第一部分：Fortran 调用 Matlab 引擎

1　什么是 Matlab 引擎

所谓 Matlab 引擎（engine），是指一组 Matlab 提供的接口函数，支持 C/C++、Fortran 等语言，通过这些接口函数，用户可以在其它编程环境中实现对 Matlab 的控制。可以主要功能有：

★ 打开/关闭一个 Matlab 对话；

★ 向 Matlab 环境发送命令字符串；

★ 从 Matlab 环境中读取数据；

★ 向 Matlab 环境中写入数据。

与其它各种接口相比，引擎所提供的 Matlab 功能支持是最全面的。通过引擎方式，应用程序会打开一个新的 Matlab 进程，可以控制它完成任何计算和绘图操作。对所有的数据结构提供 100%的支持。同时，引擎方式打开的 Matlab 进程会在任务栏显示自己的图标，打开该窗口，可以观察主程序通过 engine 方式控制 Matlab 运行的流程，并可在其中输入任何 Matlab 命令。

注：Fortran 程序与在 Matlab 初次安装时，会自动执行一次：matlab /regserver 将自己在系统的控件库中注册。如果因为特殊原因，无法打开 Matlab 引擎，可以在 Dos 命令提示符后执行上述命令，重新注册。

2　Fortran 调用 Matlab 语句原理

实际上，Fortran 通过引擎函数调用 MATLAB，实质是把 Matlab 作为一个 ActiveX 服务器，也叫 Matlab 计算引擎。当 Fortran 程序调用某个 MATLAB 函数或命令时，首先通过引擎函数启动 Matlab 并建立 ActiveX 通道，然后把这个

函数或命令通过 ActiveX 通道传给 Matlab，由 Matlab 在后台执行（这可以分成两步来完成：第一步将 mxArray 转换成 Matlab 可理解的形式。用 mxCreate 来创建一个和要传递的数据类型大小相同的矩阵 mxArray；第二步将矩阵放入 Matlab 工作区中，用程序 engPutVariable 和 engEvalString 来完成。）。

下面给出几个比较重要的 Fortran 程序可以调用的引擎函数的功能说明，其他函数可以通过 Matlab 帮助找到：

engOpen：启动 MATLAB 引擎，建立 ActiveX 通道；

engClose：关闭 MATLAB 引擎，即关闭 ActiveX 通道；

engGetMatrix：从 MATLAB 引擎得到 MATLAB 数组值；

engPutMatrix：把一个 MATLAB 数组传给 MATLAB 引擎；

engEvalString：执行一个 MATLAB 命令；

engOutputBuffer：建立一个缓冲区以储存 MATLAB 的文本输出。

当然，要实现 Matlab 函数或命令的调用执行，除了调用 Matlab 语言的引擎函数外，还需要调用 Matlab 提供的 API(应用程序接口)函数。由于 Fortran 与 Matlab 语言的计算单元不一样，前者以数位运算单位，而后者以矩阵（mxArray 结构体）为基本的运算单位，导致运算的数据类型不能直接匹配。Matlab API 函数采用独特的 mxArray 数据类型进行运算，能将 Fortran 程序运行的双精度实型变量转化成矩阵类型；也可以将 Matlab 矩阵类型变化成 Fortran 程序运行的双精度实型变量。

Fortran 调用 Matlab 引擎编程步骤如下：

(1) 在 Fortran 编译器中声明引擎函数，API 函数等。

(2) 在 Fortran 编译器中定义指针。在 Matlab 与 Fortran 混合编程中，指针是传递数据的唯一方式。因此，每一个传递到 Matlab 中的 Fortran 数据都需要一个指针来指向其在 Matlab 中对应的 mxArray 结构体。

(3) 通过 Matlab 引擎函数启动 Matlab 引擎，让 Matlab 在后台开始运行。

(4) 创建 Matlab 矩阵，将 Fortran 变量传递给 Matlab 矩阵。

(5) 执行所需的 Matlab 命令完成运算、函数调用、绘图等。

(6) 将 Matlab 运算结果返回给 Fortran 程序。

3　编译器设置

3.1　Compaq Visual Fortran 6.6 中的设置

（1）设置头文件目录

在 tool->options->Directories 里的 Include files 添加（假设 D:\Program Files\MATLAB 是 Matlab 安装路径）：

D:\Program Files\MATLAB\R2008a\extern\include

（2）设置库文件目录

Library files 里添加（假设 D:\Program Files\MATLAB 是 Matlab 安装路径）：

D:\Program Files\MATLAB\R2008a\extern\lib\win32\microsoft

（3）添加链接库输入项

在 project->setting->link->Object/library modules 添加 libmx.lib libeng.lib

★ 注意*.lib 文件之间用空格分隔，不可用逗号分隔。

（1）和（2）设置一次就可以，而（3）是每新建一个 Project 都要重新设置。也可以在 Compaq Visual Fortran6.6 中可以保存（3）中的设置，方法：点击菜单 Files->Save Fortran Environment，在 Save Fortran Console Environment as：输入一个名字，点击 Save Environment。这样就可以在以后直接使用这一编译环境，例如：在新建一个 Project 后会弹出对话框”Would you like to consider applying options from a Saved Fortran Console Environment?”点击”是(Y)”，在对话框中 List of Console Environment 中，选择前面 save 过的环境名称，点击”Apply”按钮并确认，在接下来的对话框中点击”OK”。

