Matrix Function Examples

TK5 includes many new functions for calculations involving matrices. Although, many of these functions were previously available through the TK Library, the performance will now be much better. This collection of examples introduces the reader to the use of these new functions.

Some of the new functions can save you time in processing lists. For example, let's say you want to create a list c which is made up of the differences between the elements in two existing lists a and b. In the past, you could either do a list solve or you could create a procedure function such as the following.

```
N = Length('a)
For i = 1 to N
  'c[i] = 'a[i] – 'b[i]
Next i
```

Now, a single function call does the job…

```
Call $VVDIF('a,'b,'c)
```

This is a significant reduction in required effort.

Similarly, if you have a list called x and you want to subtract the value of some variable k from each of the values in x, you can use

```
Call $NVSUM(-k, 'x, 'y)
```

The quickest way to create and view a new matrix is with the following function calls.

```
Call $MZERO('S,3,3)        ; creates a 3x3 matrix of zeros.
Call ShowMatrix('S)        ; displays the matrix in a table.
```

Upon solving (F9), TK displays the following interactive table with horizontal orientation, representing the matrix S.

List	1	2	3
S[1]	0	0	0
S[2]	0	0	0
S[3]	0	0	0

Behind the scenes, TK automatically created the table subsheet and filled the lists with 0's. It also created the row headings S[1], S[2], and S[3]. The user can now input values into the matrix. There is also a function called $MONE which is similar to $MZERO, except that the resulting matrix is filled with 1's.

Iterative Solutions Involving Matrices

A common problem in many fields (engineering, market research, psychometrics) is one which requires that the determinant of a matrix be set equal to zero. Often, a square, symmetric matrix must be transformed by subtracting a value from each of the diagonal elements such that the resulting determinant equals zero.

Suppose you are working with an engineering problem involving a machine element subjected to a general three-dimensional state of stress. Such a state consists of three normal stresses, sx, sy, and sz, and three shear stresses, txy = tyx, tyz = tzy, and txz = tzx. A 3-D state of stress can be represented in a 3-by-3 matrix form

sx-s	txy	txz
tyx	sy-s	tyz
tzx	tzy	sz-s

It can be shown that the principal stresses on the element are defined by finding the values of s which result in a determinant of 0.

We start with the following stress matrix.

List	1	2	3
S[1]	10	5	3
S[2]	5	10	4
S[3]	3	4	15

These rules can be used to subtract s from each of the diagonal elements of the matrix.

```
call $MIDENTITY('I,3)          ; create a 3x3 identity matrix called I
call $MNPRODUCT('I,-sp,'SP)    ; multiply I by –sp, giving the matrix SP
call $MMSUM('S,'SP,'SS)        ; add S and SP, giving SS
```

For any particular value of sp, TK computes the SS matrix. Here is SS for sp = 7.

List	1	2	3
SS[1]	3	5	3
SS[2]	5	3	4
SS[3]	3	4	8

We can use the following rule to compute the determinant of SS.

```
DSS = $DET('SS)
```

TK reports the value of DSS as –83. Our goal is to make DSS = 0. We can either use TK's built-in iterative solver or the TK Optimizer to solve this problem. The iterative solver will require setting a guess on the variable sheet for the variable sp. It will also require a modification to the equation computing the determinant.

If known(‘sp) then DSS = $DET('SS)

This conditional rule prevents the function call from being executed until the iterative solver has been launched. Since sp is the guess variable, TK will not call the function until the guess has been triggered. If you do not use the known function, TK will respond with an Inconsistent message. Here is the variable sheet before iteration.

Status	Input	Name	Output	Unit	Comment
	0	DSS			
Guess	7	sp			

Here is the corresponding solution.

Status	Input	Name	Output	Unit	Comment
	0	DSS			
		sp	4.93448179		

Here is the resulting SS matrix.

List	1	2	3
SS[1]	5.0655182	5	3
SS[2]	5	5.0655182	4
SS[3]	3	4	10.065518

Note that there are as many valid solutions as rows in the matrix. Different initial guesses can be used to find the other solutions. In fact, there is a single function call that can be used to generate a list of all three solutions without any iteration whatsoever.

Call $EIGVAL(‘S,’ser,’sei)

This function generates the eigenvalues of the original S matrix. The results are given in two lists representing the real (ser) and imaginary (sei) portions of the eigenvalues. Here is the ser list. The sei list contains all 0’s for this example, indicating that all the solutions are real numbers.

Value
4.93448178629862
19.9664796487685
10.0990385649329

Notice that the first value is the same as that obtained using the iterative solver. Using the EIGVAL function is much simpler. We went the long way first to show how the iterative solver can interact with matrices.

