Matlab与C混合编程API之mxCreateDoubleMatrix

zcl.space

目录

1 引言

通过matlab与C混合编程之基本原理, 我们知道，在matlab与C的混合编程中，桥梁函数 mexFunction 发挥着至关重要的作用。在这个桥梁函数中，通过调用matlab提供的API可以很方便的实现matlab和C之间的混合编程。本节介绍一个API，名称为：mxCreateDoubleMatrix 。该API的作用是创建二维的双精度浮点数组。

2 调用语法

在C文件中，调用 mxCreateDoubleMatrix 的语法为：

```
#include "matrix.h"
mxArray *mxCreateDoubleMatrix(mwSize m, mwSize n,
  mxComplexity ComplexFlag);
```

3 输入输出参数

输入参数如表1所示

表 1: mxCreateDoubleMatrix输入输出参数表

参数	性质	描述	类型
m	输入	矩阵行数	mwSize
n	输入	矩阵列数	mwSize
ComplexFlag	输入	是否包含复数元素	ComplexFlag

mwSize 是matlab自己定义的用于表示矩阵大小的类型。在C语言中，使用 int 也可以。但是 mwSize 具有跨平台的特性。默认情况下， mwSize 和 C中的 int 是等价的。当使用 mex -largeArrayDims 编译选项的时候， mwSize 与C中的 size_t 等价。 mxComplexity 用来指定矩阵中数值元素是否包含虚部（即，元素是否既有实部又有虚部）。 mxComplexity 的值只有两个 mxREAL 和 mxComCOMPLEXITY ，前者表示矩阵中元素有虚部，后者表示矩阵中元素没有虚部。之所以存在这样的类型，是因为matlab向C传送数值矩阵的时候，实部和虚部是分开完成的。

如果成功创建矩阵，则该函数的输出（即返回值）是一个指向 mxArray 类型的指针。否则返回NULL.

4 功能描述

该函数创建一个 m 行 n 列的矩阵，矩阵中的元素类型依 mxComplexity 的值而定。如果 mxComplexity 的值是 mxREAL 则矩阵中的元素类型全是实数，matlab会分配足够的空间来存放这些实数，并将这些实数初始化为0。如果 mxComplexity 的值是 mxCOMPLEX ，则matlab分

配足够的空间来存放这个复数空间（实部地址为 pr ,虚部地址为 pi ），实部和虚部都初始化为0。

通过调用 mxDestroyArray 来释放 mxCreateDoubleMatrix 创建的矩阵占用的内存空间。

5 一个例子

举一个很简单的例子，写一个函数返回一个实数乘以2的值。这个例子很简单，但是包含了matlab到C之间互传数据的过程。

```
#include "mex.h"
void timestwo(double y[], double x[])
{
  y[0] = 2.0*x[0];
}

void mexFunction( int nlhs, mxArray *plhs[],
                  int nrhs, const mxArray *prhs[] )
{
  double *x,*y;
  size_t mrows,ncols;

  /* Check for proper number of arguments. */
  if(nrhs!=1) {
    mexErrMsgIdAndTxt( "MATLAB:timestwo:invalidNumInputs",
            "One input required.");
  } else if(nlhs>1) {
    mexErrMsgIdAndTxt( "MATLAB:timestwo:maxlhs",
            "Too many output arguments.");
  }

  /* The input must be a noncomplex scalar double.*/
  mrows = mxGetM(prhs[0]);
  ncols = mxGetN(prhs[0]);
  if( !mxIsDouble(prhs[0]) || mxIsComplex(prhs[0]) ||
      !(mrows==1 && ncols==1) ) {
    mexErrMsgIdAndTxt(
    "MATLAB:timestwo:inputNotRealScalarDouble",
    "Input must be a noncomplex scalar double.");
  }

  /* Create matrix for the return argument. */
  plhs[0] = mxCreateDoubleMatrix((mwSize)mrows,
                                 (mwSize)ncols, mxREAL);

  /* Assign pointers to each input and output. */
  x = mxGetPr(prhs[0]);
  y = mxGetPr(plhs[0]);

  /* Call the timestwo subroutine. */
  timestwo(y,x);
```

```
42 }
```

代码的第2-5行实现了乘以2的功能。代码的第7-42行实现了 mexFunction 。代码的第14-20行用来检查输入数据的格式，这些代码的主要作用是方便调试，暂时不管。代码的第23-24行获得了输入数据的维度，用于为输出数据指定内存空间大小。第33-34行调用了本文介绍的 mxCreateDoubleMatrix API。并将返回地址赋给 plhs[0] 。

第33行至第41行是 mexFunction 的核心。通过 mxCreateDoubleMatrix 申请保存输出的内存空间，并把地址赋给 plhs[0] 这样matlab脚本就可以得到C函数的输出。通过调用 mxGetPr 函数把 plhs[0] 的值赋给 y ，然后在C函数中为 y 指向的内存单元复制，这里 y 也是指针。 mxGetPr 函数实现了 mxArray 指针向 double 类型指针的转换。

