^[en],n,n,False);
 end;
```

```
    end;
    diagA^[na]:=A^[na]^[na];  diagA^[en]:=A^[en]^[en];
    diagB^[na]:=B^[na]^[na];  diagB^[en]:=B^[en]^[en];
    X:=diagB^[na];
    if (X.Re=0.0) and (X.Im=0.0) then
    RMult(eps*NormB,One,diagB^[na]);
    X:=diagB^[en];
    if (X.Re=0.0) and (X.Im=0.0) then
    RMult(eps*NormB,One,diagB^[en]);
    DivC(diagA^[na],diagB^[na],WW^[na]);
    DivC(diagA^[en],diagB^[en],WW^[en]);
    en:=en-2
   end
  until en=0;
  iter:=True
end;

procedure fin(n:Integer;NormA,NormB:Double;A,B,QU,ZET:CompPtr2;
              WW:CompPtr1);
var
 i,j,en,na:Integer;
 S:Double;
 C:CompPtr1;
 W,X,R,Y:Complex;
begin
 GetMem(C,n shl 4);
 for en:=n downto 1 do
 begin
  X:=WW^[en];  na:=en-1;
  if(SAbs(B^[en]^[en])<=eps*NormB) then
  begin
   for i:=na downto 1 do
   begin
    W:=B^[i]^[i];
    if (W.Re=0.0) and (W.Im=0.0) then RMult(eps*NormB,One,W);
    B^[en]^[i]:=B^[i]^[en];  Neg(B^[en]^[i]);
    VecProd(B^[i]^[i+1],1,B^[en]^[i+1],1,B^[en]^[i],0,na-i,W);
   end
  end
   else
  begin
   for i:=na downto 1 do
   begin
    for j:=i to en do
    begin
     CMult(X,B^[i]^[j],R);  Sub(A^[i]^[j],R,C^[j])
    end;
    W:=C^[i];
    if (W.Re=0.0) and (W.Im=0.0) then
    begin
     S:=CAbs(X)*NormB;
     if S>NormA then RMult(eps*S,One,W) else
     RMult(eps*NormA,One,W)
    end;
```

```
   Neg(C^[en]);
   B^[en]^[i]:=C^[en];
   VecProd(C^[i+1],1,B^[en]^[i+1],1,B^[en]^[i],0,na-i,W);
  end
 end;
end;
for en:=1 to n do
begin
 X:=WW^[en];   na:=en+1;
 if(SAbs(B^[en]^[en])<=eps*NormB) then
 begin
  for i:=na to n do
  begin
   W:=B^[i]^[i];
   if (W.Re=0.0) and (W.Im=0.0) then RMult(eps*NormB,One,W);
   Neg(B^[en]^[i]);
   VecProd(B^[na]^[i],n,B^[en]^[na],1,B^[en]^[i],0,i-na,W);
  end
 end
  else
 begin
  for i:=na to n do
  begin
   for j:=en to i do
   begin
    CMult(X,B^[j]^[i],R);   Sub(A^[j]^[i],R,C^[j])
   end;
   W:=C^[i];
   if (W.Re=0.0) and (W.Im=0.0) then
  begin
    S:=CAbs(X)*NormB;
    if S>NormA then RMult(eps*S,One,W) else
    RMult(eps*NormA,One,W)
   end;
   Neg(C^[en]);
   B^[en]^[i]:=C^[en];
   VecProd(C^[na],1,B^[en]^[na],1,B^[en]^[i],0,i-na,W);
  end
 end;
end;
for en:=1 to n do B^[en]^[en]:=One;
for i:=1 to n do
for j:=1 to n do
begin
 X:=Zero;
 VecProd(ZET^[j]^[1],1,B^[i]^[1],1,X,0,i,Skip);
 Sub(Zero,X,A^[j]^[i])
end;
for i:=1 to n do
for j:=1 to n do
begin
 X:=Zero;
 VecProd(B^[i]^[i],1,QU^[i]^[j],n,X,0,n-i+1,Skip);
```

```
  Sub(Zero,X,ZET^[i]^[j])
 end;
 for i:=1 to n do Move(ZET^[i]^[1],B^[i]^[1],n shl 4);
 FreeMem(C,n shl 4)
end;

function qz(n,itm:Integer;A,B:CompPtr2;diagA,diagB,WW:
           CompPtr1):Boolean;
var
 QU,ZET:CompPtr2;
 i,j:Integer;
 F,G,NormA,NormB:Double;
 Suc:Boolean;

function check0(n,k:Integer;var i:Integer;C:CompPtr2):Boolean;
var
 j:Integer;
 Nul,Kon:Boolean;
begin
 Nul:=False;
 Kon:=False;
 j:=1;
 repeat
  Nul:=(C^[i]^[j].Re<>0.0) or (C^[i]^[j].Im<>0.0);
  j:=j+1;
  if j=i-k then if i<n then
  begin
   j:=1;
   i:=i+1
  end
  else Kon:=True
 until Nul or Kon;
 check0:=Nul
end;

begin             {body of qz}
 GetMem(ZET,n shl 2);
 for i:=1 to n do GetMem(ZET^[i],n shl 4);
 GetMem(QU,n shl 2);
 for i:=1 to n do GetMem(QU^[i],n shl 4);
 for i:=1 to n do
 begin
  FillChar(ZET^[i]^[1],n shl 4,#0);
  ZET^[i]^[i]:=One;
  FillChar(QU^[i]^[1],n shl 4,#0);
  QU^[i]^[i]:=One
 end;
 i:=2;  Time1:=0;
 if check0(n,0,i,B) then
 begin
  unittr(n,A,B,QU);
  unithes(n,A,B,QU,ZET);
 end
  else
```