3.2 Intel Visual Fortran 10.1 中的设置

（1）设置头文件目录

在菜单中选择工具->选项，在对话框左边选择 Intel(R) Fortran->Compilers，在右边的 Includes 里添加：D:\Program Files\MATLAB\R2008a\extern\include，如图 1。

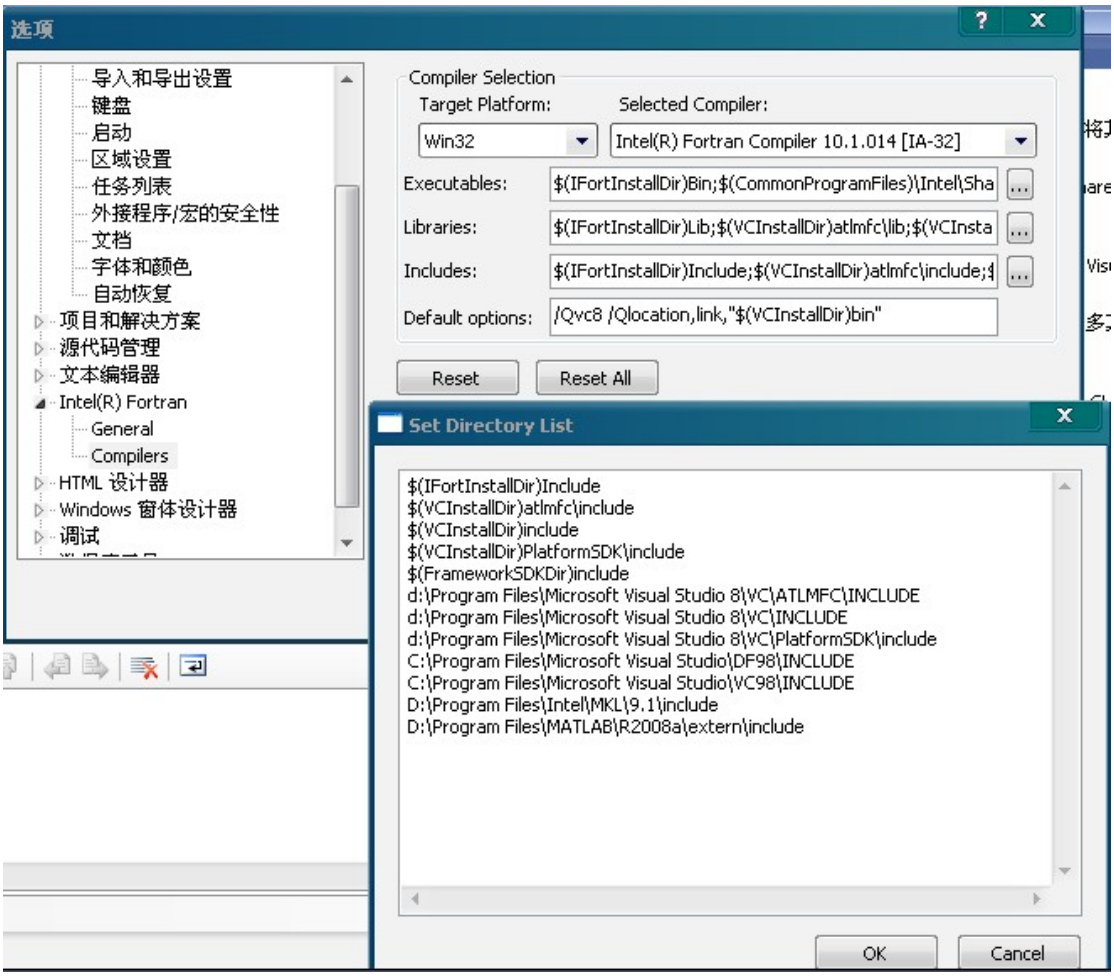

图 1

（2）设置库文件目录

同理，在右边的 Libraries 里添加：

D:\Program Files\MATLAB\R2008a\extern\lib\win32\microsoft

（3）添加链接库输入项

在解决方案资源管理器里选择对应的项目单机右键选择最下面的“属性”弹出属性对话框，在右边选择 Linker->Input，在 Additional Dependencies 里添加：libeng.lib libmx.lib，如图 2。

