Jacobi's Method for Eigenvalues: an Illustration of J

by Donald McIntyre

Introduction

J is Iverson's powerful new dialect of APL available for a wide range of computers [1, 2]. I have already given examples of J, explaining some of its interesting syntax with reference to applications ranging from elementary arithmetic to system programming and the construction of a database [3-5]. The conciseness of the notation enables many quite complicated operations to be performed by entering a few short lines, in the manner of Iverson's Direct Definition in older APL [6, 7]. In this paper I take an example of a problem that must be solved iteratively. It requires a program of several lines with conditional branching.

Eigenvalues and Eigenvectors

The inner product of a matrix and a vector is another vector, which in general will differ in length and orientation. If there is no change in orientation, the vector is called an *eigenvector* of the matrix. In this case the ratio of the length of the new vector to the length of the original one is an *eigenvalue* (originally called a *latent* or *characteristic root*) of the matrix. Eigenvalues can be complex numbers, but the eigenvalues of a symmetric matrix of real numbers are always real. Jacobi's method is used to compute the eigenvectors and eigenvalues of such matrices.

The number of linearly independent rows or columns of a matrix (for these are equal) is called the *rank* of the matrix. This is not to be confused with the use of *rank* in other contexts, such as denoting the shape of the shape of an array. The rank of the inner product of two matrices cannot exceed the rank of either matrix from which it is produced. In particular, when a matrix is multiplied by its own transpose the product has the same rank as the original matrix. Because the product of a matrix with its own transpose is necessarily symmetric, Jacobi's method can be used. The rank of a matrix equals the number of non-zero eigenvalues, consequently the Jacobi solution measures the amount of independent information in the data.

When you hold up your open hand your fingers and thumb approximate to 5 radiating vectors in 3-dimensional space. Their orientation could be described by a matrix of 5 rows (items) and 3 columns. Multiply the matrix by its transpose to get a 3 by 3 product. It has 3 eigenvalues, and one of them is close to zero because the vectors lie nearly in one plane. If the matrix represents five students each taking 3 tests, then we know that the tests do not provide independent information.

Jacobi's method can be used with geometrical data (as in structural geology) or data represented in an imaginary multi-dimensional space (as in educational tests or chemical analyses).

Jacobi's Method

Vectors lying in a plane are rotated through the angle *y* by multiplying them by the 2 by 2 rotation matrix:

cos y	-sin y
sin y	cos y

Rotations in multi-dimensional space are performed by a modified identity matrix in which the elements in the cells (i,i), (i,j), (j,i), and (j,j) correspond to those in the 2 by 2 rotation matrix.

The method is based on the fact that eigenvalues remain unchanged on rotation. A sequence of rotations is performed, each time chosen so as to reduce to zero the largest absolute off-diagonal value (the *pivot*). The result is a matrix with eigenvalues on the diagonal and zeros in all other positions. When the same sequence of rotations is performed on an identity matrix, the columns of the final result are the eigenvectors of the original matrix.

Procedure

To write the Jacobi algorithm in J we must first define some basic operations. We must be able to *amend* a matrix, based on the position and value of the pivot. Using a matrix of random numbers as a partial example (even although it is not symmetric), place the numbers 996 997 998 999 in the required positions:

```
   setrl=. 9!:1                    NB. Set Random Link
   setrl 7^5
   ]y=. 50-~ ?6 6$100
_37  25  _5   3 _29 _46
 17  17  43 _12   1  33
_47 _45   2  17 _50 _12
_44  _9  18   8  43  34
  2 _41  15  _9  20  41
 26 _24 _46  23 _18  13
```

Define the following verbs:

```
   ut=. ,@(</~@i.)@#               NB. Upper Triangle
   pt=. (, i. >./@(ut # ,))@|      NB. Pivot in Triangle
   pm=. <.@(pt % #) , # | pt       NB. Pivot in Matrix
   pv=. {~ <@pm                    NB. Pivot
   pa=. 0 0&{ ; ] ; |. ; 1 1&{     NB. Permutations for Amend
   ia=. pa@pm { i.@$               NB. Indices for Amend
```

The value of the pivot (in the upper triangle) is:

```
   pv y
_50
```

With respect to the ravel of y, the linear indexes of cells to be amended are:

```
   ia y
14 16 26 28
```

Identification of the cells that are to be amended is a function of y

```
   amend=. ia@])
```

