

NAG C Library Function Document

nag_zgebrd (f08ksc)

1 Purpose

nag_zgebrd (f08ksc) reduces a complex m by n matrix to bidiagonal form.

2 Specification

```
void nag_zgebrd (Nag_OrderType order, Integer m, Integer n, Complex a[],  
    Integer pda, double d[], double e[], Complex tauq[], Complex taup[],  
    NagError *fail)
```

3 Description

nag_zgebrd (f08ksc) reduces a complex m by n matrix A to real bidiagonal form B by a unitary transformation: $A = QBP^H$, where Q and P^H are unitary matrices of order m and n respectively.

If $m \geq n$, the reduction is given by:

$$A = Q \begin{pmatrix} B_1 \\ 0 \end{pmatrix} P^H = Q_1 B_1 P^H,$$

where B_1 is a real n by n upper bidiagonal matrix and Q_1 consists of the first n columns of Q .

If $m < n$, the reduction is given by

$$A = Q(B_1 \ 0)P^H = QB_1P_1^H,$$

where B_1 is a real m by m lower bidiagonal matrix and P_1^H consists of the first m rows of P^H .

The unitary matrices Q and P are not formed explicitly but are represented as products of elementary reflectors (see the f08 Chapter Introduction for details). Functions are provided to work with Q and P in this representation (see Section 8).

4 References

Golub G H and Van Loan C F (1996) *Matrix Computations* (3rd Edition) Johns Hopkins University Press, Baltimore

5 Parameters

1: **order** – Nag_OrderType *Input*

On entry: the **order** parameter specifies the two-dimensional storage scheme being used, i.e., row-major ordering or column-major ordering. C language defined storage is specified by **order = Nag_RowMajor**. See Section 2.2.1.4 of the Essential Introduction for a more detailed explanation of the use of this parameter.

Constraint: **order = Nag_RowMajor** or **Nag_ColMajor**.

2: **m** – Integer *Input*

On entry: m , the number of rows of the matrix A .

Constraint: $m \geq 0$.

3:	n – Integer	<i>Input</i>
<i>On entry:</i> n , the number of columns of the matrix A .		
<i>Constraint:</i> $n \geq 0$.		
4:	a [<i>dim</i>] – Complex	<i>Input/Output</i>
Note: the dimension, dim , of the array a must be at least $\max(1, \text{pda} \times n)$ when order = Nag_ColMajor and at least $\max(1, \text{pda} \times m)$ when order = Nag_RowMajor .		
If order = Nag_ColMajor , the (i, j) th element of the matrix A is stored in a [($j - 1$) \times pda + $i - 1$] and if order = Nag_RowMajor , the (i, j) th element of the matrix A is stored in a [($i - 1$) \times pda + $j - 1$].		
<i>On entry:</i> the m by n matrix A .		
<i>On exit:</i> if $m \geq n$, the diagonal and first super-diagonal are overwritten by the upper bidiagonal matrix B , elements below the diagonal are overwritten by details of the unitary matrix Q and elements above the first super-diagonal are overwritten by details of the unitary matrix P .		
If $m < n$, the diagonal and first sub-diagonal are overwritten by the lower bidiagonal matrix B , elements below the first sub-diagonal are overwritten by details of the unitary matrix Q and elements above the diagonal are overwritten by details of the unitary matrix P .		
5:	pda – Integer	<i>Input</i>
<i>On entry:</i> the stride separating matrix row or column elements (depending on the value of order) in the array a .		
<i>Constraints:</i>		
if order = Nag_ColMajor , pda $\geq \max(1, m)$; if order = Nag_RowMajor , pda $\geq \max(1, n)$.		
6:	d [<i>dim</i>] – double	<i>Output</i>
Note: the dimension, dim , of the array d must be at least $\max(1, \min(m, n))$.		
<i>On exit:</i> the diagonal elements of the bidiagonal matrix B .		
7:	e [<i>dim</i>] – double	<i>Output</i>
Note: the dimension, dim , of the array e must be at least $\max(1, \min(m, n) - 1)$.		
<i>On exit:</i> the off-diagonal elements of the bidiagonal matrix B .		
8:	tauq [<i>dim</i>] – Complex	<i>Output</i>
Note: the dimension, dim , of the array tauq must be at least $\max(1, \min(m, n))$.		
<i>On exit:</i> further details of the unitary matrix Q .		
9:	taup [<i>dim</i>] – Complex	<i>Output</i>
Note: the dimension, dim , of the array taup must be at least $\max(1, \min(m, n))$.		
<i>On exit:</i> further details of the unitary matrix P .		
10:	fail – NagError *	<i>Output</i>
The NAG error parameter (see the Essential Introduction).		

