MATC Manual

CSC – IT Center for Science

April 30, 2009

1 Introduction

MATC is a library for the numerical evaluation of mathematical expressions. In the Elmer package it can be used in defining expressions in the command file of ElmerSolver. The expressions may be evaluated either while the command file is read in, or at the time of execution. MATC may also be used in the command window of ElmerPost.

2 MATC Variables

There are two types of variables in MATC: matrices and strings. Both are actually stored the same way in a double precision array, so that storing large arrays of strings is wasteful sizewise.

A variable is created by assigning value to it:

$$x = 1.$$

The above statement creates a 1×1 matrix whose name is x.

The statement

$$k = “hello''$$

creates a 1×5 string variable k. That is, string variables are created with enclosed in ””-marks.

Variables can be indexed by two row vectors. The statement

$$x(0, 0 : 5) = x(0, 5 : 0)$$

reverses the order of the first six elements of the first row of a variable called x. Indexing thus begins from zero.

Yet one example of creating a variable (or affecting its values) is

$$x(0 : 9) = 1234$$

Variable x will be a 1×10 matrix, if not previously created and bigger. It’s elements are 1 2 3 4 1 2 3 4 1 2. The example shows that you don’t have to give the row index if it’s zero, and that when scanning for values for elements, the values end, they are repeated from the beginning.

Size of the variables are dynamic. If variable x is non-existent the statement

$$x(9, 9) = 1$$

creates a new matrix whose size is 10×10 and is zero except for the last element whose value is 1.

Another way of indexing a matrix is logical expression whose size is the same as the matrix which is being indexed. The statement

$$x(x < 0.05) = 0.05$$

sets the values of matrix x which are less than 0.05 to the value 0.05.

3 MATC general flow control structures

The syntax of the flow control structures of MATC are given below.

```
if ( expr ) expr; else expr;
if ( expr )

    expr;
    ...
    expr;
 else
    expr;
    ...
    expr;


for( i=vector ) expr;
for( i=vector )

    expr;
    ...
    expr;


while( expr ) expr;
while( expr )

    expr;
    ...
    expr;
```

4 MATC Operators

Assume the following definitions:

- a,b and c ordinary matrices
- l,t and r logical matrices
- s,n and m scalars

`b=a'`
: is transpose of matrix a.

`b=@a`
: evaluate content of a string variable a as a MATC statement.

`t=~l`
: elementwise logical not of if x is not zero.

`b=a^s`
: if a is a square matrix and s is integral, a matrix power is computed, otherwise an elementwise power.

`c=a*b`
: if a and b are compatible for matrix product, that is computed, otherwise if they are of the same size or at least one of them is scalar, an elementwise product is computed.

`c=a#b`
: elementwise multiplication of a and b.

`c=a/b`
: is fraction of a and b computed elementwise.

`c=a+b`
: is sum of matrices a and b computed elementwise.

`c=a-b`
: is difference of matrices a and b computed elementwise.

`l=a==b`
: equality of matrices a and b elementwise.

`l=a<>b`
: inequality of matrices a and b elementwise.

`l=a<b`
: true if a is less than b computed elementwise.

`l=a>b`
: true if a is greater than b computed elementwise.

`l=a<=b`
: true if a is less than or equal to b computed elementwise.

`l=a>=b`
: true if a is greater than or equal to b computed elementwise.

`a=n:m`
: return a vector of values starting from n and ending to m by increment of (plus-minus) one.

`r=l&t`
: elementwise logical and of a and b.

`l=a|b`
: elementwise logical or of a and b.

`c=a?b`
: reduction: set values of a where b is zero to zero.

`b=n m % a`
: resize a to matrix of size n by m.

`b=a`
: assing a to b.

5 MATC Function Statement

The syntax of the function definition is given below.

```
function name(arg1,arg2,...)
!
! Optional function description (seen with help("name"))
!
import var1,var2
export var3,var4

    expr;
     ...
    expr;

    _name = value
```

Functions have their own list of variables. Global variables are not seen in this function unless imported by import or given as arguments. Local variables can be made global by the export statement.

Functions, if returing matrices, behave in many ways as variables do. So if you have defined function mult as follows

```
function mult(a,b)

   _mult = a*b;
```

you can get element (3,5) of the a times b matrix by the following statement

```
mult(x,y)[3,5]
```

or diagonal values of the same matrix by

```
diag(mult(x,y)).
```

6 MATC Internal Functions

`funcdel(name)`
: Delete given function definition from parser.

`funclist(name)`
: Give header of the function given by name.
SEE ALSO: help.

`str = sprintf(fmt[,vec] )`
: Return a string formatted using fmt and values from vec. A call to corresponding C-language function is made.

`vec = sscanf(str,fmt)`
: Return values from str using format fmt. A call to corresponding C-language function is made.

`special = matcvt(matrix, type)`
: Makes a type conversion from MATC matrix double precision array to given type, which can be one of the following: ”int”, ”char” or ”float”.
SEE ALSO: cvtmat, fwrite.

`r = cvtmat( special, type )`
: Makes a type conversion from given type to MATC matrix. Type can be one of the following: ”int”, ”char” or ”float”.
SEE ALSO: fread, matcvt.

`r = eval(str)`
: Evaluate content of string str. Another form of this command is @str.

`source(name)`
: Execute commands from file given name.

`help or help(”symbol”)`
: First form of the command gives list of available commands. Second form gives help on specific routine.

