

```

1: *****
2: /part3.f90
3:
4:
5: PROGRAM part3
6:
7:   USE part3_mod
8:
9:   IMPLICIT NONE
10:
11:   INTEGER, PARAMETER :: m=3,n=3
12:   REAL, DIMENSION(m,n) :: mat1
13:   REAL, DIMENSION(m) :: eigvec
14:   REAL :: tol=0.000001,eigval
15:   LOGICAL, DIMENSION(2) :: flag
16:   INTEGER :: iters
17:
18:   mat1=getmat(m,n) !** Input matrix1 from the keyboard
19:
20:   !** Explicitly assign matrix to mat1 for testing
21:   mat1(1,:)=(/1.,5./)
22:   mat1(2,:)=(/6.,3./)
23:   mat1(3,:)=(/2.,8./)
24:   eigvec=(/1.,1./)
25:
26:   !** Call the power method
27:   iters=power1(mat1,eigvec,tol,eigval,flag)
28:
29:   PRINT*,"Estimate of eigenvalue : ",eigval
30:   PRINT*,"Estimate of eigenvector : ",3f11.6',eigvec(1),eigvec(2),eigvec(3)
31:
32:   IF (flag(1)) THEN
33:     PRINT*,"The method converged to the solution within the given tolerance."
34:   ELSE
35:     PRINT*,"The method did not to the solution within the given tolerance."
36:   ENDIF
37:
38:
39:   IF (flag(2)) THEN
40:     PRINT*,"The maximum number of iterations was exceeded."
41:   ELSE
42:     PRINT*,"The maximum number of iterations was not exceeded."
43:   ENDIF
44:
45: END PROGRAM part3
46:
47: !***** OUTPUT FROM THE CODE WHEN RUN IS *****
48: ! Estimate of eigenvalue : 13.06310
49: ! Estimate of eigenvector : 0.603874 0.856918 1.000000
50: ! The method converged to the solution within the given tolerance.
51: ! The maximum number of iterations was not exceeded.
52: ! Number of iterations taken : 12
53: *****
54:
55:
56: *****
57: /part3_mod.f90
58:
59: MODULE part3_mod
60:
61: IMPLICIT NONE
62:
63: CONTAINS
64:
65: ! *****
66:
67: FUNCTION infnorm(vec)

```

```

68: !** Calculates the infinity norm of vector vec.
69: !** Dummy declarations
70: REAL, DIMENSION(:), INTENT(IN) :: vec
71: !** Local declarations
72: REAL :: infnorm
73:
74: infnorm=MAXVAL(ABS(vec))
75:
76: END FUNCTION infnorm
77:
78: ! *****
79:
80: FUNCTION twonorm(vec)
81: !** Calculates the Euclidean norm of vector vec.
82: !** Dummy declarations
83: REAL, DIMENSION(:), INTENT(IN) :: vec
84: !** Local declarations
85: REAL :: twonorm
86:
87: twonorm=SQRT(SUM(vec**2))
88:
89: END FUNCTION twonorm
90:
91: ! *****
92:
93: FUNCTION cont(y,x,tol,max_iters)
94: !** Returns a logical type. If .TRUE. then the tolerance has been met or
95: !** the maximum number of iterations has been exceeded.
96:
97: !** Dummy variables
98: REAL, DIMENSION(:), INTENT(IN) :: y,x
99: REAL, INTENT(IN) :: tol
100: INTEGER, INTENT(IN) :: max_iters
101: !** Local declarations
102: LOGICAL, DIMENSION(2) :: cont
103: INTEGER, SAVE :: iter=0
104:
105: cont(1)=(infnorm(ABS(y-x)) < tol)
106: cont(2)=(iter>max_iters)
107: iter=iter+1
108:
109: END FUNCTION cont
110:
111: ! *****
112:
113: FUNCTION power1(mat,x,tol,eigv,flag)
114: !** Func. to calc. the dominant eigenvalue and corresponding normalised
115: !** eigenvector of the (n) by (n) matrix [mat] the initial guess is the
116: !** input vector (x) and the method is considered to have converged if
117: !** absolute error is less than the given tolerance (tol). The
118: !** eigenvalue is returned in (eigv) and the eigenvector is
119: !** returned in x
120:
121: !** Dummy arguments
122: REAL, DIMENSION(:), INTENT(IN) :: mat
123: REAL, DIMENSION(:), INTENT(INOUT) :: x
124: REAL, INTENT(IN) :: tol
125: REAL, INTENT(OUT) :: eigv
126: LOGICAL, DIMENSION(:), INTENT(OUT) :: flag
127:
128: !** Local Declarations
129: REAL, DIMENSION(SIZE(x)) :: y
130: INTEGER :: loc,power1
131: INTEGER, PARAMETER :: max_iters=10000
132: power1=0
133: flag=.FALSE.
134: !** Normalise initial estimate