6 Error Indicators and Warnings

NE_INT

On entry, **m** = $\langle value \rangle$.

Constraint: **m** ≥ 0 .

On entry, $\mathbf{n} = \langle \text{value} \rangle$.

Constraint: $\mathbf{n} \geq 0$.

On entry, $\mathbf{pda} = \langle \text{value} \rangle$.

Constraint: $\mathbf{pda} > 0$.

NE_INT_2

On entry, $\mathbf{pda} = \langle \text{value} \rangle$, $\mathbf{m} = \langle \text{value} \rangle$.

Constraint: $\mathbf{pda} \geq \max(1, \mathbf{m})$.

On entry, $\mathbf{pda} = \langle \text{value} \rangle$, $\mathbf{n} = \langle \text{value} \rangle$.

Constraint: $\mathbf{pda} \geq \max(1, \mathbf{n})$.

NE_ALLOC_FAIL

Memory allocation failed.

NE_BAD_PARAM

On entry, parameter $\langle \text{value} \rangle$ had an illegal value.

NE_INTERNAL_ERROR

An internal error has occurred in this function. Check the function call and any array sizes. If the call is correct then please consult NAG for assistance.

7 Accuracy

The computed bidiagonal form B satisfies $QBP^H = A + E$, where

$$\|E\|_2 \leq c(n)\epsilon\|A\|_2,$$

$c(n)$ is a modestly increasing function of n , and ϵ is the **machine precision**.

The elements of B themselves may be sensitive to small perturbations in A or to rounding errors in the computation, but this does not affect the stability of the singular values and vectors.

8 Further Comments

The total number of real floating-point operations is approximately $16n^2(3m - n)/3$ if $m \geq n$ or $16m^2(3n - m)/3$ if $m < n$.

If $m \gg n$, it can be more efficient to first call nag_zgeqrf (f08asc) to perform a QR factorization of A , and then to call nag_zgebrd (f08ksc) to reduce the factor R to bidiagonal form. This requires approximately $8n^2(m + n)$ floating-point operations.

If $m \ll n$, it can be more efficient to first call nag_zgelqf (f08avc) to perform an LQ factorization of A , and then to call nag_zgebrd (f08ksc) to reduce the factor L to bidiagonal form. This requires approximately $8m^2(m + n)$ operations.

To form the unitary matrices P^H and/or Q , this function may be followed by calls to nag_zungbr (f08ktc): to form the m by m unitary matrix Q

```
nag_zungbr (order, Nag_FormQ, m, m, n, &a, pda, tauq, &fail)
```

but note that the second dimension of the array **a** must be at least **m**, which may be larger than was required by nag_zgebrd (f08ksc);

to form the n by n unitary matrix P^H

```
nag_zungbr (order, Nag_FormP, n, n, m, &a, pda, taup, &fail)
```

but note that the first dimension of the array **a**, specified by the parameter **pda**, must be at least **n**, which may be larger than was required by nag_zgebrd (f08ksc).

To apply Q or P to a complex rectangular matrix C , this function may be followed by a call to `nag_zunmbr(f08kuc)`.

The real analogue of this function is nag_zgebrd (f08ksc).

9 Example

To reduce the matrix A to bidiagonal form, where

$$A = \begin{pmatrix} 0.96 - 0.81i & -0.03 + 0.96i & -0.91 + 2.06i & -0.05 + 0.41i \\ -0.98 + 1.98i & -1.20 + 0.19i & -0.66 + 0.42i & -0.81 + 0.56i \\ 0.62 - 0.46i & 1.01 + 0.02i & 0.63 - 0.17i & -1.11 + 0.60i \\ -0.37 + 0.38i & 0.19 - 0.54i & -0.98 - 0.36i & 0.22 - 0.20i \\ 0.83 + 0.51i & 0.20 + 0.01i & -0.17 - 0.46i & 1.47 + 1.59i \\ 1.08 - 0.28i & 0.20 - 0.12i & -0.07 + 1.23i & 0.26 + 0.26i \end{pmatrix}.$$