★ 注意：*.lib 文件之间用空格分隔，不可用逗号分隔。

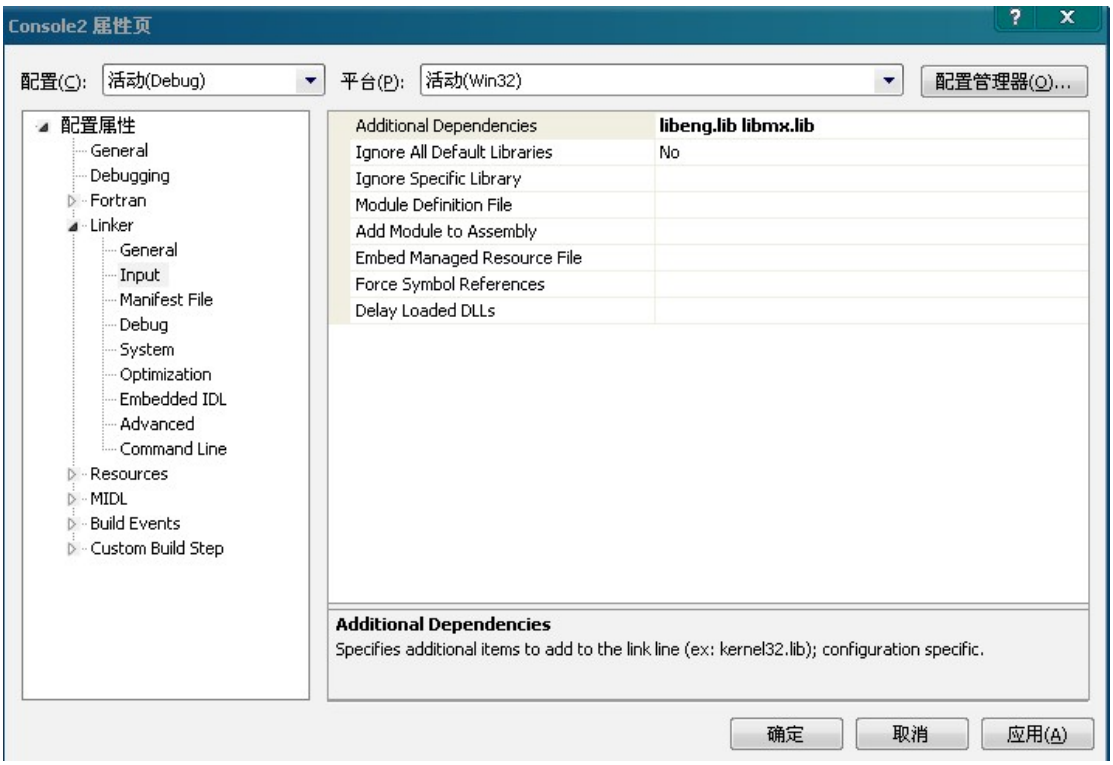

图　2

（4）设置环境变量

现在生成时就没问题出现了，但是运行程序时可能会出现“因为计算机中丢失 libeng.dll”等类似问题。第（4）步就解决这个问题的，如图 3。

把 D:\Program Files\MATLAB\R2008a\bin\win32;添加到 PATH 里边（如果是 64 位系统则添加 D:\Program Files\MATLAB\R2008a\bin\win64）。

在 Win7 下具体可以这样做：在桌面上选择“计算机”图标，右键弹出选单，在其中选择“属性”，然后在弹出的窗口中选中左边的高级系统设置，在属性对话框里选择“高级”选项卡，载选择“环境变量”，在新对话框里的系统环境下

边找到 PATH 这一项选中，然后编辑，在后边添加

D:\Program Files\MATLAB\R2008a\bin\win32;

64 位系统选择

D:\Program Files\MATLAB\R2008a\bin\win64;