```
 begin
  i:=3;
  if n>=3 then
  if check0(n,1,i,A) then unithes(n,A,B,QU,ZET);
  end;
 end;
 NormA:=0.0;   NormB:=0.0;
 for i:=1 to n do
 begin
  F:=0.0;  G:=0.0;
  for j:=1 to n do F:=F+SAbs(A^[j]^[i]);
  for j:=1 to n do G:=G+SAbs(B^[j]^[i]);
  if F>NormA then NormA:=F;
  if G>NormB then NormB:=G;
 end;
 SUC:=iter(n,itm,NormA,NormB,A,B,QU,ZET,diagA,diagB,WW);
 if Suc then
 begin
  fin(n,NormA,NormB,A,B,QU,ZET,WW);
  qz:=True
 end
 else qz:=False;
 FreeMem(ZET,n shl 2);
 for i:=1 to n do FreeMem(ZET^[i],n shl 4);
 FreeMem(QU,n shl 2);
 for i:=1 to n do FreeMem(QU^[i],n shl 4)
end;

end.
```

WARTOŚCI WŁASNE ZAGADNIENIA Ax = λBx Z MACIERZAMI RZECZYWISTYMI

Plik QZVALR.PAS

Opis części zewnętrznej

```
INTERFACE
uses NewType,VecReal;
function qz(n,itm:Integer;A,B:DoublePtr2;CNT:IntPtr1):Boolean;
end;
```

Opis parametrów formalnych:

```
qz(n,itm,A,B,CNT);
```

Parametry wejściowe:

n - stopień macierzy;

itm - maksymalna liczba iteracji dla wydzielenia się jednej wartości własnej (w przypadku zerowej wartości tego parametru ustawiana jest wartość domniemana itm = 30);

A - tablica dwuwymiarowa zawierająca w wierszach 1,...,n zadaną macierz A;

B - tablica dwuwymiarowa zawierająca w wierszach 1,...,n zadaną macierz B;

Parametry wyjściowe:

A - tablica dwuwymiarowa, w której pierwszym wierszu zapisane są części rzeczywiste, a w drugim wierszu części urojone elementów diagonalnych macierzy trójkątnej górnej QAZ;

B - tablica dwuwymiarowa zawierająca w wierszach 1,...,n macierz trójkątną górną QBZ;

UWAGA: w celu uzyskania wartości własnych należy wykonać dzielenie:

$$(\lambda_i).Re = A^[1]^[i]/B^[i]^[i]$$

$$(\lambda_i).Im = A^[2]^[i]/B^[i]^[i]$$

CNT - tablica jednowymiarowa zawierająca liczby iteracji wykonanych do chwili wydzielenia się poszczególnych wartości własnych; jeżeli dwie wartości własne zostały wydzielone jednocześnie jako para, to liczba iteracji podana jest z dodatnim znakiem dla pierwszej wartości własnej, z ujemnym - dla drugiej.

qz - przyjmuje wartość TRUE, jeżeli podczas obliczania każdej wartości własnej nie została przekroczona maksymalna liczba iteracji (itm); w przeciwnym razie proces iteracyjny zostaje przerwany, co sygnalizowane jest wartością qz = FALSE.

WARTOŚCI I PRAWE WEKTORY WŁASNE ZAGADNIENIA $Ax = \lambda Bx$ Z MACIERZAMI RZECZYWISTYMI

Plik QZVECR.PAS

Opis części zewnętrznej

```
INTERFACE

uses NewType,VecReal;
function qz(n,itm:Integer;A,B:DoublePtr2;WR,WI:DoublePtr1):
          Boolean;
end;
```

Opis parametrów formalnych:

```
qz(n,itm,A,B,WR,WI);
```

Parametry wejściowe:

n - stopień macierzy;

itm - maksymalna liczba iteracji dla wydzielenia się jednej wartości własnej (w przypadku zerowej wartości tego parametru ustawiana jest wartość domniemana itm = 30);

A - tablica dwuwymiarowa zawierająca w wierszach 1,...,n zadaną macierz A;

B - tablica dwuwymiarowa zawierająca w wierszach 1,...,n zadaną macierz B;

Parametry wyjściowe:

B - tablica dwuwymiarowa zawierająca:

w pierwszym wierszu - części rzeczywiste elementów diagonalnych macierzy trójkątnej górnej QAZ;

w drugim wierszu - części urojone elementów diagonalnych macierzy trójkątnej górnej QAZ;

w trzecim wierszu - elementy diagonalne macierzy trójkątnej górnej QBZ;

WR - tablica jednowymiarowa zawierająca części rzeczywiste wartości własnych;

WI - tablica jednowymiarowa zawierająca części urojone wartości własnych;

A - tablica dwuwymiarowa zawierająca n prawych wektorów własnych (nie unormowanych), zapisanych kolumnami; dostęp do j-tego elementu i-tego wektora - A^[j]^[i];

UWAGA: stosuje się następujący sposób zapisu wektorów zespolonych w tablicy rzeczywistej: jeśli i-ta oraz i+1-sza wartość własna stanowią parę zespoloną (sprzężoną), wówczas kolumny i oraz i+1 macierzy A zawierają odpowiednio: część rzeczywistą i urojoną wektorów odpowiadających wartości własnej z dodatnią częścią urojoną (traktowanej jako i-ta wartość własna).

qz - przyjmuje wartość TRUE, jeżeli podczas obliczania każdej wartości własnej nie została przekroczona maksymalna liczba iteracji (itm); w przeciwnym razie proces iteracyjny zostaje przerwany, co sygnalizowane jest wartością qz = FALSE.