9.1 Program Text

```

/* nag_zgebrd (f08ksc) Example Program.
*
* Copyright 2001 Numerical Algorithms Group.
*
* Mark 7, 2001.
*/
#include <stdio.h>
#include <nag.h>
#include <nag_stdlib.h>
#include <nagf08.h>

int main(void)
{
    /* Scalars */
    Integer i, j, m, n, pda, d_len, e_len, tauq_len, taup_len;
    Integer exit_status=0;
    NagError fail;
    Nag_OrderType order;
    /* Arrays */
    Complex *a=0, *taup=0, *tauq=0;
    double *d=0, *e=0;

#ifdef NAG_COLUMN_MAJOR
#define A(I,J) a[(J-1)*pda + I - 1]
    order = Nag_ColMajor;
#else
#define A(I,J) a[(I-1)*pda + J - 1]
    order = Nag_RowMajor;
#endif

    INIT_FAIL(fail);
    Vprintf("f08ksc Example Program Results\n");

    /* Skip heading in data file */
    Vscanf("%*[^\n] ");
    Vscanf("%ld%ld%*[^\n] ", &m, &n);
#ifdef NAG_COLUMN_MAJOR
    pda = m;
#else
    pda = n;
#endif
    d_len = MIN(m,n);
    e_len = MIN(m,n)-1;
    tauq_len = MIN(m,n);
    taup_len = MIN(m,n);

    /* Allocate memory */
    if ( !(a = NAG_ALLOC(m * n, Complex)) ||
        !(d = NAG_ALLOC(d_len, double)) ||
        !(e = NAG_ALLOC(e_len, double)) ||
        !(tauq = NAG_ALLOC(tauq_len, Complex)) ||
        !(taup = NAG_ALLOC(taup_len, Complex)) )
        exit_status = 1;
    else
        /* Initialize a to m by n
           matrix with elements
           1 2 3 4
           5 6 7 8
           9 10 11 12
           13 14 15 16
           17 18 19 20
           21 22 23 24
           25 26 27 28
           29 30 31 32
           33 34 35 36
           37 38 39 40
           41 42 43 44
           45 46 47 48
           49 50 51 52
           53 54 55 56
           57 58 59 50
           61 62 63 64
           65 66 67 68
           69 60 61 62
           73 74 75 76
           77 78 79 70
           81 82 83 84
           85 86 87 88
           89 80 81 82
           93 94 95 96
           97 98 99 90
           101 102 103 104
           105 106 107 108
           109 100 101 102
           113 114 115 116
           117 118 119 110
           121 122 123 124
           125 126 127 128
           129 120 121 122
           133 134 135 136
           137 138 139 130
           141 142 143 144
           145 146 147 148
           149 140 141 142
           153 154 155 156
           157 158 159 150
           161 162 163 164
           165 166 167 168
           169 160 161 162
           173 174 175 176
           177 178 179 170
           181 182 183 184
           185 186 187 188
           189 180 181 182
           193 194 195 196
           197 198 199 190
           201 202 203 204
           205 206 207 208
           209 200 201 202
           213 214 215 216
           217 218 219 210
           221 222 223 224
           225 226 227 228
           229 220 221 222
           233 234 235 236
           237 238 239 230
           241 242 243 244
           245 246 247 248
           249 240 241 242
           253 254 255 256
           257 258 259 250
           261 262 263 264
           265 266 267 268
           269 260 261 262
           273 274 275 276
           277 278 279 270
           281 282 283 284
           285 286 287 288
           289 280 281 282
           293 294 295 296
           297 298 299 290
           301 302 303 304
           305 306 307 308
           309 300 301 302
           313 314 315 316
           317 318 319 310
           321 322 323 324
           325 326 327 328
           329 320 321 322
           333 334 335 336
           337 338 339 330
           341 342 343 344
           345 346 347 348
           349 340 341 342
           353 354 355 356
           357 358 359 350
           361 362 363 364
           365 366 367 368
           369 360 361 362
           373 374 375 376
           377 378 379 370
           381 382 383 384
           385 386 387 388
           389 380 381 382
           393 394 395 396
           397 398 399 390
           401 402 403 404
           405 406 407 408
           409 400 401 402
           413 414 415 416
           417 418 419 410
           421 422 423 424
           425 426 427 428
           429 420 421 422
           433 434 435 436
           437 438 439 430
           441 442 443 444
           445 446 447 448
           449 440 441 442
           453 454 455 456
           457 458 459 450
           461 462 463 464
           465 466 467 468
           469 460 461 462
           473 474 475 476
           477 478 479 470
           481 482 483 484
           485 486 487 488
           489 480 481 482
           493 494 495 496
           497 498 499 490
           501 502 503 504
           505 506 507 508
           509 500 501 502
           513 514 515 516
           517 518 519 510
           521 522 523 524
           525 526 527 528
           529 520 521 522
           533 534 535 536
           537 538 539 530
           541 542 543 544
           545 546 547 548