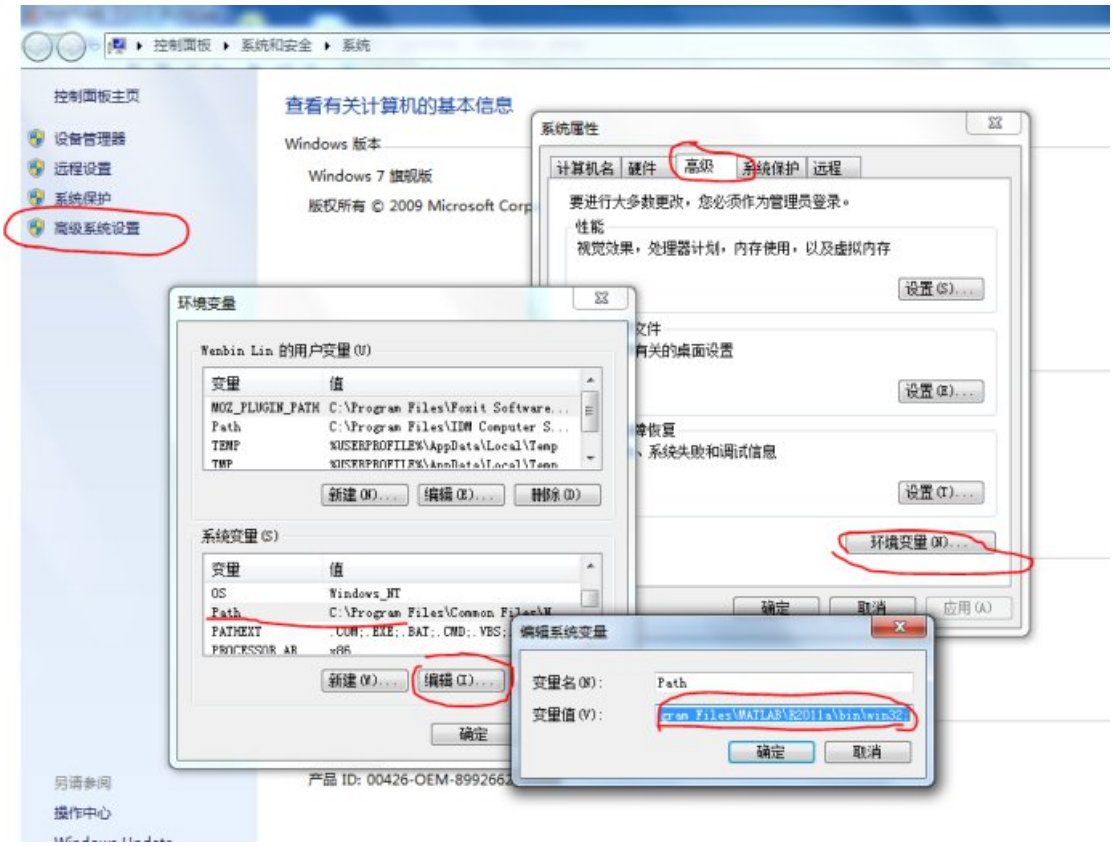

图　3

★ 注意：不要删除 path 里原有的其他路径。

如果设置环境变量前打开了 visual studio ，那么需要重启 visual studio。

第二部分：Matlab 调用 Fortran 程序

1　创建 MEX 文件实现对 Fortran 的调用

1.1　MEX 文件介绍

Matlab MEX 文件是 Matlab 系统的外部程序调用接口。MEX 文件是由 C/Fortran 语言编写的，编译后生成 Matlab 动态链接子程序，可在 Matlab 环境下导入和执行，如同 Matlab 的内置函数一样。这样可以使用 Fortran 语言进行算法设计，然后再 Matlab 环境下调用，提高 Matlab 环境中数据处理效率，主要应用有：对已有的 C/Fortran 程序，可通过 MEX 方式在 Matlab 环境中直接调用；对影响 Matlab 执行速度的 FOR 循环，可以编写相应的 C/Fortran 子程序完成相同功能，并编译成 MEX，提高运行速度。

MEX 文件实际上为一种 Matlab 专用的动态链接库文件。

1.2　MEX 文件调用的基本原理

Matlab 和 Fortran 语言的运算单位不同，Matlab 以矩阵（mxArray 结构体）为基本数据结构，而在 Fortran 中，文件是由按顺序排列的记录组成的，而记录是数值或字符的序列，是 Fortran 程序输入输出的基本单位。记录有两种格式：格式记录和无格式记录。格式记录中，数据在文件中的存放是用 ASCII 码形式；无格式记录中，数据在文件中存放是用二进制代码形式。由于 Fortran 和 Matlab 的数据类型不能直接匹配，故需调用 Matlab 的 API 函数来完成二者的数据转换，其基本原理是：Matlab 将需要传递的 mxArray 型数据的内存地址作为一个整型

数值传递给Fortran程序，然后在Fortran中，API提供的访问函数（access routiness）使用此整数值来访问 mxArray 的内容，并将此值作为内存地址，读取相应内容。

1.3 Fortran 语言 MEX 文件源程序的构成

Fortran MEX 文件通常由两个显著不同的部分组成：(1)计算子程序，是完成计算功能的纯 Fortran 程序，通常为现有的 Fortran 程序代码；(2)入口子程序，是计算子程序与 Matlab 环境之间的接口，用来完成相互之间的调用。

计算子程序实际上被入口子程序当作子程序来调用，如果非常简单，也可以直接嵌入到入口子程序中，但为了保持可读性，一般不建议这么做。

入口子程序名字为 mexFunction，包含 4 个虚拟参数：

prhs，输入参数的 mxArray 类型指针；

nrhs，输入参数个数；

plhs，输出参数的 mxArray 类型指针；

nlhs，输出参数个数。

调用格式为：

subroutine mexFunction (nlhs, plhs, nrhs, prhs)

入口子程序通过 MEX 文件的 API 函数与 Matlab 进行数据交换。

★ 注意：其实上述输入、输出参数的名称非常容易记忆，lhs 代表 Left hand parameters，rhs 代表 right hand parameters，n 代表 number，p 代表 pointer。这里的左手和右手如何划分呢？例如在 MATLAB 中调用求伯特图的函数时，可以这样调用：

[mag,phase,w] = bode(sys)

这里以“=”为分界，左边的参数 mag、phase 和 w 为输出参数，即左手参数，而等号右边的参数 sys，为输入参数，即右手参数。

1.4 编译器配置

在 Matlab 命令提示符下键入配置命令：mex -setup，并按提示进行操作逐步完成。

★：只用 mex -setup 一次即可，以后运行 mex 文件等都不用再 mex -setup。

1.5 MEX 文件的执行流程

以由计算子程序和入口子程序组成的 plus.for 文件为例。首先，在 Matlab 环境下输入命令：mex plus.for；即对 Fortran 文件进行编译。编译完成后会创建出 Matlab 的 MEX 文件，即该目录下出现 plus.mexw32 文件，只需正确的键入该 MEX 文件名及其所需参数即可使用，例如 z=plus(x, y)。可以在 Matlab 命令行下键入 what 指令查看当前路径下是否具有 MEX 文件。

1.6 %val 结构

如果你的 FORTRAN 编译器支持%val 结构(如 Fortran90)，有一种不要入口路径的指针类型，直接用 mxGetPr 或 mxGetPi 来取得数据指针，再用%val 把指针内容传给 Fortrran 子程序，在这里把它作为一个 FORTRAN 的双精度矩阵，这种传递称为“值传递”。如果你的编译器不支持%val 结构，就必须采用 mxCopy 路线(如 mxCopyPtrToReal8)获得指针的内容。

%Val 结构的定义及语法：

Built-in Function: Changes the form of an actual argument. Passes the argument as an immediate value.