Note that a noun (produced by applying `ia` to y) and an adverb (`)`) give a noun.

```
   996 997 998 999 amend y
_37  25  _5   3 _29 _46
 17  17  43 _12   1  33
_47 _45 996  17 997 _12
_44  _9  18   8  43  34
  2 _41 998  _9 999  41
 26 _24 _46  23 _18  13
```

Define the following verbs:

```
ip=. +/ .*                              NB. inner product
readkb=. ".@(1!:1@1:)                   NB. read from keyboard
write=. 1!:2&2                          NB. write to screen
diag=. (<0 1)&|:                        NB. matrix diagonal
uf=. -:@-/@(pm{diag)                    NB. u function
rss=. %:@+/@*:                          NB. sq rt of sum of sqs
vf=. rss@(pv,uf)                        NB. v function
sign=. >:&0 - <&0                       NB. sign of the sine
cosf=. %:@((vf + |@uf) % +:@vf)             NB. cosine function
sinf=. sign@uf * -@pv % +:@(vf * cosf)  NB. sine function
```

The Jacobi Program

Now define and display the dyadic function `jacobi` whose left argument is the tolerance allowed for convergence. The statements are boxed. Names ending with a right parenthesis are labels (a feature not implemented in February 1991 when the program was originally written); and `$.` is the `suite`, i.e. the vector controlling the sequence of execution [2].

The maximum number of iterations is set as a global variable. Should this limit be reached, an opportunity is given to permit further iterations.

```
maxit=. 20
s0=. <'it=. 0 [ I=. Q=. =/~ i.#R=. y.'
s1=. <'loop) $.=. >(end;$.){~ x.<|p=. -pv R'
s2=. <'v=. %: +/*~p,u=. -: -/ (pm R){ diag R'
s3=. <'sin=. (sign u)*p%+:v* cos=. %: (v+|u)%+:v'
s4=. <'$.=. >(end;$.){~ *./0~:-.-|sin,cos'
s5=. <'r=. ((cos,-sin),sin,cos) (ia R)} I'
s6=. <'Q=. Q ip |:r [ R=. r ip R ip |:r'
s7=. <'$.=. >(loop;$.){~ maxit<it=. it+1'
s8=. <'write ''Number of further iterations or 0?'''
s9=. <'$.=. >(loop;end){~ maxit<it=. it-readkb 1'
s10=. <'end) R,:Q'
jacobi=. '' : (s0,s1,s2,s3,s4,s5,s6,s7,s8,s9,s10)
```

```
jacobi
```

```
:
  it=. 0 [ I=. Q=. =/~ i.#R=. y.
  loop) $.=. >(end;$.){~ x.<|p=. -pv R
  v=. %: +/*~p,u=. -: -/ (pm R){ diag R
  sin=. (sign u)*p%+:v* cos=. %: (v+|u)%+:v
  $.=. >(end;$.){~ *./0~:-.-|sin,cos
  r=. ((cos,-sin),sin,cos) (ia R)} I
  Q=. Q ip |:r [ R=. r ip R ip |:r
  $.=. >(loop;$.){~ maxit<it=. it+1
  write 'Number of further iterations or 0?'
  $.=. >(loop;end){~ maxit<it=. it-readkb 1
  end) R,:Q
```

Notice how the line which defines r amends by placing the four rotation values in the matrix. The line defining Q shows how 3 inner products give new values for Q and R. R begins as the original matrix and ends as the matrix with the eigenvalues on its diagonal; Q begins as the identity matrix and ends as the matrix whose columns are the eigenvectors.

Using Jacobi

It will be convenient to have the following utilities:

```
clean=. ] * (<:|)              NB. Sets small values to 0
rfd=. %&180@o.                 NB. Radians From Degrees
dfr=. rfd^:_1                  NB. Degrees From Radians
arcos=. dfr@(_2&o.)            NB. Arcosine in degrees
round=. [ * <.@(0.5&+@(%~))    NB. Tolerance is on left
smatrix=. |: ip ]              NB. symmetric mat:  as a Fork
smatrix=. ip~ |:               NB. symmetric mat:  as a Hook
```