```
unit QZVECR;

INTERFACE

uses NewType,VecReal;
function qz(n,itm:Integer;A,B:DoublePtr2;WR,WI:DoublePtr1):
            Boolean;

IMPLEMENTATION

const
 tol:Double=1E-280;
 eps:Double=1.11E-16;

procedure orttr(n:Integer;A,B:DoublePtr2);
var
 i,j:Integer;
 S,F,G,H:Double;
begin
 for i:=1 to n-1 do
 begin
  S:=NormE(B^[i]^[i],n,n-i+1);
  if S<tol then G:=0.0 else
  begin
   F:=B^[i]^[i];
   if F<0.0 then G:=Sqrt(S) else G:=-Sqrt(S);
   H:=S-F*G;  B^[i]^[i]:=F-G;
   for j:=i+1 to n do Househ(B^[i]^[i],n,H,B^[i]^[j],n,n-i+1);
   for j:=1 to n do Househ(B^[i]^[i],n,H,A^[i]^[j],n,n-i+1)
  end;
  B^[i]^[i]:=G
 end;
 for i:=1 to n-1 do  FillChar(B^[i+1]^[1],i shl 3,#0);
end;
```

```
procedure orthes(n:Integer;A,B,ZET:DoublePtr2);
var
 i,j:Integer;
 S,C,W:Double;
begin
 for j:=1 to n-2 do
 for i:=n downto j+2 do
 begin
  C:=A^[i-1]^[j];
  S:=A^[i]^[j];
  W:=Sqrt(Sqr(C)+Sqr(S));
  if W>tol then
  begin
   C:=C/W;   S:=S/W;
   A^[i-1]^[j]:=W;
   A^[i]^[j]:=0.0;
   Rotate(S,C,A^[i-1]^[j+1],A^[i]^[j+1],1,n-j);
   Rotate(S,C,B^[i-1]^[i-1],B^[i]^[i-1],1,n-i+2);
   C:=B^[i]^[i];
   S:=B^[i]^[i-1];
   W:=Sqrt(Sqr(C)+Sqr(S));
   if W>tol then
   begin
    C:=C/W;   S:=-S/W;
    B^[i]^[i]:=W;
    B^[i]^[i-1]:=0.0;
    Rotate(S,C,B^[1]^[i-1],B^[1]^[i],n,i-1);
    Rotate(S,C,A^[1]^[i-1],A^[1]^[i],n,n);
    Rotate(S,C,ZET^[1]^[i-1],ZET^[1]^[i],n,n);
   end
  end
 end;
end;

function iter(n:Integer;var itm:Integer;A,B,ZET:DoublePtr2;
    dAR,dAI,dB,WR,WI:DoublePtr1;NormA,NormB:Double):Boolean;
var
 en,na,l,i,j,it,rc:Integer;
 RR,X,Y,W,C,S:Double;
 NmA,NmB,F,G,H:Double;
 WA,NL,NM:Boolean;

procedure Hous1(Q,X,Y:Double;var A,B:Double);
var
 P:Double;
begin
 P:=A+Q*B;
 B:=B-P*Y;
 A:=A-P*X
end;

procedure Hous2(Q,R,X,Y,Z:Double;var A,B,C:Double);
var
 P:Double;
```

```
begin
 P:=A+Q*B+R*C;
 A:=A-P*X;
 B:=B-P*Y;
 C:=C-P*Z
end;

procedure contsub1(en,na,l:Integer;var m:Integer;T:Double;
                   var P,Q:Double);
var
 i,j,k:Integer;
 X,Y,S,F:Double;
 NS:Boolean;
begin
 i:=en;   NS:=False;
 if en>=3 then
 repeat
  i:=i-1;
  X:=A^[i]^[i-1]*A^[i+1]^[i];
  Y:=A^[i]^[i]-T*B^[i]^[i];
  if Y<>0.0 then NS:=Abs(X)<=eps*Abs(Y)*NormA
 until (NS) or (i<=l+1);
 if (not NS) or (i<=l+1) then i:=l;
 NS:=True;
 k:=0;
 repeat
  k:=k+1;
  NS:=Abs(B^[i]^[i])<=eps*NormB;
  if not NS then m:=i else
  begin
   P:=A^[i]^[i];
   Q:=A^[i+1]^[i];
   F:=Abs(P)+Abs(Q);
   P:=P/F;  Q:=Q/F;
   S:=Sqrt(P*P+Q*Q);
   if P<0.0 then S:=-S;
   A^[i]^[i]:=-F*S;
   A^[i+1]^[i]:=0.0;
   if (i<>l) and (k=1) then A^[i]^[i-1]:=-A^[i]^[i-1];
   P:=P+S;   X:=P/S;   Y:=Q/S;
   Q:=Q/P;
   for j:=i+1 to n do Hous1(Q,X,Y,A^[i]^[j],A^[i+1]^[j]);
   for j:=i+1 to n do Hous1(Q,X,Y,B^[i]^[j],B^[i+1]^[j]);
   i:=i+1
  end
 until (not NS) or (i=en);
 if i=en then begin m:=en; exit end;
 P:=A^[m]^[m]/B^[m]^[m]-T;
 Q:=A^[m+1]^[m]/B^[m]^[m];
 F:=Abs(P)+Abs(Q);
 P:=P/F;   Q:=Q/F
end;
```

```
procedure contsub2(en,na:Integer;var l,m:Integer;
   Dn,Dm,Amn,Anm:Double;varP,Q,R:Double;var DBL:Boolean);
var
 i:Integer;
 S,C,W:Double;
 NL,NM:Boolean;

procedure pqr(i:Integer;Dn,Dm,Amn,Anm:Double;var P,Q,R:Double);
var
 D1,D2,A1,A2,B1,B2,X:Double;
begin
 D1:=A^[i]^[i]/B^[i]^[i];
 D2:=A^[i+1]^[i+1]/B^[i+1]^[i+1];
 A1:=A^[i]^[i+1]/B^[i+1]^[i+1];
 A2:=A^[i+1]^[i]/B^[i]^[i];
 B1:=B^[i]^[i]/A^[i+1]^[i];
 B2:=B^[i]^[i+1]/B^[i+1]^[i+1];
 R:=Dn-D1;  Q:=Dm-D1;
 P:=(R*Q-Amn+Anm*D1)*B1+A1-D1*B2;
 Q:=D2-D1-R-Q-A2*B2+Anm;
 R:=A^[i+2]^[i+1]/B^[i+1]^[i+1];
 X:=Abs(P)+Abs(Q)+Abs(R);
 P:=P/X;   Q:=Q/X;   R:=R/X;
end;