Syntax

result = **%VAL** (*a*)

a (Input): An expression, record name, procedure name, array, character array section, or array element.

You must specify **%VAL** in the actual argument list of a **CALL** statement or function reference. You cannot use it in any other context.

例，不使用%val 结构(Fortran 77)写法：

```
y_pr=mxGetPr(plhs(1))
x_pr=mxGetPr(prhs(1))
call mxCopyPtrToReal8(x_pr,x,size)
```

```
call times2(size,x,y)
call mxCopyReal8ToPtr(y,y_pr,size)
```

使用%val 结构(Fortran 90)写法：

```
y_pr=mxGetPr(plhs(1))
x_pr=mxGetPr(prhs(1))
call times2(size,%val(x_pr),%val(y_pr))
```

1.7 Fortran 语言 MEX 文件中对 Matlab 函数的调用

在 Fortran 语言 MEX 文件中，用户不但可以调用 Fortran 语言的内建函数，而且还可以通过 API 提供的子程序 mexCallMATLAB 来完成对 Matlab 内建函数、运算符、M 文件的调用。

2 创建 DLL 文件实现对 Fortran 的调用

2.1 DLL 文件介绍

动态链接库（Dynamic Link Library，DLL）是一个包含可由多个程序同时使用的代码和数据的库。动态链接提供了一种方法，使进程可以调用不属于其可执行代码的函数。函数的可执行代码位于一个 DLL 中，该 DLL 包含一个或多个已被编译、链接并与使用它们的进程分开存储的函数。多个应用程序可同时访问内存中单个 DLL 副本的内容。使用动态链接库具有如下优点：扩展了应用程序的特性；简化了软件项目的管理与升级；有助于节省内存、实现资源共享；可以用多种编程语言来编写。

2.2 Matlab 调用 Fortran 动态连接库

混合语言程序设计中的主要问题：如何在编程时遵循不同语言中变量和过程的命名约定、堆栈使用约定以及函数调用过程中的参数传递约定。其中堆栈约定确定子过程的参数数目是否可变以及何时进行调用后的清理堆栈工作；命名约定确定标识符是否对大小写敏感及编译后对标识符如何修饰；参数约定确定参数传递时是传值方式还是传址（引用）方式及不同语言之间的数据类型和数据结构如何对应。Fortran 语言调用约定有三种，即：STDCALL 约定、C 约定和 Default 调用约定。

Matlab 可由两种方式调用 Fortran 语言编写的动态链接库。第一种方式是利用 Matlab 生成的 MEX 文件，即第二部分第 1 章节的内容；第二种方式是利用 Fortran 编译器，将 Fortran 语言编写的函数编译成通用 32 位的动态链接库（后缀名为 dll），Matlab 通过几个专门的 API 接口函数调用。

2.3 DLL 文件的生成

生成动态链接库具体步骤如下：

（1）创建一个 Fortran Dynamic Link Library 工程。

（2）在工程中添加已经写好的子程序 dll_test.for 文件。

（3）在 dll_test.for 文件中，subroutine add(x,y,z)之后插入伪注释语句：

```
!DEC$ ATTRIBUTES C,DLLEXPORT :: add
!DEC$ ATTRIBUTES REFERENCE ::z
```

第一句表明库函数 add 在动态链接库外可被调用，并按 C 调用约定，只有声明了 DLLEXPORT 属性，才能从 DLL 中导出过程，该过程也才能为外部程序所调用。第二句指定 z 为引用传递。也可以指定 z 为值传递，即

```
!DEC$ ATTRIBUTES VALUE ::z
```

★ 调用约定规定了过程中所有参数的传递，而 REFERENCE 和 VALUE 属性则最终决定参数的传递（在任何情况下,数组参数只能以引用方式进行传递）。

（4）编译生成 dll_test.DLL 动态链接库文件。

（5）编写和 C 语言等效的 DLL 输出函数定义头文件 dll_test.h，内容如下：

void add(int,int,int*);回车

void 表示“空”，无返回值，int 定义对应的变量为整型变量，int*代表指针类型，表示指向整型变量的指针。

★ 注意要有回车，语句下要有空行。

2.3 DLL 动态链接库的调用

Matlab 利用三个内部函数 loadlibrary、calllib 和 unloadlibrary 调用 Fortran 成的通用 DLL 动态链接库，具体步骤如下：

（1）利用 loadlibrary 加载动态链接库及对应头文件。

（2）利用 calllib 调用动态链接库中的输出函数。

（3）利用 unloadlibrary 释放动态链接库。

在 Matlab 中编写 m 文件：

```
clc
clear
hfile='dll_test.h';
loadlibrary('dll_test.dll', hfile);        %加载 dll 及对应的头文件
x=0;
result=calllib('dll_test', 'add', 3,5,x);   %matlab 里的 calllib 函数返回
disp('result = '),disp(result)
unloadlibrary dll_test
```

第三部分：函数说明

1 引擎函数功能说明

1.1 engOpen：启动 MATLAB 引擎，建立 ActiveX 通道。

Fortran Syntax（语法）：

Integer*4 function engOpen(startcmd)

Integer*4 ep

Character*(*) startcmd

Arguments（参数）：

Startcmd:String to start the MATLAB process. On Windows systems, the startcmd string must be NULL.