Test data from a seismological study gave the following Direction Cosines for five coplanar vectors [8, Table 1]. Direction Cosines are the cosines of the angles between a vector and each of three Cartesian axes; they are identical to the coordinates of a unit vector with the same direction.

```
   n=. -p=. 0.7071
   ]table1=. (p,n,0),(n,p,0),(0,p,n),(0,n,p),:n,0,p
 0.7071 _0.7071       0
_0.7071  0.7071       0
      0  0.7071 _0.7071
      0 _0.7071  0.7071
_0.7071       0  0.7071

   ]m=.0.0001 round smatrix table1
 1.5 _1 _0.5
  _1  2   _1
_0.5 _1  1.5

   ]z=. 1e_5 clean 1e_6 jacobi  m
      0         0         0
      0         3         0
      0         0         2
0.57735 _0.408248 _0.707107
0.57735  0.816497         0
0.57735 _0.408248  0.707107
```

The eigenvalues are the diagonal elements of the first item; the eigenvectors are the corresponding column vectors of the second item.

We now verify that the results are indeed eigenvectors and eigenvalues. The product of the matrix and an eigenvector equals the product of this vector and the corresponding eigenvalue. Because the computations were carried out with limited tolerance, we cannot use `match` (-:).

```
   m  ip  0{"1 (1{z)
1.66533e_15 _2.9976e_15 1.44329e_15
   (0{0{0{z) * 0{"1 (1{z)
0 0 0

   m  ip  1{"1 (1{z)
_1.22474 2.44949 _1.22474
   (1{1{0{z) * 1{"1 (1{z)
_1.22474 2.44949 _1.22474

   m  ip  2{"1 (1{z)
_1.41421 _9.8829e_10 1.41421
   (2{2{0{z) * 2{"1 (1{z)
_1.41421 0 1.41421
```

These comparisons can be made with a single expression:

```
   (1e_9 clean |:m ip 1{z);(diag 0{z) *"0 1 |: 1{z
```

0	0	0	0	0	0
_1.22474	2.44949	_1.22474	_1.22474	2.44949	_1.22474
_1.41421	0	1.41421	_1.41421	0	1.41421

The eigenvectors of a symmetric matrix should be orthogonal. We show that they are:

```
   1e_9 clean smatrix 1{z
1 0 0
0 1 0
0 0 1
```

It is convenient to format a sorted summary of eigenvalues and eigenvectors. Being familiar with APL's Direct Definition, but as a beginner in J ,I found it easiest to write the verb in explicit definition (the place-holder for the argument is `y.`):

```
evf=. '|: (\:d){"1 (d=. diag 0{y.),1{y.' : ''
```

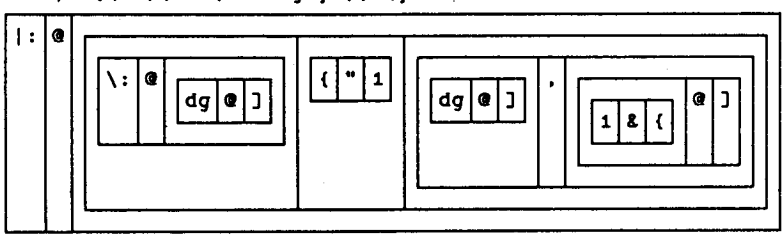
In this context recognition of the parallelism of `0{y.` and `1{y.` has an advantage.

```
   ]ez=. evf z
3 _0.408248 0.816497 _0.408248
2 _0.707107        0  0.707107
0   0.57735  0.57735   0.57735
```

Each item (row) is an eigenvalue followed by the corresponding eigenvector, and the table is sorted by decreasing size of eigenvalue. One eigenvalue is zero, which confirms that the vectors are indeed coplanar. The corresponding eigenvector is the normal to this plane.

We can use this explicit definition to illustrate a way of converting from explicit to tacit definition. First abbreviate by introducing the verb `dg` for the eigenvalues (i.e. the diagonal of the first item). Then apply the adverb `:11` as described in [9].

```
dg=. diag@{.
'|: (\:d){"1 (d=. dg y.),1{y.':11
```



Using this display as a guide, and remembering that conjunctions grab whatever is to their right, we can write:

```
evf=. |:@(\:@dg {"1 dg,1&{)
```

Note that the rank of `1{z` is 2 whereas the rank of `}.z` is 3.

```
   ez-: evf z
1
```

If we run again with `maxit=. 6` convergence will not be completed to tolerance 1e_5. Enter 0 to see the answer at this stage, or 2 to do one more iteration — which will be successful.

