[VS2008 与 Matlab 的混合编程(quadprog函数为例)](http://blog.csdn.net/jiabingxi/article/details/17633063)

本文简单介绍VS2008与Matlab的混合编程(quadprog函数为例)，C/C++与Matlab的混合编程有多种方式，这里只介绍采用动态链接库的方式。

附件：[M文件及生成的DLL](http://www.yongblog.com/wp-content/uploads/2011/04/QAP.rar" \t "_blank) | [Matlab\_C源代码](http://www.yongblog.com/wp-content/uploads/2011/04/Matlab_C_src.rar" \t "_blank) | [意大利人的QuadProg++](http://www.yongblog.com/wp-content/uploads/2011/04/QuadProg.rar)

众所周知Matlab的强大运算能力让各种工程软件望而却步，而VC程序的友好界面又让人爱不释手，如果能够让两者的优势结合起来的话，势必能让程序员欢欣鼓舞。还好MathWorks已经为我们想到了，基本上现在市面上主流的Matlab版本都可以支持VC和Matlab的混合编程。但是目前网上资料基本上都是基于Matlab 6.0和VC++ 6.0的，和新版本的实现方法有所不同，由于笔者在实现论文算法的时候要用到求解二次规划问题(QAP)，经过大约几天的摸索，初步还是决定采用Matlab优化工具箱中的quadprog来解决，毕竟基于C/C++的求解二次规划问题的工具程序包极为稀少(目前在网上也只能找到一位意大利人写的Quadprog++)，这是由于Quadprog++没有尝试过，故所能解决问题的规模（论文中要求的二次规划问题变量个数上千个，限制条件至少也有上百个，甚至上千个）并不能得到保证，而Matlab经过大量的实践，优化工具箱的效果非常好！言归正传，下面简单介绍介绍VS2008与Matlab的混合编程：  
为了更加直观的说明如何实现混合编程，笔者写了一个求解二次规划的程序作为测试。先在

Matlab里面写一个M文件，代码如下：  
function [x, fval] = QUADASSINPROG(H, C, A, b, lb)  
[x, fval] = quadprog(H, C, A, b, [], [], lb);  
将M文件保存为QUADASSINPROG.m，保存在F:/test文件夹中。然后在Matlab主窗口中将当前文件夹改为F:/test：  
cd F:/test;

为了能够让VC调用Matlab程序，我们可以采用将Matlab程序包装成动态链接文件DLL的形式，然后让VC调用动态链接文件，因此在Matlab中，我们需要设置一下Matlab的编译库，在Matlab主窗口中键入如下代码：  
mbuild -setup  
然后出现  
Please choose your compiler for building standalone MATLAB applications:

Would you like mbuild to locate installed compilers [y]/n?

键入y后回车，出现  
Select a compiler:

[1] Lcc-win32 C 2.4.1 in D:\MATLAB\R2010b\sys\lcc

[2] Microsoft Visual C++ 2008 SP1 in c:\Program Files\Microsoft Visual Studio 9.0

[3] Microsoft Visual C++ 6.0 in C:\Program Files\Microsoft Visual Studio

[0] None

Compiler:

由于我们是需要用VC来调用Matlab函数，因此我们选择2，回车，出现

Please verify your choices:

Compiler: Microsoft Visual C++ 2008 SP1

Location: c:\Program Files\Microsoft Visual Studio 9.0

Are these correct [y]/n?

确定，键入y后回车，出现

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Warning: Applications/components generated using Microsoft Visual Studio

2008 require that the Microsoft Visual Studio 2008 run-time

libraries be available on the computer used for deployment.

To redistribute your applications/components, be sure that the

deployment machine has these run-time libraries.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Trying to update options file:

C:\Users\Yong!\AppData\Roaming\MathWorks\MATLAB\R2010b\compopts.bat

From template:              D:\MATLAB\R2010b\bin\win32\mbuildopts\msvc90compp.bat

Done . . .

如果出现以上说明，则证明Matlab编译器设置成功了。接下来是生成M文件的DLL文件。在Matlab主窗口中键入如下代码：  
mcc -W cpplib:QuadAssinProg -T link:lib QUADASSINPROG.m -C  
其中cpplib:后面的是需要生成文件的文件名，是自己取的，link:lib后面的QUADASSINPROG.m是转换为DLL的M文件的文件名。 -W/-T/-C是参数，具体含义可以通过mcc –help命令查看，注意参数的大小写。Matlab编译后会在test文件夹下生成9个文件，分别是：

mccExcludedFiles.log

QuadAssinProg.cpp

QuadAssinProg.ctf

QuadAssinProg.dll

QuadAssinProg.exp

QuadAssinProg.exports

QuadAssinProg.h

QuadAssinProg.lib

readme.txt

其中QuadAssinProg.dll，QuadAssinProg.lib，QuadAssinProg.ctf，QuadAssinProg.h是我们需要的文件，至此，Matlab方面就已经设置妥当了。

