<http://blog.csdn.net/caoenze/article/details/46699327>

<http://blog.csdn.net/zhujiahui622/article/details/50504678>

<http://blog.csdn.net/zhujiahui622/article/details/50504164>

<http://www.cnblogs.com/darkknightzh/p/5578027.html>

# [Visual Studio下实现Eigen+Intel MKL的矩阵乘法](http://blog.csdn.net/zhujiahui622/article/details/50504678)

标签： [Eigen](http://www.csdn.net/tag/Eigen" \t "http://blog.csdn.net/zhujiahui622/article/details/_blank)[Intel-MKL](http://www.csdn.net/tag/Intel-MKL" \t "http://blog.csdn.net/zhujiahui622/article/details/_blank)[矩阵乘法](http://www.csdn.net/tag/%e7%9f%a9%e9%98%b5%e4%b9%98%e6%b3%95" \t "http://blog.csdn.net/zhujiahui622/article/details/_blank)[visual studio](http://www.csdn.net/tag/visual studio" \t "http://blog.csdn.net/zhujiahui622/article/details/_blank)

2016-01-12 15:45 2053人阅读 [评论](http://blog.csdn.net/zhujiahui622/article/details/50504678" \l "comments)(7) [收藏](http://blog.csdn.net/zhujiahui622/article/details/javascript:void(0);" \o "收藏" \t "http://blog.csdn.net/zhujiahui622/article/details/_blank) [举报](http://blog.csdn.net/zhujiahui622/article/details/50504678" \l "report" \o "举报)

IMG_256 分类：

项目配置（1） IMG_257

版权声明：本文为博主原创文章，未经博主允许不得转载。

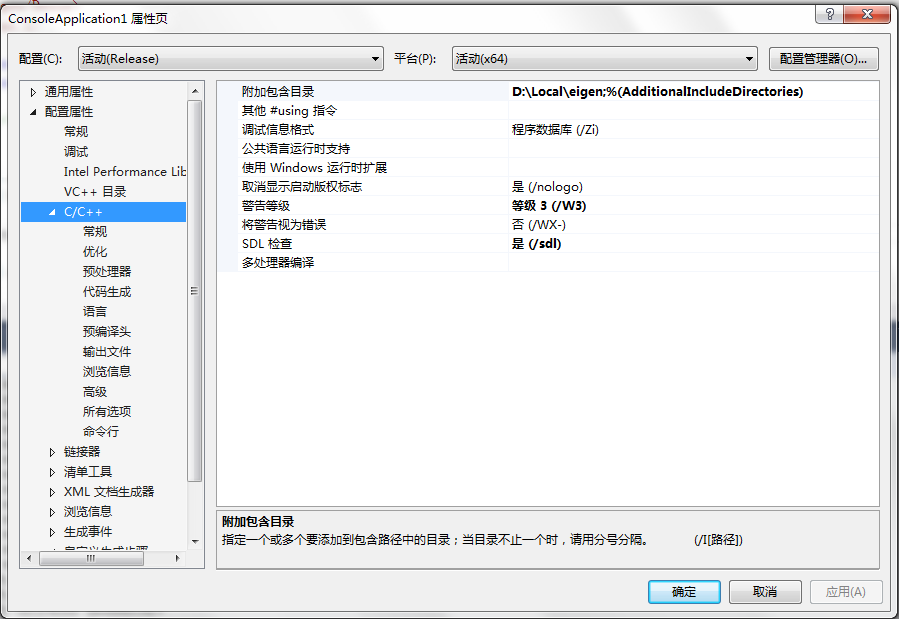
科学计算过程中经常会用到线性代数运算，尤其是矩阵乘法，自己手写的C++代码想要赶上MATLAB几乎是不可能的。在C++中，谈到高性能计算就往往要用Intel MKL，但是用过的人都知道，这东西烦人的很，函数风格也怪异，不好用，一个简单的矩阵乘法函数dgemm的参数列表就老长老长。Eigen是一个相对来说十分优雅和好用的线性代数运算库（尤其是在Windows下，Blas，Lapack太麻烦，基本不考虑），缺点就是密集型计算“不那么快”。