begin
 for i:=en-2 downto 1 do
 begin
  S:=Abs(B^[i]^[i]);
  C:=Abs(B^[i+1]^[i+1]);
  if (S>tol) and (C>tol) then
  begin
   pqr(i,Dn,Dm,Amn,Anm,P,Q,R);
   if i=1 then begin m:=1; exit end;
   if (Abs(A^[i]^[i-1])*(Abs(Q)+Abs(R))<=eps*Abs(P)*NormA
   then
   begin
    m:=i;  exit
   end
  end
 end;
 m:=1;
 NL:=Abs(B^[i]^[i]<=eps*NormB;
 NM:=Abs(B^[i+1]^[i+1]<=eps*NormB;
 if NL then
 begin
  repeat
   C:=A^[i]^[i];   S:=A^[i+1]^[i];
   W:=Sqrt(C*C+S*S);
   if W>tol then
   begin
    C:=C/W;  S:=S/W;
    A^[i]^[i]:=W;   A^[i+1]^[i]:=0.0;
    Rotate(S,C,A^[i]^[i+1],A^[i+1]^[i+1],1,n-i);
```

```
    Rotate(S,C,B^[i]^[i+1],B^[i+1]^[i+1],1,n-i);
   end;
   i:=i+1;   l:=i;  m:=l;
   NL:=Abs(B^[i]^[i])<=eps*NormB;
  until (not NL) or (i=en);
  if i>=na then begin m:=en; exit end;
  NM:=Abs(B^[i+1]^[i+1])<=eps*NormB;;
 end;
 if NM then begin DBL:=False; exit end
 else pqr(l,Dn,Dm,Amn,Anm,P,Q,R)
end;

procedure step1(en,na,l,m,k:Integer;var P,Q:Double);
var
 i,j:Integer;
 C,S,X,Y:Double;
 NL:Boolean;
begin
 if k<>m then
 begin
  P:=A^[k]^[k-1];
  Q:=A^[k+1]^[k-1];
  X:=Abs(P)+Abs(Q);
  if X<tol then exit;
  P:=P/X;   Q:=Q/X
 end;
 S:=Sqrt(P*P+Q*Q);
 if P<0.0 then S:=-S;
 if k<>m then
 begin
  A^[k]^[k-1]:=-S*X;
  A^[k+1]^[k-1]:=0.0;
 end
  else
 if l<>m then
 begin
  A^[k]^[k-1]:=-A^[k]^[k-1];
  A^[k+1]^[k-1]:=0.0;
 end;
 P:=P+S;   X:=P/S;   Y:=Q/S;
 Q:=Q/P;
 for j:=k to n do Hous1(Q,X,Y,A^[k]^[j],A^[k+1]^[j]);
 for j:=k to n do Hous1(Q,X,Y,B^[k]^[j],B^[k+1]^[j]);
 C:=B^[k+1]^[k+1];
 S:=B^[k+1]^[k];
 X:=Sqrt(C*C+S*S);
 if X>tol then
 begin
  C:=C/X;   S:=-S/X;
  B^[k+1]^[k+1]:=X;
  B^[k+1]^[k]:=0.0;
  Rotate(S,C,B^[1]^[k],B^[1]^[k+1],n,k);
  if k<>na then Rotate(S,C,A^[1]^[k],A^[1]^[k+1],n,k+2)
  else Rotate(S,C,A^[1]^[k],A^[1]^[k+1],n,en);
```

```
  Rotate(S,C,ZET^[1]^[k],ZET^[1]^[k+1],n,n);
 end
end;

procedure step2(en,na,l,m,k:Integer;var P,Q,R:Double);
var
 i,j:Integer;
 C,S,Z,X,Y,B1:Double;
 NL:Boolean;
begin
 NL:=k<>na;
 if k<>m then
 begin
  P:=A^[k]^[k-1];
  Q:=A^[k+1]^[k-1];
  if NL then R:=A^[k+2]^[k-1] else R:=0.0;
  X:=Abs(P)+Abs(Q)+Abs(R);
  if X=0.0 then exit;
  P:=P/X;    Q:=Q/X;    R:=R/X
 end;
 S:=Sqrt(P*P+Q*Q+R*R);
 if P<0.0 then S:=-S;
 if k<>m then
 begin
  A^[k]^[k-1]:=-S*X;
  A^[k+1]^[k-1]:=0.0;
  if NL then A^[k+2]^[k-1]:=0.0
 end
  else
 if l<>m then
 begin
  A^[k]^[k-1]:=-A^[k]^[k-1];
  A^[k+1]^[k-1]:=0.0;
  if NL then A^[k+2]^[k-1]:=0.0
 end;
 P:=P+S;    X:=P/S;    Y:=Q/S;    Z:=R/S;
 Q:=Q/P;    R:=R/P;
 if NL then
 for j:=k to n do
 begin
  Hous2(Q,R,X,Y,Z,A^[k]^[j],A^[k+1]^[j],A^[k+2]^[j]);
  Hous2(Q,R,X,Y,Z,B^[k]^[j],B^[k+1]^[j],B^[k+2]^[j]);
 end
  else
 for j:=k to n do
 begin
  Hous1(Q,X,Y,A^[k]^[j],A^[k+1]^[j]);
  Hous1(Q,X,Y,B^[k]^[j],B^[k+1]^[j]);
 end;
 if NL then
 begin
  P:=B^[k+2]^[k+2];
  Q:=B^[k+2]^[k+1];
  R:=B^[k+2]^[k];
```