In engineering terms, these three eigenvalues represent the principal stresses on the element. We could also use the $EIGVEC function to simultaneously compute the principal stresses (eigenvalues) and the corresponding direction cosines (eigenvectors).

Call $EIGVEC(‘S,’SEVR,’SEVI,’ser,’sei)

The function arguments represent the name of the matrix to process, the two matrices representing the real and imaginary components of the eigenvectors, and the two lists representing the real and imaginary components of the eigenvalues. Here is the resulting SEVR matrix, containing the direction cosines in the rows corresponding with the principal stresses in the rows of the ser list.

List	1	2	3
SEV1[1]	-.67131891	.735415675	-.09216674
SEV1[2]	.47522675	.522525959	.70790264
SEV1[3]	-.56876222	-.43142833	.70027075

We can add rules to do related calculations. For example, the maximum shear stress on the element is defined as half of the difference between the largest and smallest principal stress value. The corresponding rule could be written as follows.

MaxShear = (max(‘ser) – min(‘ser))/2

Interacting with Matrices on the Variables Sheet

Some users may want to input values and display the matrix and list solutions as variables on the variables sheet. There are several benefits. You can assign units and get conversions independently for each value. You can organize the solutions and display them concisely on a single worksheet. You can add comments. You can backsolve.

The following rules place the normal and shear stress values from the variables sheet into the matrix elements. The PLACE function is used to assign values to lists.

Rule
place('S[1],1) = sx
place('S[1],2) = txy
place('S[1],3) = txz
place('S[2],1) = txy
place('S[2],2) = sy
place('S[2],3) = tyz
place('S[3],1) = txz
place('S[3],2) = tyz
place('S[3],3) = sz

It is very important to note that the sequence of rules on the rules sheet is critical to the performance of the program when processing of lists and matrices occur. In this case, the $EIGVEC function call must occur after the rules which fill the input matrix and before the rules which extract values from the results matrix. If you mistakenly place the $EIGVEC function call at the end of the rule sheet, the variable sheet will not be updated as expected. The matrix gets updated after the values are sent to the variable sheet.

The following rules place the three principal stresses on the variables sheet along with the corresponding angles. PLACE is not required when pulling values from lists.

Rule
s1 = 'ser[1]
α1 = acosd('SEV1[1][1])
β1 = acosd('SEV1[1][2])
γ1 = acosd('SEV1[1][3])
s2 = 'ser[2]
α2 = acosd('SEV1[2][1])
β2 = acosd('SEV1[2][2])
γ2 = acosd('SEV1[2][3])
s3 = 'ser[3]
α3 = acosd('SEV1[3][1])
β3 = acosd('SEV1[3][2])
γ3 = acosd('SEV1[3][3])

The Variables can now be organized and assigned units and comments.

St	Input	Name	Output	Unit	Comment
					Normal Stresses
	10	sx		psi	normal to x plane
	10	sy		psi	normal to y plane
	15	sz		psi	normal to z plane
					Shear Stresses
	5	txy		psi	on x,y plane
	3	txz		psi	on x,z plane
	4	tyz		psi	on y,z plane
					Principal Stresses and Angles
		s1	4.93448	psi	
		α1	132.169	deg	
		β1	42.6576	deg	
		γ1	95.2883	deg	
		s2	19.9665	psi	
		α2	61.6259	deg	
		β2	58.4982	deg	
		γ2	44.9355	deg	
		s3	10.099	psi	
		α3	124.664	deg	
		β3	115.558	deg	
		γ3	45.5513	deg	
		MaxShear	7.516	psi	Maximum shear stress

The TK Optimizer can now be used to determine the value of sy resulting in a MaxShear value of 8. After a few iterations, the variables are updated as follows and we see that the value of sy is 7.40517.

St	Input	Name	Output	Unit	Comment
					Normal Stresses
	10	sx		psi	normal to x plane
	7.40517	sy		psi	normal to y plane
	15	sz		psi	normal to z plane
					Shear Stresses
	5	txy		psi	on x,y plane
	3	txz		psi	on x,z plane
	4	tyz		psi	on y,z plane
					Principal Stresses and Angles
		s1	19.3502	psi	
		α1	61.1359	deg	
		β1	63.061	deg	
		γ1	41.4546	deg	
		s2	3.35018	psi	
		α2	56.337	deg	
		β2	145.147	deg	
		γ2	82.0088	deg	
		s3	9.70482	psi	
		α3	132.689	deg	
		β3	110.385	deg	
		γ3	49.6642	deg	
		MaxShear	8	psi	Maximum shear stress