           549 540 541 542
           553 554 555 556
           557 558 559 550
           561 562 563 564
           565 566 567 568
           569 560 561 562
           573 574 575 576
           577 578 579 570
           581 582 583 584
           585 586 587 588
           589 580 581 582
           593 594 595 596
           597 598 599 590
           601 602 603 604
           605 606 607 608
           609 600 601 602
           613 614 615 616
           617 618 619 610
           621 622 623 624
           625 626 627 628
           629 620 621 622
           633 634 635 636
           637 638 639 630
           641 642 643 644
           645 646 647 648
           649 640 641 642
           653 654 655 656
           657 658 659 650
           661 662 663 664
           665 666 667 668
           669 660 661 662
           673 674 675 676
           677 678 679 670
           681 682 683 684
           685 686 687 688
           689 680 681 682
           693 694 695 696
           697 698 699 690
           701 702 703 704
           705 706 707 708
           709 700 701 702
           713 714 715 716
           717 718 719 710
           721 722 723 724
           725 726 727 728
           729 720 721 722
           733 734 735 736
           737 738 739 730
           741 742 743 744
           745 746 747 748
           749 740 741 742
           753 754 755 756
           757 758 759 750
           761 762 763 764
           765 766 767 768
           769 760 761 762
           773 774 775 776
           777 778 779 770
           781 782 783 784
           785 786 787 788
           789 780 781 782
           793 794 795 796
           797 798 799 790
           801 802 803 804
           805 806 807 808
           809 800 801 802
           813 814 815 816
           817 818 819 810
           821 822 823 824
           825 826 827 828
           829 820 821 822
           833 834 835 836
           837 838 839 830
           841 842 843 844
           845 846 847 848
           849 840 841 842
           853 854 855 856
           857 858 859 850
           861 862 863 864
           865 866 867 868
           869 860 861 862
           873 874 875 876
           877 878 879 870
           881 882 883 884
           885 886 887 888
           889 880 881 882
           893 894 895 896
           897 898 899 890
           901 902 903 904
           905 906 907 908
           909 900 901 902
           913 914 915 916
           917 918 919 910
           921 922 923 924
           925 926 927 928
           929 920 921 922
           933 934 935 936
           937 938 939 930
           941 942 943 944
           945 946 947 948
           949 940 941 942
           953 954 955 956
           957 958 959 950
           961 962 963 964
           965 966 967 968
           969 960 961 962
           973 974 975 976
           977 978 979 970
           981 982 983 984
           985 986 987 988
           989 980 981 982
           993 994 995 996
           997 998 999 990
           1001 1002 1003 1004
           1005 1006 1007 1008
           1009 1000 1001 1002
           1013 1014 1015 1016
           1017 1018 1019 1010
           1021 1022 1023 1024
           1025 1026 1027 1028
           1029 1020 1021 1022
           1033 1034 1035 1036
           1037 1038 1039 1030
           1041 1042 1043 1044
           1045 1046 1047 1048
           1049 1040 1041 1042
           1053 1054 1055 1056
           1057 1058 1059 1050
           1061 1062 1063 1064
           1065 1066 1067 1068
           1069 1060 1061 1062
           1073 1074 1075 1076
           1077 1078 1079 1070
           1081 1082 1083 1084
           1085 1086 1087 1088
           1089 1080 1081 1082
           1093 1094 1095 1096
           1097 1098 1099 1090
           1101 1102 1103 1104
           1105 1106 1107 1108
           1109 1100 1101 1102
           1113 1114 1115 1116
           1117 1118 1119 1110
           1121 1122 1123 1124
           1125 1126 1127 1128
           1129 1120 1121 1122
           1133 1134 1135 1136
           1137 1138 1139 1130
           1141 1142 1143 1144
           1145 1146 1147 1148
           1149 1140 1141 1142
           1153 1154 1155 1156
           1157 1158 1159 1150
           1161 1162 1163 1164
           1165 1166 1167 1168
           1169 1160 1161 1162
           1173 1174 1175 1176
           1177 1178 1179 1170
           1181 1182 1183 1184
           1185 1186 1187 1188
           1189 1180 1181 1182
           1193 1194 1195 1196
           1197 1198 1199 1190
           1201 1202 1203 1204
           1205 1206 1207 1208
           1209 1200 1201 1202
           1213 1214 1215 1216
           1217 1218 1219 1210
           1221 1222 1223 1224
           1225 1226 1227 1228
           1229 1220 1221 1222
           1233 1234 1235 1236
           1237 1238 1239 1230