6 另一个例子

再来一个例子，本例完成标量和矩阵相乘的功能。代码如下：

```
1  #include "mex.h"
2  void xtimesy(double x, double *y,
3               double *z, size_t m, size_t n)
4  {
5    mwSize i,j,count=0;
6
7    for (i=0; i<n; i++) {
8      for (j=0; j<m; j++) {
9        *(z+count) = x * *(y+count);
10       count++;
11     }
12   }
13 }
14
15 /* the gateway function */
16 void mexFunction( int nlhs, mxArray *plhs[],
17                   int nrhs, const mxArray *prhs[])
18 {
19   double *y,*z;
20   double  x;
21   size_t mrows,ncols;
22
23   /*  check for proper number of arguments */
24   /* NOTE: You do not need an else statement when
25      using mexErrMsgIdAndTxt within an if statement,
26      because it will never get to the else statement
27      if mexErrMsgIdAndTxt is executed.
28      (mexErrMsgIdAndTxt breaks you out of the
29     MEX-file) */
30   if(nrhs!=2)
31     mexErrMsgIdAndTxt(
32             "MATLAB:xtimesy:invalidNumInputs",
33             "Two␣inputs␣required.");
```

```
34    if(nlhs!=1)
35      mexErrMsgIdAndTxt(
36              "MATLAB:xtimesy:invalidNumOutputs",
37              "One output required.");
38
39    /* check to make sure the first input argument is
40       a scalar */
41    if( !mxIsDouble(prhs[0]) ||
42        mxIsComplex(prhs[0]) ||
43        mxGetN(prhs[0])*mxGetM(prhs[0])!=1 ) {
44      mexErrMsgIdAndTxt( "MATLAB:xtimesy:xNotScalar",
45              "Input x must be a scalar.");
46    }
47
48    /*  get the scalar input x */
49    x = mxGetScalar(prhs[0]);
50
51    /*  create a pointer to the input matrix y */
52    y = mxGetPr(prhs[1]);
53
54    /*  get the dimensions of the matrix input y */
55    mrows = mxGetM(prhs[1]);
56    ncols = mxGetN(prhs[1]);
57
58    /*  set the output pointer to the output matrix */
59    plhs[0] = mxCreateDoubleMatrix( (mwSize)mrows,
60                                    (mwSize)ncols, mxREAL);
61
62    /*  create a C pointer to a copy of the
63        output matrix */
64    z = mxGetPr(plhs[0]);
65
66    /*  call the C subroutine */
67    xtimesy(x,y,z,mrows,ncols);
68
69  }
```

本例的第2-13行完成了标量乘以矩阵的c函数。在桥梁函数中，matlab传入的矩阵 y 体现为指向矩阵 y 的第一个元素的地址。这个通过第52行代码实现。

```
y = mxGetPr(prhs[1]);
```

第55行和第56行代码分别调用 mxGetM 和 mxGetN 获得传入矩阵 y 的行数和列数。第59行调用 mxCreateDoulbeMatrix 为输出分配内存空间，创建了行数为 mrows ,列数为 ncols 的实数矩阵空间。第64行把这个是数据矩阵空间的地址通过 mxGetPr 赋给 z ，经过第64行，plhs[0] 指向内存单元就和 z 指向的内存单元相同。

最后需要注意的一点：在matlab中矩阵的索引是按 **列** 进行的，比如一个矩阵x=[1 2; 3 4]，则用一个索引变量遍历x结果为：x(1) = 1,x(2) = 3,x(3) =2, x(4) =4，也就是说矩阵的第二个元素是3不是2。用两个索引变量为：x(1,1) = 1, x(1,2) =2,x(2,1) =x3,x(2,2)=4。之所以

交代这个的原因是在matlab中，传入的是指向矩阵第一个元素的指针，其加1指向的是矩阵的第二个元素，也就是矩阵的第二列第一个元素，不是矩阵的第一行第二列的那个元素。

7 mxCreateDoubleScalar

既然介绍了 mxCreateDoulbeMatrix ，就顺带提一下另一个API mxCreateDoubleScalar 。这个API是 mxCreateDoulbeMatrix 的特殊形式，其创建指向一个double标量的指针，并初始化。其语法为：

```
pa = mxCreateDoubleScalar(value);
```

pa是个地址指向一个doulbe值，大小为 value 。

用 mxCreateDoulbeMatrix 完成就是：

```
pa = mxCreateDoubleMatrix(1, 1, mxREAL);
*mxGetPr(pa) = value;
```

显然，在桥梁变量里创建标量的时， mxCreateDoubleScalar 更方便。

8 尾声

本文通过简单介绍了 mxCreateDoulbeMatrix API，并通过两个例子阐述了其用法，还介绍了其特殊形式 mxCreateDoubleScalar 。在两个例子中顺带介绍了 mxGetM mxGetN mxGetPr ，关于这后三个简单的API就不另开博文做专门介绍了。