```
  X:=Abs(P)+Abs(Q)+Abs(R);
  if X=0.0 then exit;
  P:=P/X;   Q:=Q/X;   R:=R/X;
  S:=Sqrt(P*P+Q*Q+R*R);
  if P<0.0 then S:=-S;
  B^[k+2]^[k+2]:=-S*X;
  for j:=k to k+1 do B^[k+2]^[j]:=0.0;
  P:=P+S;   X:=P/S;   Y:=Q/S;   Z:=R/S;
  Q:=Q/P;   R:=R/P;
  for j:=k+1 downto 1 do
  Hous2(Q,R,X,Y,Z,B^[j]^[k+2],B^[j]^[k+1],B^[j]^[k]);
  i:=k+3;
  if i>en then i:=en;
  for j:=i downto 1 do
  Hous2(Q,R,X,Y,Z,A^[j]^[k+2],A^[j]^[k+1],A^[j]^[k]);
  for j:=n downto 1 do
  Hous2(Q,R,X,Y,Z,ZET^[j]^[k+2],ZET^[j]^[k+1],ZET^[j]^[k]);
  C:=B^[k+1]^[k+1];
  S:=B^[k+1]^[k];
  Z:=Sqrt(Sqr(C)+Sqr(S));
  if Z>tol then
  begin
   C:=C/Z;   S:=-S/Z;
   B^[k+1]^[k+1]:=Z;   B^[k+1]^[k]:=0.0;
   Rotate(S,C,B^[1]^[k],B^[1]^[k+1],n,k);
   Rotate(S,C,A^[1]^[k],A^[1]^[k+1],n,i);
   Rotate(S,C,ZET^[1]^[k],ZET^[1]^[k+1],n,n)
  end
 end
  else
 begin
  C:=B^[en]^[en];
  S:=B^[en]^[na];
  Z:=Sqrt(Sqr(C)+Sqr(S));
  if Z>tol then
  begin
   C:=C/Z;   S:=-S/Z;
   B^[en]^[en]:=Z;
   B^[en]^[na]:=0.0;
   Rotate(S,C,B^[1]^[na],B^[1]^[en],n,na);
   Rotate(S,C,A^[1]^[na],A^[1]^[en],n,en);
   Rotate(S,C,ZET^[1]^[na],ZET^[1]^[en],n,n)
  end
 end
end;

procedure nextw(en,na,itm:Integer;var l,it,rc:Integer;
                var RR,Y:Double;var WA,NL,NM:Boolean);
var
 m,k:Integer;
 T,P,Q,R,V,Z,W,Dm,Dn,Amn,Anm,Det:Double;
 DBL:Boolean;
begin
 repeat
```

```
l:=en+1;   T:=0.0;
NL:=False;
if en>=2 then
repeat
 l:=l-1;
 NL:=Abs(A^[l]^[l-1])<=eps*NormA
until (l=2) OR (NL);
if NL then A^[l]^[l-1]:=0.0 else l:=1;
if l=en then exit;
if (((it=10) or (it=20)) and (it<itm)) and (l<>na) then
begin
 X:=B^[en-2]^[en-2];
 if Abs(X)<=eps*NormB then X:=eps*NormB;
 V:=A^[na]^[en-2]/X;
 X:=B^[na]^[na];
 if Abs(X)<=eps*NormB then X:=eps*NormB;
 Z:=A^[en]^[na]/X;
 T:=Abs(Z)+Abs(V);
 if DBL then
 begin
  Dn:=0.75*T;  Dm:=Dn;  Anm:=0.0;  Amn:=-0.4375*Sqr(T)
 end
end
 else
begin
 if it=itm then
 begin
  WA:=True;  exit
 end;
 NM:=Abs(B^[en]^[en])<=eps*NormB;
 NL:=Abs(B^[na]^[na])<=eps*NormB;
 if (not NL) and (not NM) then
 begin
  Dn:=A^[en]^[en]/B^[en]^[en];
  Dm:=A^[na]^[na]/B^[na]^[na];
  V:=A^[na]^[en]/B^[en]^[en];
  Z:=A^[en]^[na]/B^[na]^[na];
  X:=B^[na]^[en]/B^[en]^[en];
  Amn:=V*Z;
  Anm:=Z*X;
  W:=Amn-Dm*Anm;
  Y:=Dn-Dm-Anm;  P:=0.5*Y;
  Q:=P*P+W;
  Y:=Sqrt(Abs(Q));
  if Q>=0.0 then
  begin                    {para rzeczywista}
   rc:=0;
   DBL:=False;
   if P<0.0 then Y:=-Y;
   Y:=P+Y;
   RR:=Dm+Y;
   if l=na then exit;
   if Y=0.0 then C:=Dm else C:=Dm-W/Y;
   if Abs(Dn-RR)<Abs(Dn-C) then T:=RR else T:=C
```

```
    end
     else
    begin                    {para zespolona}
     rc:=1;
     DBL:=True;
     T:=Dm+P;    RR:=T;
     if l=na then exit
    end
   end
    else
   begin
    if l=na then begin rc:=0; exit end;
    DBL:=FALSE;
    Det:=-A^[na]^[na]*A^[en]^[en]+A^[en]^[na]*A^[na]^[en];
    V:=A^[en]^[na]*B^[na]^[en];
    if (NL) and (NM) then
    begin
     if V<>0.0 then T:=Det/V
    end
     else
    if NL then
    begin
     Z:=V-A^[na]^[na]*B^[en]^[en];
     if Z<>0.0 then T:=Det/Z
    end
     else
    begin
     Z:=V-A^[en]^[en]*B^[na]^[na];
     if Z<>0.0 then T:=Det/Z
    end;
    X:=A^[l]^[l]-B^[l]^[l]*T;
    if Abs(X)>Abs(A^[l+1]^[l]) then T:=0.0
   end
  end;
  it:=it+1;
  if DBL then contsub2(en,na,l,m,Dn,Dm,Amn,Anm,P,Q,R,DBL);
  if not DBL then
  begin
   contsub1(en,na,l,m,T,P,Q);
   for k:=m to na do
   step1(en,na,l,m,k,P,Q);
  end
   else
  for k:=m to na do
  step2(en,na,l,m,k,P,Q,R);
 until WA;
end;

begin                {body of iter}
 en:=n;  WA:=False;
 if itm=0 then itm:=30;
 repeat
  it:=0;   na:=en-1;
  nextw(en,na,itm,l,it,rc,RR,Y,WA,NL,NM);
```