Returns（返回值）：

A pointer to an engine handle or NULL if the open fails.

1.2 engClose：关闭 MATLAB 引擎，即关闭 ActiveX 通道。

Fortran Syntax（语法）：

Integer*4 function engClose(ep)

Integer*4 ep

Arguments（参数）：

Ep:Engine pointer

Returns（返回值）：

0 on success, and 1 otherwise. Possible failure includes attempting to terminate a MATLAB engine session that was already terminated.

1.3 engPutMatrix为旧的写法,现替换为engPutVariable:把一个变量传给MATLAB引擎。

Fortran Syntax（语法）:

Integer*4 function engPutVariable(ep,name,pm)

Integer*4 ep,pm

Character*(*) name

Arguments（参数）:

Ep:Engine pointer

Name:Name given to the mxArray in the engine's workspace

Pm:mxArray pointer

Returns（返回值）:

0 if successful and 1 if an error occurs

1.4 engGetMatrix为旧的写法,现替换为engGetVariable:从MATLAB引擎得到变量值。

Fortran Syntax（语法）:

Integer*4 function engGetVariable(ep,name)

Integer*4 ep

Character*(*) name

Arguments（参数）:

Ep:Engine pointer

Name: Name of mxArray to get from MATLAB workspace

Returns（返回值）:

A pointer to a newly allocated mxArray structure, or NULL if the attempt fails. engGetVariable fails if the named variable does not exist.

1.5 engEvalString: 执行一个 MATLAB 命令。

Fortran Syntax（语法）:

Integer*4 function engEvalString(ep,string)

Integer*4 ep

Character*(*) string

Arguments（参数）:

Ep:Engine pointer

String: String to execute

Returns（返回值）:

0 if the command was evaluated by the MATLAB engine session, and nonzero otherwise. Possible reasons for failure include the MATLAB engine session is no longer running or the engine pointer is invalid or NULL.

★ engEvalString 也可以当 subroutine 使用，例如 call engEvalString(ep,string)。

1.6 engOutputBuffer：建立一个缓冲区以储存 Matlab 的文本输出。

Fortran Syntax（语法）：

Integer*4 function engOutputBuffer(ep,p)

Integer*4 ep

Character*n p

Arguments（参数）：

Ep:Engine pointer

P:pointer to character buffer

N:length of buffer p

Returns（返回值）：

1 if you pass it a NULL engine pointer. Otherwise, it returns 0.

2 mx 函数功能说明

Mx function 功能：操作 mxArray 矩阵（MX Array Manipulation）。

2.1 mxCreateFull 为旧的写法，现替换为 mxCreateDoubleMatrix：创建一个二维双精度矩阵。

Fortran Syntax（语法）：

Integer*4 function mxCreateDoubleMatrix(m, n, Complexflag)

Integer*4 m, n, Complexflag

Arguments（参数）：

m: The desired number of rows

n: The desired number of columns

Complexflag: If the data you plan to put into the mxArray has no imaginary components, specify 0. If the data has some imaginary components, specify 1.

Returns（返回值）:

A pointer to the created mxArray, if successful. If unsuccessful returns 0.

2.2 mxCopyReal8ToPtr:复制Fortran中双精度实型数组的值到矩阵。

Fortran Syntax（语法）:

```
subroutine mxCopyReal8ToPtr(y, px, n)
real*8 y(n)
integer*4 px, n
```

Arguments（参数）:

y: real*8 Fortran array

px: Pointer to the real or imaginary data of a double-precision Matlab array

n: Number of elements to copy.

Description:

mxCopyReal8ToPtr copies n REAL*8 values from the Fortran REAL*8 array y into the MATLAB array pointed to by px, either a pr or pi array.

2.3 mxCopyPtrToReal8:把Matlab中的矩阵传给Fortran中的双精度实型数组。

Fortran Syntax（语法）：

subroutine mxCopyPtrToReal8 (px, y, n)

real*8 y(n)

integer*4 px, n

Arguments（参数）：

y: real*8 Fortran array

px: Pointer to the real or imaginary data of a double-precision Matlab array

n: Number of elements to copy.

Description：

mxCopyPtrToReal8 copies n REAL*8 values from the MATLAB array pointed to by px, either a pr or pi array, into the Fortran REAL*8 array.

2.4 mxCopyComplex16ToPtr:复制Fortran中复型数组的值到矩阵。

Fortran Syntax（语法）：

subroutine mxCopyComplex16ToPtr(y, pr, pi, n)

complex*16 y(n)

integer*4 pr, pi, n

Arguments（参数）：

y: complex*16 Fortran array

pr: Pointer to the real data of a double-precision Matlab array

pi: Pointer to the imaginary data of a double-precision Matlab array

n: Number of elements to copy.