`str = fread(fp,n)`
: Read n bytes from file given by fp. File pointer fp shoud have been obtained from a call to fopen or freopen, or be the standard input file stdin. Data is returned as function value.
SEE ALSO: fopen,freopen,fgets,fscanf,matcvt,cvtmat.

`vec = fscanf(fp,fmt)`
: Read file fp as given in format. Format fmt is equal to C-language format. File pointer fp shoud have been obtained from a call to fopen or freopen, or be the standard input.
SEE ALSO: fopen,freopen,fgets,fread,matcvt,cvtmat.

`str = fgets(fp)`
Read next line from fp. File pointer fp shoud have been obtained from a call to fopen or freopen or be the standard input.
SEE ALSO: fopen,freopen,fread,fscanf,matcvt,cvtmat.

`n = fwrite(fp,buf,n)`
Write n bytes form buf to file fp. File pointer fp shoud have been obtained from a call to fopen or freopen or be the standard output (stdout) or standard error (stderr). Return value is number of bytes actually written. Note that one matrix element reserves 8 bytes of space.
SEE ALSO: fopen,freopen,fputs,fprintf,matcvt,cvtmat.

`n = fprintf(fp,fmt[, vec])`
Write formatted string to file fp. File pointer fp shoud have been obtained from a call to fopen or freopen or be the standard output (stdout) or standard error (stderr). The format fmt is equal to C-language format.
SEE ALSO: fopen,freopen,fputs,fwrite,matcvt,cvtmat.

`fputs(fp,str)`
Write string str to file fp. File pointer fp should have been obtained from a call to fopen or freopen or be the standard input (stdin).
SEE ALSO: fopen,freopen,fwrite,matcvt,cvtmat.

`fp = fopen(name,mode)`
Reopen file given previous file pointer, name and access mode. The most usual modes are ”r” for reading and ”w” for writing. Return value fp is used in functions reading and writing the file.
SEE ALSO: fopen.

`fp = freopen(fp,name,mode)`
Reopen file given previous file pointer, name and access mode. The most usual modes are ”r” for reading and ”w” for writing. Return value fp is used in functions reading and writing the file.
SEE ALSO: fopen.

`fclose(fp)`
Close file previously opened with fopen or freopen.
SEE ALSO: fopen, freopen.

`save(name, a[,ascii_flag] )`
Close file previously opened with fopen or freopen.
SEE ALSO: fopen, freopen.

`r = load(name)`
Load matrix from a file given name and in format used by save command.
SEE ALSO: save.

`r = min(matrix)`
Return value is a vector containing smallest element in columns of given matrix. r = min(min(matrix)) gives smallest element of the matrix.

`r = max(matrix)`
Return value is a vector containing largest element in columns of given matrix. r = max(max(matrix)) gives largest element of the matrix.

`r = sum(matrix)`
Return vector is column sums of given matrix. r = sum(sum(matrix)) gives the total sum of elements of the matrix.

`r = trace(matrix)`
Return value is the sum of matrix diagonal elements.

`r = det(matrix)`
: Return value is determinant of given square matrix.

`r = inv(matrix)`
: Invert given square matrix. Computed also by operator $^{-1}$

`r = tril(x)`
: Return the lower triangle of the matrix x.

`r = triu(x)`
: Return the upper triangle of the matrix x.

`r = eig(matrix)`
: Return eigenvalues of given square matrix. The expression r(n,0) is real part of the n:th eigenvalue, r(n,1) is the imaginary part respectively.

`r = jacob(a,b,eps)`
: Solve symmetric positive definite eigenvalue problem by Jacob iteration. Return values are the eigenvalues. Also a variable eigv is created containing eigenvectors.

`r = lud(matrix)`
: Return value is LUD decomposition of given matrix.

`r = hesse(matrix)`
: Return the upper hessenberg form of given matrix.

`r = eye(n)`
: Return n by n identity matrix.

`r = zeros(n,m)`
: Return n by m matrix with elements initialized to zero.

`r = ones(n,m)`
: Return n by m matrix with elements initialized to one.

`r = rand(n,m)`
: Return n by m matrix with elements initialized with random numbers from zero to one.

`r = diag(matrix) or r=diag(vector)`
: Given matrix return diagonal entries as a vector. Given vector return matrix with diagonal elements from vector. r = diag(diag(a)) gives matrix with diagonal elements from matrix a, otherwise elements are zero.

`r = vector(start,end,inc)`
: Return vector of values going from start to end by inc.

`r = size(matrix)`
: Return size of given matrix.

`r = resize(matrix,n,m)`
: Make a matrix to look as a n by m matrix. This is the same as r = n m

`where(a)`
: Return a row vector giving linear index to a where a is not zero.

`exists(name)`
: Return true (non-zero) if variable by given name exists otherwise return false (=0).

`who`
: Give list of currently defined variables.

`format(precision)`
: Set number of digits used in printing values in MATC.

7 Basic math routines

`r = sin(x)`

`r = cos(x)`

`r = tan(x)`

`r = asin(x)`

`r = acos(x)`

`r = atan(x)`

`r = sinh(x)`

`r = cosh(x)`

`r = tanh(x)`

`r = exp(x)`

`r = ln(x) Natural logarithm.`

`r = log(x) Base 10 logarithm.`

`r = sqrt(x)`

`r = ceil(x) Smallest integer not less than x.`

`r = floor(x) Largest integer not more than x.`

`r = abs(x)`

`r = pow(x,y)`