打开VC2008，新建一个工程，然后将上述四个文件复制到工程目录下。选择“工具-选项-项目和解决方案-VC++目录”，在“显示以下内容目录”里选择“包含文件”将“D:\MATLAB\R2010b\extern\include”添加进去，然后选择“库文件”将“D:\MATLAB\R2010b\extern\lib\win32\microsoft”添加进去，其中D:是Matlab的安装目录，可根据实际情况进行修改。

然后选择“项目-属性-配置属性-链接器-输入”，在右边“附加依赖项”一栏填入“QuadAssinProg.lib mclmcrrt.lib”，其中QuadAssinProg.lib就是刚刚用Matlab生成的lib文件。

在编写代码时首先得包含头文件“QuadAssinProg.h”，然后开始编写需要的代码，为了说明问题编

写了一个测试代码如下所示：

int \_tmain(int argc, TCHAR\* argv[], TCHAR\* envp[])

{

QuadAssinProgInitialize(); //初始化DLL动态连接文件

//输入矩阵、数组  
double \_H[2][2] = {{1, -1}, {-1, 2}};  
double \_C[2] = {-6, -2};  
double \_A[3][2] = {{2, 1}, {-1, 1}, {1, 2}};  
double \_b[3] = {3, 1, 2};  
double \_lb[2] = {0, 0};

//输出矩阵、数组  
double \_x[2];  
double \_fval[1];

//为QUADASSINPROG函数创建输入参数矩阵  
mxArray\* Input\_H = mxCreateDoubleMatrix(2, 2, mxREAL);  
//将输入矩阵、数组拷贝至QUADASSINPROG输入参数  
memcpy(mxGetPr(Input\_H), (void\*)\_H, sizeof(\_H));

mxArray\* Input\_C = mxCreateDoubleMatrix(1, 2, mxREAL);  
memcpy(mxGetPr(Input\_C), (void\*)\_C, sizeof(\_C));

mxArray\* Input\_A = mxCreateDoubleMatrix(3, 2, mxREAL);  
memcpy(mxGetPr(Input\_A), (void\*)\_A, sizeof(\_A));

mxArray\* Input\_b = mxCreateDoubleMatrix(1, 3, mxREAL);  
memcpy(mxGetPr(Input\_b), (void\*)\_b, sizeof(\_b));

mxArray\* Input\_lb = mxCreateDoubleMatrix(1, 2, mxREAL);  
memcpy(mxGetPr(Input\_lb), (void\*)\_lb, sizeof(\_lb));

mxArray \*Input[5] = {Input\_H, Input\_C, Input\_A, Input\_b, Input\_lb};  
mxArray \*Output[2];

//调用Matlab中的函数 function [x, fval] = QUADASSINPROG(H, C, A, b, lb)  
mlxQUADASSINPROG(2, Output, 5, Input);

//为QUADASSINPROG函数创建输出参数指针  
mxArray \*x;  
mxArray \*fval;

//将输出参数传递给输出参数指针  
x = Output[0];  
fval = Output[1];

memcpy(\_x, mxGetPr(x), sizeof(\_x));  
memcpy(\_fval, mxGetPr(fval), sizeof(\_fval));

QuadAssinProgTerminate(); //结束DLL库

CString resultStr;  
resultStr.Format(“%f %f %f”, \_x[0], \_x[1], \_fval[0]);

cout<<resultStr<<endl;

char anyKey;  
cin>>anyKey;

return 0;  
}  
其中QuadAssinProgInitialize()，mlxQUADASSINPROG()，QuadAssinProgTerminate()等函数可以在QuadAssinProg.h中找到函数原型，mlxQUADASSINPROG函数原型为：  
mlxQUADASSINPROG(int nlhs, mxArray \*plhs[], int nrhs, mxArray \*prhs[]);

nlhs代表输出参数个数，plhs[]里面存放的是输出参数的指针，nrhs代表输入参数个数，prhs[]中存放的是输入参数的指针。

上例中由于输入输出参数只有一个，因此在plhs[0]中存放的就是M文件中b的指针，prhs[0]中存放的就是M文件中a的指针。编译运行之后就可以在控制台中打印所求得的结果。将项目文件夹下生成的可执行文件复制出来，将QuadAssinProg.dll文件、QuadAssinProg.ctf文件和.exe文件放在同一目录下便可脱离VC++独立运行。如果需要在其他未装Matlab的机器上运行，则还需要将“D:\MATLAB\R2010b\toolbox\compiler\deploy\win32”中的MCRInstaller.exe复制出来，在未安装Matlab的电脑上运行此程序，才可使得刚刚生成的可执行文件顺利运行。

以上程序在Windows 7 旗舰版 + Visual Studio 2008 专业版 SP1 + Matlab R2010b下编译运行通过。

本文链接： <http://www.yongblog.com/archives/202.html> 转载请注明出处。

参考链接： <http://blog.sina.com.cn/s/blog_49cb42490100nm5j.html>