           1241 1242 1243 1244
           1245 1246 1247 1248
           1249 1240 1241 1242
           1253 1254 1255 1256
           1257 1258 1259 1250
           1261 1262 1263 1264
           1265 1266 1267 1268
           1269 1260 1261 1262
           1273 1274 1275 1276
           1277 1278 1279 1270
           1281 1282 1283 1284
           1285 1286 1287 1288
           1289 1280 1281 1282
           1293 1294 1295 1296
           1297 1298 1299 1290
           1301 1302 1303 1304
           1305 1306 1307 1308
           1309 1300 1301 1302
           1313 1314 1315 1316
           1317 1318 1319 1310
           1321 1322 1323 1324
           1325 1326 1327 1328
           1329 1320 1321 1322
           1333 1334 1335 1336
           1337 1338 1339 1330
           1341 1342 1343 1344
           1345 1346 1347 1348
           1349 1340 1341 1342
           1353 1354 1355 1356
           1357 1358 1359 1350
           1361 1362 1363 1364
           1365 1366 1367 1368
           1369 1360 1361 1362
           1373 1374 1375 1376
           1377 1378 1379 1370
           1381 1382 1383 1384
           1385 1386 1387 1388
           1389 1380 1381 1382
           1393 1394 1395 1396
           1397 1398 1399 1390
           1401 1402 1403 1404
           1405 1406 1407 1408
           1409 1400 1401 1402
           1413 1414 1415 1416
           1417 1418 1419 1410
           1421 1422 1423 1424
           1425 1426 1427 1428
           1429 1420 1421 1422
           1433 1434 1435 1436
           1437 1438 1439 1430
           1441 1442 1443 1444
           1445 1446 1447 1448
           1449 1440 1441 1442
           1453 1454 1455 1456
           1457 1458 1459 1450
           1461 1462 1463 1464
           1465 1466 1467 1468
           1469 1460 1461 1462
           1473 1474 1475 1476
           1477 1478 1479 1470
           1481 1482 1483 1484
           1485 1486 1487 1488
           1489 1480 1481 1482
           1493 1494 1495 1496
           1497 1498 1499 1490
           1501 1502 1503 1504
           1505 1506 1507 1508
           1509 1500 1501 1502
           1513 1514 1515 1516
           1517 1518 1519 1510
           1521 1522 1523 1524
           1525 1526 1527 1528
           1529 1520 1521 1522
           1533 1534 1535 1536
           1537 1538 1539 1530
           1541 1542 1543 1544
           1545 1546 1547 1548
           1549 1540 1541 1542
           1553 1554 1555 1556
           1557 1558 1559 1550
           1561 1562 1563 1564
           1565 1566 1567 1568
           1569 1560 1561 1562
           1573 1574 1575 1576
           1577 1578 1579 1570
           1581 1582 1583 1584
           1585 1586 1587 1588
           1589 1580 1581 1582
           1593 1594 1595 1596
           1597 1598 1599 1590
           1601 1602 1603 1604
           1605 1606 1607 1608
           1609 1600 1601 1602
           1613 1614 1615 1616
           1617 1618 1619 1610
           1621 1622 1623 1624
           1625 1626 1627 1628
           1629 1620 1621 1622
           1633 1634 1635 1636
           1637 1638 1639 1630
           1641 1642 1643 1644
           1645 1646 1647 1648
           1649 1640 1641 1642
           1653 1654 1655 1656
           1657 1658 1659 1650
           1661 1662 1663 1664
           1665 1666 1667 1668
           1669 1660 1661 1662
           1673 1674 1675 1676
           1677 1678 1679 1670
           1681 1682 1683 1684
           1685 1686 1687 1688
           1689 1680 1681 1682
           1693 1694 1695 1696
           1697 1698 1699 1690
           1701 1702 1703 1704
           1705 1706 1707 1708
           1709 1700 1701 1702
           1713 1714 1715 1716
           1717 1718 1719 1710
           1721 1722 1723 1724
           1725 1726 1727 1728
           1729 1720 1721 1722
           1733 1734 1735 1736
           1737 1738 1739 1730
           1741 1742 1743 1744
           1745 1746 1747 1748
           1749 1740 1741 1742
           1753 1754 1755 1756
           1757 1758 1759 1750
           1761 1762 1763 1764
           1765 1766 1767 1768
           1769 1760 1761 1762
           1773 1774 1775 1776
           1777 1778 1779 1770
           1781 1782 1783 1784
           1785 1786 1787 1788
           1789 1780 1781 1782
           1793 1794 1795 1796
           1797 1798 1799 1790
           1801 1802 1803 1804
           1805 1806 1807 1808
           1809 1800 1801 1802
           1813 1814 1815 1816
           1817 1818 1819 1810
           1821 1822 1823 1824
           1825 1826 1827 1828
           1829 1820 1821 1822
           1833 1834 1835 1836
           1837 1838 1839 1830
           1841 1842 1843 1844
           1845 1846 1847 1848
           1849 1840 1841 1842
           1853 1854 1855 1856
           1857 1858 1859 1850
           1861 1862 1863 1864
           1865 1866 1867 1868
           1869 1860 1861 1862