所以，如果既能有Intel MKL的速度，又能有Eigen的优雅，还是在Windows下，那真真是极好的。

**[操作系统](http://lib.csdn.net/base/operatingsystem" \o "操作系统知识库" \t "http://blog.csdn.net/zhujiahui622/article/details/_blank)**：Windows 7 64位

安装Visual Studio 2013   
下载C++下的一个线性代数运算库Eigen并解压：[http://eigen.tuxfamily.org/index.php?title=Main\_Page](http://eigen.tuxfamily.org/index.php?title=Main_Page" \t "http://blog.csdn.net/zhujiahui622/article/details/_blank)   
安装Intel Parallel Studio XE 2015

在Visual Studio下配置Intel MKL：详见 [http://blog.csdn.net/zhujiahui622/article/details/50504164](http://blog.csdn.net/zhujiahui622/article/details/50504164" \t "http://blog.csdn.net/zhujiahui622/article/details/_blank)

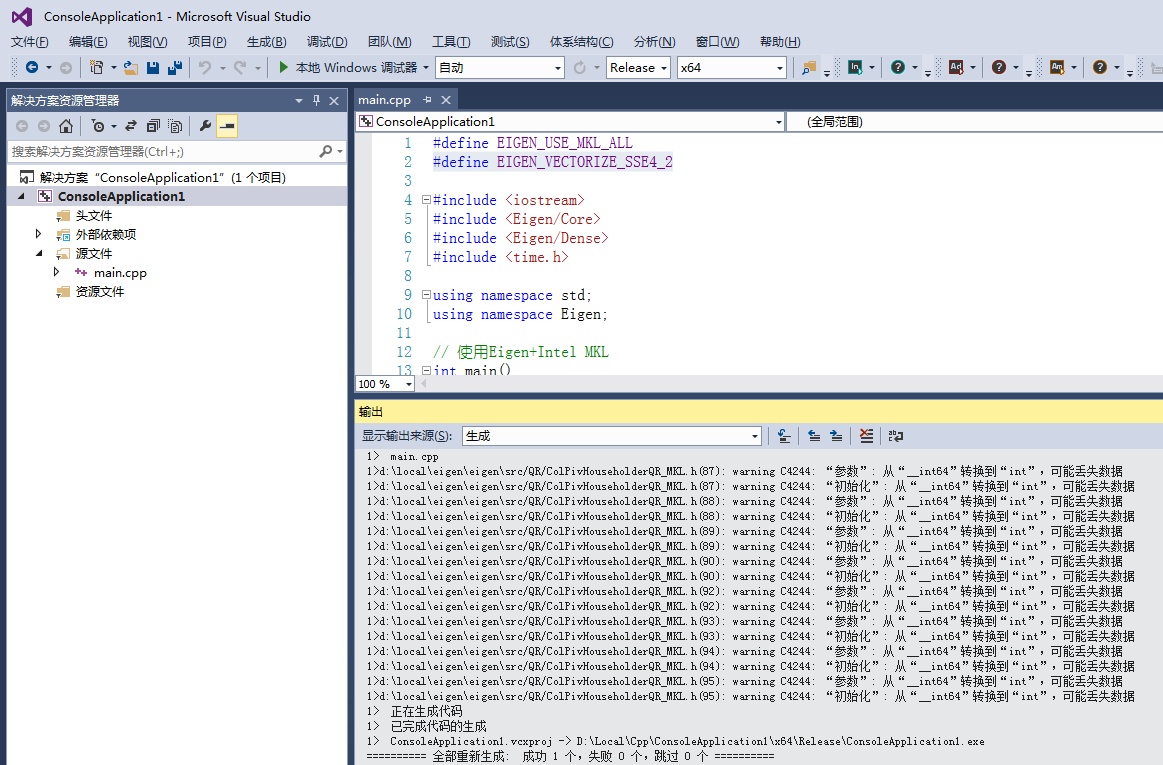
在Visual Studio下配置Eigen。假设Eigen解压在D:\Local\eigen下，则在项目属性中的“**配置属性**”下的“**C/C++**”面板中将“**附加包含目录**”设置为D:\Local\eigen   