```
if WA then
begin
 iter:=False;  exit
end;
if l=en then
begin
 dAR^[en]:=A^[en]^[en];  dAI^[en]:=0.0;
 dB^[en]:=B^[en]^[en];
 if dB^[en]=0.0 then dB^[en]:=eps*NormB;
 WR^[en]:=dAR^[en]/dB^[en];
 WI^[en]:=0.0;
 en:=na
end
 else
begin
 if (not NL) and (not NM) then
 if rc=0 then
 begin
  NmA:=Abs(A^[na]^[na])+Abs(A^[na]^[en]);
  F:=Abs(A^[en]^[na])+Abs(A^[en]^[en]);
  if F>NmA then NmA:=F;
  NmB:=Abs(B^[na]^[na])+Abs(B^[na]^[en]);
  F:=Abs(B^[en]^[en]);
  if F>NmB then NmB:=F;
  NmB:=NmB*Abs(RR);
  X:=A^[na]^[na]-RR*B^[na]^[na];
  W:=A^[na]^[en]-RR*B^[na]^[en];
  G:=Abs(X)+Abs(W);
  C:=A^[en]^[en]-RR*B^[en]^[en];
  S:=Abs(A^[en]^[na])+Abs(C);
  if G>S then
  begin
   C:=W;   S:=X
  end
  else  S:=A^[en]^[na];
  G:=Sqrt(C*C+S*S);
  if G>tol then
  begin
   C:=C/G;   S:=-S/G;
   Rotate(S,C,A^[1]^[na],A^[1]^[en],n,en);
   Rotate(S,C,B^[1]^[na],B^[1]^[en],n,en);
   Rotate(S,C,ZET^[1]^[na],ZET^[1]^[en],n,n);
  end;
  if NmB<NmA then
  begin
   C:=B^[na]^[na];   S:=B^[en]^[na];
   G:=Sqrt(C*C+S*S);
   if G>tol then
   begin
    C:=C/G;   S:=S/G;
    B^[na]^[na]:=G;
    A^[na]^[na]:=C*A^[na]^[na]+S*A^[en]^[na];
    Rotate(S,C,A^[na]^[en],A^[en]^[en],1,n-na);
    Rotate(S,C,B^[na]^[en],B^[en]^[en],1,n-na);
```

```
  end
 end
  else
 begin
  C:=A^[na]^[na];   S:=A^[en]^[na];
  G:=Sqrt(C*C+S*S);
  if G>tol then
  begin
   C:=C/G;   S:=S/G;
   A^[na]^[na]:=G;
   B^[na]^[na]:=C*B^[na]^[na]+S*B^[en]^[na];
   Rotate(S,C,A^[na]^[en],A^[en]^[en],1,n-na);
   Rotate(S,C,B^[na]^[en],B^[en]^[en],1,n-na);
  end
 end;
 A^[en]^[na]:=0.0;
 B^[en]^[na]:=0.0;
end
 else
begin
 WR^[na]:=RR;  WR^[en]:=RR;
 WI^[na]:=Y;   WI^[en]:=-Y;
 dB^[na]:=B^[na]^[na];   dB^[en]:=B^[en]^[en];
 dAR^[na]:=dB^[na]*WR^[na];
 dAR^[en]:=dB^[en]*WR^[en];
 dAI^[na]:=dB^[na]*WI^[na];
 dAI^[en]:=dB^[en]*WI^[en];
end
 else
begin
 if NL then
 begin
  C:=A^[na]^[na];   S:=A^[en]^[na];
  G:=Sqrt(C*C+S*S);
  if G>tol then
  begin
   C:=C/G;   S:=S/G;
   A^[na]^[na]:=G;
   A^[en]^[na]:=0.0;
   Rotate(S,C,A^[na]^[en],A^[en]^[en],1,n-na);
   Rotate(S,C,B^[na]^[en],B^[en]^[en],1,n-na);
  end
 end
  else
 begin
  C:=A^[en]^[en];   S:=A^[en]^[na];
  G:=Sqrt(C*C+S*S);
  if G>tol then
  begin
   C:=C/G;   S:=-S/G;
   A^[en]^[en]:=G;   A^[en]^[na]:=0.0;
   Rotate(S,C,A^[1]^[na],A^[1]^[en],n,na);
   Rotate(S,C,B^[1]^[na],B^[1]^[en],n,na);
   Rotate(S,C,ZET^[1]^[na],ZET^[1]^[en],n,n);
```

```
      end;
     end;
    end;
    if rc=0 then
    begin
     dAI^[na]:=0.0;  dAI^[en]:=0.0;
     WI^[na]:=0.0;   WI^[en]:=0.0;
     dAR^[na]:=A^[na]^[na];   dAR^[en]:=A^[en]^[en];
     dB^[na]:=B^[na]^[na];    dB^[en]:=B^[en]^[en];
     if dB^[en]=0.0 then dB^[en]:=eps*NormB;
     if dB^[na]=0.0 then dB^[na]:=eps*NormB;
     WR^[en]:=dAR^[en]/dB^[en];
     WR^[na]:=dAR^[na]/dB^[na]
    end;
    en:=en-2
   end
  until en=0;
  iter:=True
end;

procedure fin(n:Integer;A,B,ZET:DoublePtr2;WR,WI:DoublePtr1;
              NormA,NormB:Double);
var
 i,j,na,en,m:Integer;
 X,Y,W,Z,S,T,P,Q,R,RR,RI,QA,QP,YR,YI,VR,VI:Double;
 C:DoublePtr1;

procedure CDiv(XR,XI,YR,YI:Double;var ZR,ZI:Double);
var
 H:Double;
begin
 if Abs(YR)>Abs(YI) then
 begin
  H:=YI/YR;  YR:=H*YI+YR;
  ZR:=(XR+H*XI)/YR;
  ZI:=(XI-H*XR)/YR
 end
  else
 begin
  H:=YR/YI;  YI:=H*YR+YI;
  ZR:=(H*XR+XI)/YI;
  ZI:=(H*XI-XR)/YI
 end
end;

begin
 GetMem(C,n shl 3);  FillChar(C^[1],n shl 3,#0);
 for en:=n downto 1 do
 begin
  P:=WR^[en];  Q:=WI^[en];   na:=en-1;
  if Q=0.0 then                        {wektor rzeczywisty}
  if (Abs(B^[en]^[en])<=eps*NormB) then
  begin
   B^[en]^[en]:=1.0;
   for i:=na downto 1 do
```