The null axes of earthquakes in the Marianas Islands provide a real example [8, Table 2]. The direction cosines are measured from North, East, and Down respectively:

```
t=. _0.75 0.27 0.6, _0.85 0.53 0, _0.63 0.46 0.63,: _0.85 0 0.53
t=.t, _0.09 1 0, _0.17 0.98 0, _0.86 0 0.52,: _0.82 _0.28 0.52
t=.t,0.59 0.79 0.12, _0.59 _0.81 .07, 0.87 0.4 0.28,: .02 .87 0.5
t=.t,0.67 0.7 0.26, _0.66 0.64 0.39, 0.38 .85 .36,:_0.02 .99 0.12
t=.t,0.35 0.44 0.83, _0.88 _0.47 .09, .82 .44 .36,:0.67 0.64 0.37

   y=. t, _0.26 0.65 0.73

   m=. smatrix y
   z=. 1e_5 clean 1e_6 jacobi m
```

The eigenvectors, in descending order of eigenvalues, are:

```
    evf z
10.8801 0.501099  0.827536  0.253148
8.32209 0.815533 _0.353717 _0.458028
1.84991 0.289492 _0.435968   0.85213
```

That the results are indeed eigenvectors and eigenvalues is shown by:

```
      (1e_9 clean |:m ip 1{z);(diag 0{z) *"0 1 |: 1{z
```

6.78694	_2.94367	_3.81175	6.78694	_2.94367	_3.81175
5.45201	9.00367	2.75428	5.45201	9.00367	2.75428
0.535535	_0.806502	1.57637	0.535535	_0.806503	1.57637

That the eigenvectors are mutually perpendicular is shown by:

```
   1e_5 clean arcos smatrix 1{z
 0 90 90
90  0 90
90 90  0
```

Fara and Scheidegger said that "the average square of the variance of the sine of the scattering" is 0.088, and hence the "rms-sine scattering angle" is 18. We compute these values as follows:

```
   ] s2=. (<./ diag 0{z) % #y
0.088091
```

The corresponding angle:

```
   dfr _1 o. %: s2
17.2656
```

Acknowledgements

After I lectured on J at St. Andrews University (1 February 1991), Simon Kenyon, a geology student, asked me for help with his study of the relationship between the orientation and shape of pebbles in a sedimentary rock [10]. His need to compute eigenvalues and eigenvectors suggested the problem.

References

[1] J is available from Iverson Software Inc., 33 Major Street, Toronto, Ontario, Canada M5S 2K9. Phone (416) 925-6096; Fax (416) 488-7559. Also from I-APL Ltd. in the UK (see Product Guide). This paper was submitted in November 1991. The examples, originally executed with J Version 3.3 have been revised for compatibility with Version 5 (27 June 1992).

[2] Kenneth E. Iverson, *ISI Dictionary of J*, Version 5, Iverson Software Inc., Toronto (1992)

[3] Donald B. McIntyre, *Hooks and Forks and the Teaching of Elementary Arithmetic*, Vector, Vol.8, No.3 (January 1992) 101-123.

[4] Donald B. McIntyre, *Using J with External Data: two examples*, Vector, Vol.8, No.4 (April 1992) 97-110.

[5] Donald B. McIntyre, *Using J's Boxed Arrays*, Vector, Vol.9, No.1 (1992) 92-105.

[6] Kenneth E. Iverson, *Elementary Analysis*, APL Press, Swarthmore, Pennsylvania (1976) 219pp.

[7] Donald B. McIntyre, *Experience with Direct Definition One-liners in Writing APL Applications*, Conference Proceedings, APL Users Meeting, Toronto, September 1978. Sponsored by I.P. Sharp Associates, Toronto (1978) p.281-297.

[8] H.D. Fara & A.E. Scheidegger, *An eigenvalue method for the statistical evaluation of fault plane solutions of earthquakes*, Seismological Society of America, Bull. Vol. 53 (1963) p.811-816.

[9] Roger K.W. Hui, Kenneth E. Iverson, & Eugene E. McDonnell, *Tacit Definition*, APL91 Conference Proceedings, Stanford, California, August 1991. APL Quote Quad Vol.21 Number 4 (August 1991), p.202-211.

[10] L.D. Drake, *Till Fabric Control by Clast Shape*, Geological Society of America, Bull. Vol. 85 (1974) p.247-250