           1873 1874 1875 1876
           1877 1878 1879 1870
           1881 1882 1883 1884
           1885 1886 1887 1888
           1889 1880 1881 1882
           1893 1894 1895 1896
           1897 1898 1899 1890
           1901 1902 1903 1904
           1905 1906 1907 1908
           1909 1900 1901 1902
           1913 1914 1915 1916
           1917 1918 1919 1910
           1921 1922 1923 1924
           1925 1926 1927 1928
           1929 1920 1921 1922
           1933 1934 1935 1936
           1937 1938 1939 1930
           1941 1942 1943 1944
           1945 1946 1947 1948
           1949 1940 1941 1942
           1953 1954 1955 1956
           1957 1958 1959 1950
           1961 1962 1963 1964
           1965 1966 1967 1968
           1969 1960 1961 1962
           1973 1974 1975 1976
           1977 1978 1979 1970
           1981 1982 1983 1984
           1985 1986 1987 1988
           1989 1980 1981 1982
           1993 1994 1995 1996
           1997 1998 1999 1990
           2001 2002 2003 2004
           2005 2006 2007 2008
           2009 2000 2001 2002
           2013 2014 2015 2016
           2017 2018 2019 2010
           2021 2022 2023 2024
           2025 2026 2027 2028
           2029 2020 2021 2022
           2033 2034 2035 2036
           2037 2038 2039 2030
           2041 2042 2043 2044
           2045 2046 2047 2048
           2049 2040 2041 2042
           2053 2054 2055 2056
           2057 2058 2059 2050
           2061 2062 2063 2064
           2065 2066 2067 2068
           2069 2060 2061 2062
           2073 2074 2075 2076
           2077 2078 2079 2070
           2081 2082 2083 2084
           2085 2086 2087 2088
           2089 2080 2081 2082
           2093 2094 2095 2096
           2097 2098 2099 2090
           2101 2102 2103 2104
           2105 2106 2107 2108
           2109 2100 2101 2102
           2113 2114 2115 2116
           2117 2118 2119 2110
           2121 2122 2123 2124
           2125 2126 2127 2128
           2129 2120 2121 2122
           2133 2134 2135 2136
           2137 2138 2139 2130
           2141 2142 2143 2144
           2145 2146 2147 2148
           2149 2140 2141 2142
           2153 2154 2155 2156
           2157 2158 2159 2150
           2161 2162 2163 2164
           2165 2166 2167 2168
           2169 2160 2161 2162
           2173 2174 2175 2176
           2177 2178 2179 2170
           2181 2182 2183 2184
           2185 2186 2187 2188
           2189 2180 2181 2182
           2193 2194 2195 2196
           2197 2198 2199 2190
           2201 2202 2203 2204
           2205 2206 2207 2208
           2209 2200 2201 2202
           2213 2214 2215 2216
           2217 2218 2219 2210
           2221 2222 2223 2224
           2225 2226 2227 2228
           2229 2220 2221 2222
           2233 2234 2235 2236
           2237 2238 2239 2230
           2241 2242 2243 2244
           2245 2246 2247 2248
           2249 2240 2241 2242
           2253 2254 2255 2256
           2257 2258 2259 2250
           2261 2262 2263 2264
           2265 2266 2267 2268
           2269 2260 2261 2262
           2273 2274 2275 2276
           2277 2278 2279 2270
           2281 2282 2283 2284
           2285 2286 2287 2288
           2289 2280 2281 2282
           2293 2294 2295 2296
           2297 2298 2299 2290
           2301 2302 2303 2304
           2305 2306 2307 2308
           2309 2300 2301 2302
           2313 2314 2315 2316
           2317 2318 2319 2310
           2321 2322 2323 2324
           2325 2326 2327 2328
           2329 2320 2321 2322
           2333 2334 2335 2336
           2337 2338 2339 2330
           2341 2342 2343 2344
           2345 2346 2347 2348
           2349 2340 2341 2342
           2353 2354 2355 2356
           2357 2358 2359 2350
           2361 2362 2363 2364
           2365 2366 2
```

```

    !(e = NAG_ALLOC(e_len, double)) ||
    !(taup = NAG_ALLOC(taup_len, Complex)) ||
    !(tauq = NAG_ALLOC(tauq_len, Complex)) )
{
    Vprintf("Allocation failure\n");
    exit_status = -1;
    goto END;
}

/* Read A from data file */
for (i = 1; i <= m; ++i)
{
    for (j = 1; j <= n; ++j)
        Vscanf(" ( %lf , %lf )", &A(i,j).re, &A(i,j).im);
}
Vscanf("%*[^\n] ");

/* Reduce A to bidiagonal form */
f08ksc(order, m, n, a, pda, d, e, tauq, taup, &fail);
if (fail.code != NE_NOERROR)
{
    Vprintf("Error from f08ksc.\n%s\n", fail.message);
    exit_status = 1;
    goto END;
}

/* Print bidiagonal form */
Vprintf("\nDiagonal\n");
for (i = 1; i <= MIN(m,n); ++i)
    Vprintf("%9.4f%s", d[i-1], i%8==0 ?"\n":" ");
if (m >= n)
    Vprintf("\nSuper-diagonal\n");
else
    Vprintf("\nSub-diagonal\n");
for (i = 1; i <= MIN(m,n) - 1; ++i)
    Vprintf("%9.4f%s", e[i-1], i%8==0 ?"\n":" ");
Vprintf("\n");

END:
    if (a) NAG_FREE(a);
    if (d) NAG_FREE(d);
    if (e) NAG_FREE(e);
    if (taup) NAG_FREE(taup);
    if (tauq) NAG_FREE(tauq);

    return exit_status;
}