```
 begin
  W:=B^[i]^[i];
  if W=0.0 then W:=eps*NormB;
  B^[i]^[en]:=-B^[i]^[en];
  VecScal(B^[i]^[i+1],1,B^[i+1]^[en],n,B^[i]^[en],0,na-i,W);
 end
end
 else
begin
 B^[en]^[en]:=1.0;   m:=en;
 for i:=na downto 1 do
 begin
  for j:=m to na do C^[j]:=A^[i]^[j]-P*B^[i]^[j];
  W:=A^[i]^[i]-P*B^[i]^[i];
  R:=-(A^[i]^[en]-P*B^[i]^[en]);
  VecScal(C^[m],1,B^[m]^[en],n,R,0,na-m+1,Skip);
  if WI^[i]<0.0 then
  begin
   Z:=W;   S:=R
  end
   else
  begin
   m:=i;
   if WI^[i]=0.0 then
   begin
    if W=0.0 then
    begin
     X:=Abs(P)*NormB;
     if X>NormA then W:=eps*X else W:=eps*NormA
    end;
    B^[i]^[en]:=R/W
   end
    else
   begin
    X:=A^[i]^[i+1]-P*B^[i]^[i+1];
    Y:=A^[i+1]^[i];
    Q:=Z*W-X*Y;
    if Q=0.0 then
    begin
     T:=Abs(P)*NormB;
     if NormA>T then T:=NormA;
     Q:=eps*T*(Abs(W)+Abs(X)+Abs(Z)+Abs(Y))
    end;
    T:=(Z*R-X*S)/Q;
    B^[i]^[en]:=T;
    if Abs(X)>Abs(Z) then Q:=(R-W*T)/X  else Q:=(S-Y*T)/Z;
    B^[i+1]^[en]:=Q
   end   {WI^[i]>0}
  end   {WI^[i]>=0}
 end   {i}
end    {wektor rzeczywisty}
 else
if Q<0.0 then
begin                  {wektor zespolony dla lambda=P-i*Q}
```

```
m:=na;
YR:=A^[na]^[na]-P*B^[na]^[na];
YI:=Q*B^[na]^[na];
X:=A^[en]^[na];
if Abs(X)>(Abs(YR)+Abs(YI)) then
begin
 B^[na]^[na]:=-(A^[en]^[en]-P*B^[en]^[en])/X;
 B^[na]^[en]:=-Q*B^[en]^[en]/X
end
 else
CDiv(-A^[na]^[en]+P*B^[na]^[en],-Q*B^[na]^[en],YR,YI,
     B^[na]^[na],B^[na]^[en]);
B^[en]^[na]:=1.0;   B^[en]^[en]:=0.0;
for i:=na-1 downto 1 do
begin
 for j:=m to na do  C^[j]:=A^[i]^[j]-P*B^[i]^[j];
 W:=A^[i]^[i]-P*B^[i]^[i];
 QA:=Q*B^[i]^[i];
 RR:=-(A^[i]^[en]-P*B^[i]^[en]);
 RI:=-B^[i]^[en];
 X:=0.0;
 VecScal(C^[m],1,B^[m]^[na],n,RR,0,na-m+1,Skip);
 VecScal(B^[i]^[m],1,B^[m]^[en],n,X,0,na-m+1,Skip);
 RR:=RR-Q*X;
 VecScal(B^[i]^[m],1,B^[m]^[na],n,RI,0,na-m+1,Skip);
 RI:=Q*RI;
 VecScal(C^[m],1,B^[m]^[en],n,RI,0,na-m+1,Skip);
 if WI^[i]<0.0 then
 begin
  Z:=W;  QP:=QA;  R:=RR;  S:=RI
 end
  else
 begin
  m:=i;
  if WI^[i]=0.0 then CDiv(RR,RI,W,QA,B^[i]^[na],B^[i]^[en])
  else
  begin
   YR:=A^[i]^[i+1]-P*B^[i]^[i+1];
   YI:=Q*B^[i]^[i+1];
   X:=A^[i+1]^[i];
   VR:=W*Z-QA*QP-YR*X;
   VI:=QA*Z+QP*W-YI*X;
   if (VR=0.0) and (VI=0.0) then
   begin
    T:=(Abs(P)+Abs(Q))*NormB;
    if NormA>T then T:=NormA;
    VR:=eps*T*(Abs(W)+Abs(QA)+Abs(YR)+Abs(X)+Abs(Z)+Abs(QP))
   end;
   CDiv(-YR*R+YI*S+Z*RR-QP*RI, -YR*S-YI*R+Z*RI+QP*RR, VR,VI,
        B^[i]^[na],B^[i]^[en]);
   if (Abs(YR)+Abs(YI))>(Abs(Z)+Abs(QP)) then
   CDiv(RR-W*B^[i]^[na]+QA*B^[i]^[en],
        RI-W*B^[i]^[en]-QA*B^[i]^[na],YR,YI,
        B^[i+1]^[na],B^[i+1]^[en])
```