Description：

mxCopyComplex16ToPtr copies n COMPLEX*16 values from the Fortran COMPLEX*16 array y into the MATLAB arrays pointed to by pr and pi. This subroutine is essential for use with Fortran compilers that do not support the %VAL construct in order to set up standard Fortran arrays for passing as arguments to the computation routine of a MEX-file..

2.5 mxCopyPtrToComplex16:把 Matlab 中的矩阵传给 Fortran 中的复型数组。

Fortran Syntax（语法）：

subroutine mxCopyPtrToComplex16 (pr, pi, y, n)

complex*16 y(n)

integer*4 pr, pi, n

Arguments（参数）：

y: complex*16 Fortran array

pr: Pointer to the real data of a double-precision Matlab array

pi: Pointer to the imaginary data of a double-precision Matlab array

n: Number of elements to copy.

Description：

mxCopyPtrToComplex16 copies n COMPLEX*16 values from the MATLAB arrays pointed to by pr and pi into the Fortran COMPLEX*16 array y. This subroutine is essential for use with Fortran compilers that do not support the %VAL construct in

order to set up standard Fortran arrays for passing as arguments to the computation routine of a MEX-file.

2.6 mxGetM:Get number of rows in mxArray.

Fortran Syntax（语法）:

Integer*4 function mxGetM(pm)

Integer*4 pm

Arguments（参数）:

Pm: Pointer to an mxArray

Returns（返回值）:

The number of rows in the mxArray to which pm points.

2.7 mxGetN:Get number of columns in mxArray.

Fortran Syntax（语法）:

Integer*4 function mxGetN(pm)

Integer*4 pm

Arguments（参数）:

Pm: Pointer to an mxArray

Returns（返回值）:

The number of columns in the mxArray.

2.8 mxGetPr:得到 mxArray 中的实型数据。

Fortran Syntax（语法）：

Integer*4 function mxGetPr(pm)

Integer*4 pm

Arguments（参数）：

pm: Pointer to an mxArray

Returns（返回值）：

The address of the first element of the real data. Returns 0 if there is no real data. Once you have the starting address, you can access any other element in the mxArray.

2.9 mxGetPi:得到 mxArray 中的虚部数据。

Fortran Syntax（语法）：

Integer*4 function mxGetPi(pm)

Integer*4 pm

Arguments（参数）：

pm: Pointer to an mxArray

Returns（返回值）：

The imaginary data elements of the specified mxArray, on success. Returns 0 in Fortran if there is no imaginary data or if there is an error.

2.10 mxGetString: copy string mxArray into character array in Fortran.

Fortran Syntax（语法）:

integer*4 function mxGetString(pm, str, strlen)

integer*4 pm, strlen

character*(*) str

Arguments（参数）:

pm: Pointer to a string mxArray

str: The starting location into which the string should be written.

strlen: Maximum number of characters to read into str.

Returns（返回值）:

0 on success, and 1 on failure.

2.11 mxGetScalar: Get real component of first data element in mxArray.

Fortran Syntax（语法）:

real*8 function mxGetScalar(pm)

integer*4 pm

Arguments（参数）:

pm: Pointer to an mxArray; cannot be a cell mxArray, a structure mxArray, or an empty mxArray.

Returns（返回值）：

The value of the first real (nonimaginary) element of the mxArray.

2.12 mxCreateString: Create 1-by-N string mxArray initialized to specified string.

Fortran Syntax（语法）：

integer*4 function mxCreateString(str)

character*(*) str

Arguments（参数）：

str: The string that is to serve as the mxArray's initial data.

Returns（返回值）：

A pointer to the created string mxArray if successful, and 0 in Fortran otherwise.

2.13 mxIsNumeric: Determine whether mxArray is numeric.

Fortran Syntax（语法）：

Integer*4 function mxIsNumeric(pm)

Integer*4 pm

Arguments（参数）：

Pm: Pointer to an mxArray

Returns（返回值）：

Logical 1 (true) if the array can contain numeric data. Logical 0 (false) if the array cannot contain numeric data.

2.14 mxIsDouble: Determine whether mxArray represents data as double-precision, floating-point numbers.

Fortran Syntax（语法）：

Integer*4 function mxIsDouble(pm)

Integer*4 pm

Arguments（参数）：

Pm: Pointer to an mxArray

Returns（返回值）：

Logical 1 (true) if the mxArray stores its data as double-precision, floating-point numbers, and logical 0 (false) otherwise.

2.15 mxIsComplex: Determine whether data is complex.

Fortran Syntax（语法）：

Integer*4 function mxIsComplex(pm)

integer*4 pm

Arguments（参数）：

pm: Pointer to an mxArray

Return：

Logical 1 (true) if pm is a numeric array containing complex data, and logical 0 (false) otherwise. If pm points to a cell array or a structure array, mxIsComplex returns false.

2.16 mxIsChar: Determine whether input is string mxArray.

Fortran Syntax（语法）：

Integer*4 function mxIsChar(pm)
integer*4 pm

Arguments（参数）：

pm: Pointer to an mxArray

Return：

Logical 1 (true) if pm points to an array having the class mxCHAR_CLASS, and logical 0 (false) otherwise.