```

9.2 Program Data

```

f08ksc Example Program Data
6 4
( 0.96,-0.81) (-0.03, 0.96) (-0.91, 2.06) (-0.05, 0.41)
(-0.98, 1.98) (-1.20, 0.19) (-0.66, 0.42) (-0.81, 0.56)
( 0.62,-0.46) ( 1.01, 0.02) ( 0.63,-0.17) (-1.11, 0.60)
(-0.37, 0.38) ( 0.19,-0.54) (-0.98,-0.36) ( 0.22,-0.20)
( 0.83, 0.51) ( 0.20, 0.01) (-0.17,-0.46) ( 1.47, 1.59)
( 1.08,-0.28) ( 0.20,-0.12) (-0.07, 1.23) ( 0.26, 0.26) :Values of M and N
( 0.96,-0.81) (-0.03, 0.96) (-0.91, 2.06) (-0.05, 0.41)
(-0.98, 1.98) (-1.20, 0.19) (-0.66, 0.42) (-0.81, 0.56)
( 0.62,-0.46) ( 1.01, 0.02) ( 0.63,-0.17) (-1.11, 0.60)
(-0.37, 0.38) ( 0.19,-0.54) (-0.98,-0.36) ( 0.22,-0.20)
( 0.83, 0.51) ( 0.20, 0.01) (-0.17,-0.46) ( 1.47, 1.59)
( 1.08,-0.28) ( 0.20,-0.12) (-0.07, 1.23) ( 0.26, 0.26) :End of matrix A

```

9.3 Program Results

```

f08ksc Example Program Results

Diagonal
-3.0870    2.0660    1.8731    2.0022
Super-diagonal
  2.1126    1.2628   -1.6126

```