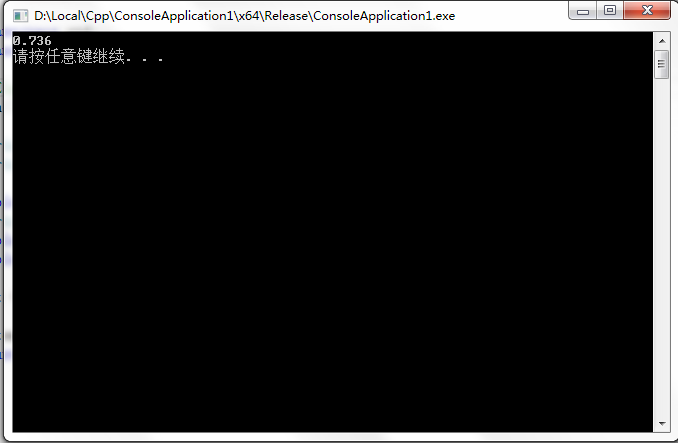
写代码

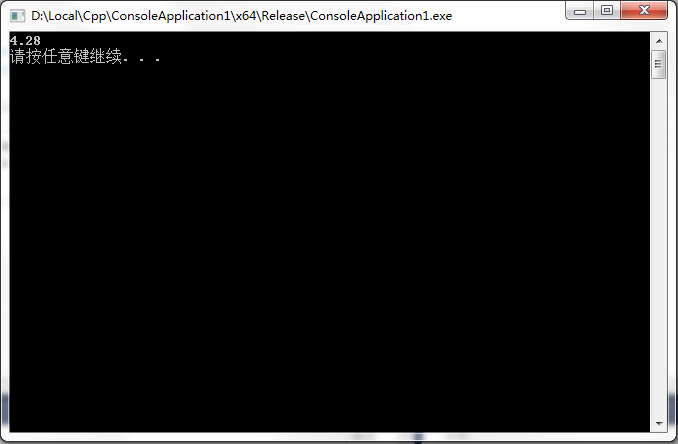
#define EIGEN\_USE\_MKL\_ALL #define EIGEN\_VECTORIZE\_SSE4\_2 #include <iostream> #include <Eigen/Core> #include <Eigen/Dense> #include <time.h> using namespace std; using namespace Eigen; // 使用Eigen+Intel MKL int main(int argc, char \*argv[]) { MatrixXd a = MatrixXd::Random(3000, 3000); // 随机初始化矩阵 MatrixXd b = MatrixXd::Random(3000, 3000); double start = clock(); MatrixXd c = a \* b; // 乘法好简洁 double endd = clock(); double thisTime = (double)(endd - start) / CLOCKS\_PER\_SEC; cout << thisTime << endl; system("PAUSE"); return 0; }

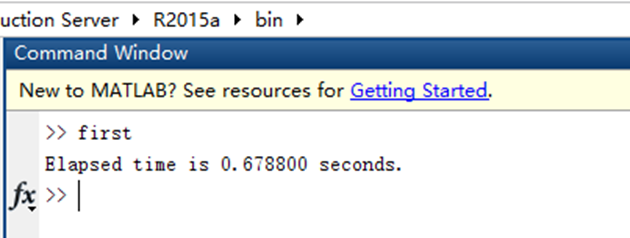
#define EIGEN\_USE\_MKL\_ALL”这句宏定义就表示在Eigen上用了Intel MKL进行计算优化，当然，其他头文件也是不能少的。