```

```

135: x=x/lnfnorm(x)
136:
137: DO
138: IF (flag(1).OR. flag(2))
139: y=mulmatvec(mat,x)
140: PRINT*,y
141:
142: loc=MAXLOC(ABS(y),1)
143: eigv=y(loc)/x(loc)
144: PRINT*,y,loc,eigv
145: y/y/lnfnorm(y)
146: flag=cont(y,x,col,max_iters)
147: x=y
148: power1=power1+1
149: ENDDO
150:
151: END FUNCTION power1
152:
153: ! *****
154:
155: FUNCTION getmat(m,n)
156: !**** Function to input a matrix from the keyboard. The number of rows
157: !**** (m) and the number of columns (n) are input arguments to the
158: !**** function
159:
160: INTEGER, INTENT(IN) :: m,n !**** Dummy declaration
161: REAL, DIMENSION(m,n) :: getmat !**** Local Declaration
162:
163: INTEGER :: i
164:
165: DO i=1,m
166: PRINT '( "Enter matrix row :",i2)',i !*** Prompt for row number
167: READ*,getmat(i,:) !*** Read in row
168: ENDDO
169:
170: END FUNCTION getmat
171:
172: ! *****
173:
174: SUBROUTINE outmat(mat)
175: !**** Subroutine to output a matrix to the screen.
176:
177: REAL, DIMENSION(:,), INTENT(IN) :: mat !*** Dummy declaration
178:
179: INTEGER :: i
180:
181: DO i=1,SIZE(mat,1)
182: PRINT*,mat(i,:)
183: ENDDO
184:
185: END SUBROUTINE outmat
186:
187: ! *****
188:
189: FUNCTION mulmat(mat1,mat2)
190: !**** Function to input two matrices [mat1] & [mat2] and check if
191: !**** [mat1]*[mat2] is a valid matrix multiplication. If it is valid the
192: !**** matrix product [mat1]*[mat2] is returned.
193:
194: REAL, DIMENSION(:,), INTENT(IN) :: mat1
195: REAL, DIMENSION(:,), INTENT(IN) :: mat2
196:
197: INTEGER :: m,n,k,i,j,p
198:
199: REAL, DIMENSION(SIZE(mat1,1),SIZE(mat2,2)) :: mulmat
200:
201: m=SIZE(mat1,1) ; n=SIZE(mat1,2) ; k=SIZE(mat2,2)

```

```

202:
203: !**** Perform the matrix multiplication
204: !**** using three DO loops
205:
206: IF (SIZE(mat2,1) == n) THEN
207: DO i=1,m !*** For each row of getmat (mat3)
208: DO j=1,k !*** For each column of getmat (mat3)
209: mulmat(i,j)=0 !*** initialise to zero
210: DO p=1,n
211: mulmat(i,j)=mulmat(i,j)+mat1(i,p)*mat2(p,j)
212: ENDDO
213: ENDDO
214: ENDDO
215: ELSE
216: PRINT*, "Size mismatch in mulmat"
217: ENDIF
218:
219: END FUNCTION mulmat
220:
221: ! *****
222:
223: FUNCTION mulmatvec(mat,vec)
224: !**** Function to input a matrix [mat] and a vector [v] check if matrix
225: !**** vector multiplication is valid w.r.t. their sizes. If it is then
226: !**** this function returns the matrix vector product.
227:
228: REAL, DIMENSION(:,), INTENT(IN) :: mat
229: REAL, DIMENSION(:), INTENT(IN) :: vec
230:
231: INTEGER :: m,n,k,i,j
232: REAL, DIMENSION(SIZE(mat,1)) :: mulmatvec
233:
234: m=SIZE(mat,1) ; n=SIZE(mat,2) ; k=SIZE(mat,1)
235:
236: !**** Perform the matrix multiplication using three DO loops
237: IF (n=k) THEN
238: DO i=1,m
239: mulmatvec(i)=0
240: DO j=1,k
241: mulmatvec(i)=mulmatvec(i)+mat(i,j)*vec(j)
242: ENDDO
243: ENDDO
244: ELSE
245: PRINT*, "Size mismatch in mulmatvec!"
246: ENDIF
247:
248: END FUNCTION mulmatvec
249:
250: ! *****
251:
252: FUNCTION transmat(mat)
253:
254: !**** Dummy arguments
255: REAL, DIMENSION(:,), INTENT(IN) :: mat
256: INTEGER :: m,n,i,j
257: REAL, DIMENSION(SIZE(mat,2),SIZE(mat,1)) :: transmat !*** Return
258:
259: !**** Findout the no. of rows and cols in the matrix
260: m=SIZE(mat,1) ; n=SIZE(mat,2)
261:
262: !**** Perform the transpose using two DO loops
263: DO i=1,n !*** Loop over rows in mat
264: DO j=1,m !*** Loop over cols in mat
265: transmat(i,j)=mat(j,i)
266: ENDDO
267: ENDDO
268:

```

solutions.txt Mon Nov 28 08:31:33 2011 5

```
269: END FUNCTION Transmat
270:
271: ! *****
272:
273: END MODULE part3_mod
274: *****
275:
```