```
      else
      CDiv(R-X*B^[i]^[na], S-X*B^[i]^[en], Z,QP,
           B^[i+1]^[na],B^[i+1]^[en]);
     end  {WI^[i]>0}
    end  {WI^[i]>=0}
   end  {i}
  end  {wektor zespolony}
 end;

 for i:=n downto 1 do
 begin
  if WI^[i]<0.0 then
  for j:=1 to n do
  begin
   X:=0.0;   Y:=0.0;
   VecScal(ZET^[j]^[1],1,B^[1]^[i],n,X,0,i,Skip);
   VecScal(ZET^[j]^[1],1,B^[1]^[i-1],n,Y,0,i,Skip);
   A^[j]^[i]:=-X;   A^[j]^[i-1]:=-Y
  end
   else
  if WI^[i]=0.0 then
  for j:=1 to n do
  begin
   X:=0.0;
   VecScal(ZET^[j]^[1],1,B^[1]^[i],n,X,0,i,Skip);
   A^[j]^[i]:=-X
  end
 end;
 FreeMem(C,n shl 3);
end;

function qz(n,itm:Integer;A,B:DoublePtr2;WR,WI:DoublePtr1):
          Boolean;
var
 ZET:DoublePtr2;
 dAR,dAI,dB:DoublePtr1;
 i,j:Integer;
 F,G,NormA,NormB:Double;
 Suc:Boolean;

function check0(n,k:Integer;var i:Integer;C:DoublePtr2):
              Boolean;
var
 j:Integer;
 Nul,Kon:Boolean;
begin
 Nul:=False;  Kon:=False;
 j:=1;
 repeat
  Nul:=C^[i]^[j]<>0.0;
  j:=j+1;
  if j=i-k then if i<n then
  begin
   j:=1;  i:=i+1
  end
```

```
  else Kon:=True
 until Nul or Kon;
 check0:=Nul
end;

begin     {body of qz}
 GetMem(ZET,n shl 2);
 for i:=1 to n do GetMem(ZET^[i],n shl 3);
 GetMem(dAR,n shl 3);
 GetMem(dAI,n shl 3);
 GetMem(dB,n shl 3);
 for i:=1 to n do
 begin
  FillChar(ZET^[i]^[1],n shl 3,#0);
  ZET^[i]^[i]:=1.0;
 end;
 i:=2;
 if check0(n,0,i,B) then
 begin
  orttr(n,A,B);
  orthes(n,A,B,ZET)
 end
  else
 begin
  i:=3;
  if n>=3 then
  if check0(n,1,i,A) then orthes(n,A,B,ZET);
 end;
 NormA:=0.0;   NormB:=0.0;
 for i:=1 to n do
 begin
  F:=0.0;  G:=0.0;
  for j:=1 to n do F:=F+Abs(A^[j]^[i]);
  for j:=1 to n do G:=G+Abs(B^[j]^[i]);
  if F>NormA then NormA:=F;
  if G>NormB then NormB:=G;
 end;
 SUC:=iter(n,itm,A,B,ZET,dAR,dAI,dB,WR,WI,NormA,NormB);
 if Suc then
 begin
  fin(n,A,B,ZET,WR,WI,NormA,NormB);
  qz:=True
 end
 else qz:=False;
 Move(dAR^[1],B^[1]^[1],n shl 3);
 Move(dAI^[1],B^[2]^[1],n shl 3);
 Move(dB^[1],B^[3]^[1],n shl 3);
 FreeMem(ZET,n shl 2);
 for i:=1 to n do FreeMem(ZET^[i],n shl 3);
 FreeMem(dAR,n shl 3);
 FreeMem(dAI,n shl 3);
 FreeMem(dB,n shl 3);
end;

end.
```

PEŁNE UOGÓLNIONE ZAGADNIENIE WŁASNE Ax = λBx
Z MACIERZAMI RZECZYWISTYMI

Plik QZEIGR.PAS

Opis części zewnętrznej

```
INTERFACE

uses NewType,VecReal;
function qz(n,itm:Integer;A,B:DoublePtr2;diagAR,diagAI,diagB,
          WR,WI:DoublePtr1):Boolean;
end;
```

Opis parametrów formalnych:

```
qz(n,itm,A,B,diagAR,diagAI,diagB,WR,WI);
```

Parametry wejściowe:

n - stopień macierzy;

itm - maksymalna liczba iteracji dla wydzielenia się jednej wartości własnej (w przypadku zerowej wartości tego parametru ustawiana jest wartość domniemana itm = 30);

A - tablica dwuwymiarowa zawierająca w wierszach 1,...,n zadaną macierz A;

B - tablica dwuwymiarowa zawierająca w wierszach 1,...,n zadaną macierz B;

Parametry wyjściowe:

diagAR - tablica jednowymiarowa zawierająca części rzeczywiste elementów diagonalnych macierzy trójkątnej górnej QAZ;

diagAI - tablica jednowymiarowa zawierająca części urojone elementów diagonalnych macierzy trójkątnej górnej AZ;

diagB - tablica jednowymiarowa zawierająca elementy diagonalne macierzy trójkątnej górnej QBZ;

WR - tablica jednowymiarowa zawierająca części rzeczywiste wartości własnych;

WI - tablica jednowymiarowa zawierająca części urojone wartości własnych;

A - tablica dwuwymiarowa zawierająca n prawych wektorów własnych (nie unormowanych), zapisanych kolumnami; dostęp do j-tego elementu i-tego wektora - A^[j]^[i];

B - tablica dwuwymiarowa zawierająca n lewych wektorów własnych (nie unormowanych), zapisanych wierszami; dostęp do j-tego elementu i-tego wektora - B^[i]^[j];

UWAGA: stosuje się następujący sposób zapisu wektorów zespolonych w tablicach rzeczywistych: jeśli i-ta oraz i+1-sza wartość własna stanowią parę zespoloną (sprzężoną), wówczas kolumny i oraz i+1 macierzy A (wiersze i oraz i+1 macierzy B) zawierają odpowiednio: część rzeczywistą i urojoną wektorów prawych (lewych) odpowiadających wartości własnej z dodatnią częścią urojoną (traktowanej jako i-ta wartość własna).

qz - przyjmuje wartość TRUE, jeżeli podczas obliczania każdej wartości własnej nie została przekroczona maksymalna liczba iteracji (itm); w przeciwnym razie proces iteracyjny zostaje przerwany, co sygnalizowane jest wartością qz = FALSE.

SPIS TREŚCI