2.17 mxDestroyArray:释放了由mxCreate函数创建的动态内存。

Fortran Syntax（语法）：

subroutine mxDestroyArray(pm)

integer*4 pm

Arguments（参数）：

pm: Pointer to the mxArray you want to free

Description：

mxDestroyArray deallocates the memory occupied by the specified mxArray. mxDestroyArray not only deallocates the memory occupied by the mxArray's characteristics fields (such as m and n), but also deallocates all the mxArray's associated data arrays.

3 mex 函数功能说明

Mex function：即编写 MEX 文件的 API 函数。

3.1 mexFunction: MEX 文件的入口函数，当在 MATLAB 命令行中执行 MEX 函数时，MATLAB 解释器将从此函数处开始执行 MEX 代码。

Fortran Syntax（语法）：

mexFunction(nlhs, plhs, nrhs, prhs)

integer*4 nlhs, nrhs

integer*4 plhs(*), prhs(*)

Arguments（参数）：

prhs，输入参数的 mxArray 类型指针；

nrhs，输入参数个数；

plhs，输出参数的 mxArray 类型指针；

nlhs，输出参数个数。

Returns（返回值）：

mexFunction is not a routine you call. Rather, mexFunction is the name of a subroutine in Fortran that you must write in every MEX-file. When you invoke a MEX-function, MATLAB® software finds and loads the corresponding MEX-file of the same name. MATLAB then searches for a symbol named mexFunction within the MEX-file. If it finds one, it calls the MEX-function using the address of the mexFunction symbol. If MATLAB cannot find a routine named mexFunction inside the MEX-file, it issues an error message.

3.2 mexErrMsgTxt: Issue error message and return to Matlab prompt.

Fortran Syntax（语法）：

mexErrMsgTxt(errormsg)

character*(*) errormsg

Arguments（参数）：

Errormsg: String containing the error message to be displayed

Returns（返回值）：

Call mexErrMsgTxt to write an error message to the MATLAB window. After the error message prints, MATLAB terminates the MEX-file and returns control to the MATLAB prompt.

Calling mexErrMsgTxt does not clear the MEX-file from memory. Consequently, mexErrMsgTxt does not invoke the function registered through mexAtExit.

If your application called mxCalloc or one of the mxCreate* routines to allocate memory, mexErrMsgTxt automatically frees the allocated memory.

In addition to the errormsg, the mexerrmsgtxt function determines where the error occurred, and displays the following information. If an error labeled Print my error message occurs in the function foo, mexerrmsgtxt displays:

??? Error using ==> foo

Print my error message

3.3 mexPrintf: Prints a string on the screen and in the diary (if the diary is in use).

Fortran Syntax（语法）：

integer*4 mexPrintf(message)

character*(*) message

Arguments（参数）：

message: String to be displayed.

Returns（返回值）：

The number of characters printed. This includes characters specified with backslash codes, such as \n and \b.

3.4 mexAtExit: Register function to call when MEX-function is cleared or MATLAB software terminates.

mexAtExit 函数可以用来向 Matlab 注册函数，注册的函数将在 MEX 文件从内存中清除时被自动调用，从而可以完成一些清除内存，释放空间的操作。

Fortran Syntax（语法）：

integer*4 function mexAtExit(ExitFcn)

subroutine ExitFcn

Arguments（参数）：

ExitFcn: Pointer to function you want to run on exit.

Returns（返回值）：

Always returns 0. In Fortran, you must declare the ExitFcn as external in the Fortran routine that calls mexAtExit if it is not within the scope of the file.

3.5 mexIsLocked: Determine whether MEX-file is locked.

Fortran Syntax（语法）：

integer*4 function mexIsLocked()

Returns（返回值）：

Logical 1 (true) if the MEX-file is locked; logical 0 (false) if the file is unlocked.

3.6 mexLock: 锁定函数。Prevent MEX-file from being cleared from memory.

Fortran Syntax（语法）：

mexLock()

Description：

By default, MEX-files are unlocked, meaning that a user can clear them at any time. Call mexLock to prohibit a MEX-file from being cleared.

3.7 mexUnlock: 解锁函数。Allow MEX-file to be cleared from memory.

Fortran Syntax（语法）：

mexUnlock()

Description：

By default, MEX-files are unlocked, meaning that a user can clear them at any time.

3.8 mexCallMATLAB: Call MATLAB function or user-defined M-file or MEX-file.

Fortran Syntax（语法）：

mexCallMATLAB(nlhs, plhs, nrhs, prhs, name)

integer*4 nlhs, nrhs

integer*4 plhs(*), prhs(*)

character*(*) name

Arguments（参数）：

prhs，输入参数的 mxArray 类型指针；

nrhs，输入参数个数；

plhs，输出参数的 mxArray 类型指针；

nlhs，输出参数个数。

name, Character string containing the name of the MATLAB built-in, operator, M-file, or MEX-file that you are calling.

★ 注意：这里的 nrhs,prhs,nlhs,plhs 是真对 name 函数（或 m 文件）的输入输出参数，而不是整个接口函数 mexFunction 的输入输出参数。

Returns（返回值）：

0 if successful, and a nonzero value if unsuccessful.

附

Function	Successful	Unsuccessful
engOpen	非 0	0
engClose	0	1
engEvalString	0	非 0
engPutVariable	0	1
engGetVariable	非 0	0
mxCreateDoubleMatrix	非 0	0
mxGetPr	非 0	0