“#define EIGEN\_VECTORIZE\_SSE4\_2”这句表示采用SSE4.2浮点数优化，不用也可。

选择“Release”，“x64”模式进行生成（编译）   


计算两个3000\*3000的double型密集矩阵相乘   
使用Eigen+Intel MKL的运行时间：0.736秒！！！已经相当快了   


不用Intel MKL，只用Eigen，运行时间：4.28秒   


MATLAB下运行时间：0.6788秒。果然还是无法超越，只是差不多能赶上它了   


顶

0

# [Visual Studio 2013配置Intel MKL](http://blog.csdn.net/zhujiahui622/article/details/50504164)

标签： [visual studio](http://www.csdn.net/tag/visual studio" \t "http://blog.csdn.net/zhujiahui622/article/details/_blank)[Intel](http://www.csdn.net/tag/Intel" \t "http://blog.csdn.net/zhujiahui622/article/details/_blank)[MKL](http://www.csdn.net/tag/MKL" \t "http://blog.csdn.net/zhujiahui622/article/details/_blank)

2016-01-12 14:48 2905人阅读 [评论](http://blog.csdn.net/zhujiahui622/article/details/50504164" \l "comments)(0) [收藏](http://blog.csdn.net/zhujiahui622/article/details/javascript:void(0);" \o "收藏" \t "http://blog.csdn.net/zhujiahui622/article/details/_blank) [举报](http://blog.csdn.net/zhujiahui622/article/details/50504164" \l "report" \o "举报)

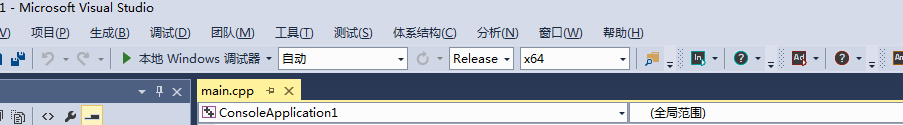
IMG_256 分类：

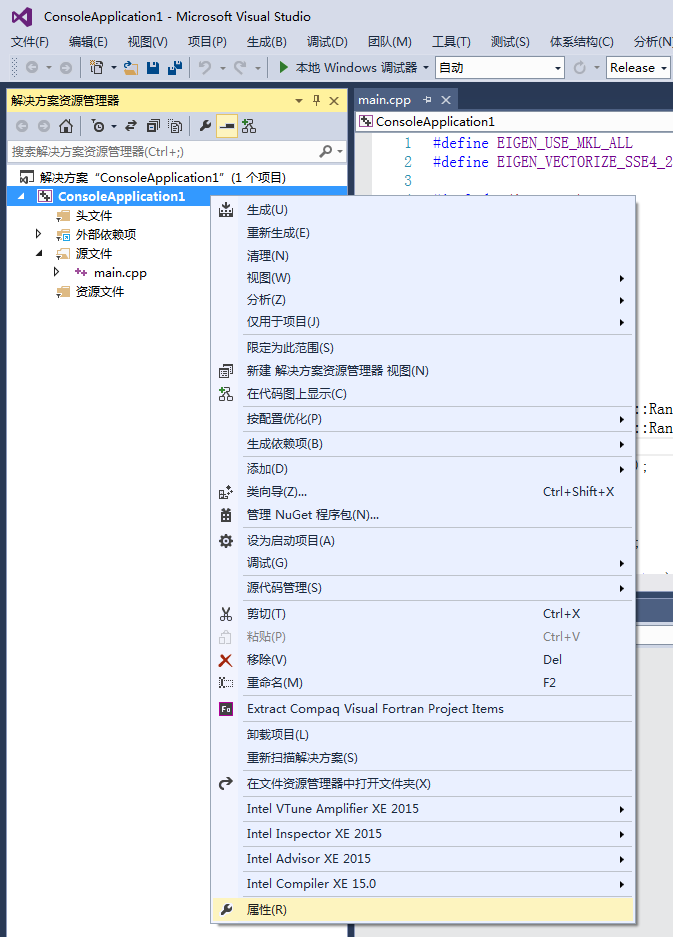
项目配置（1） IMG_257

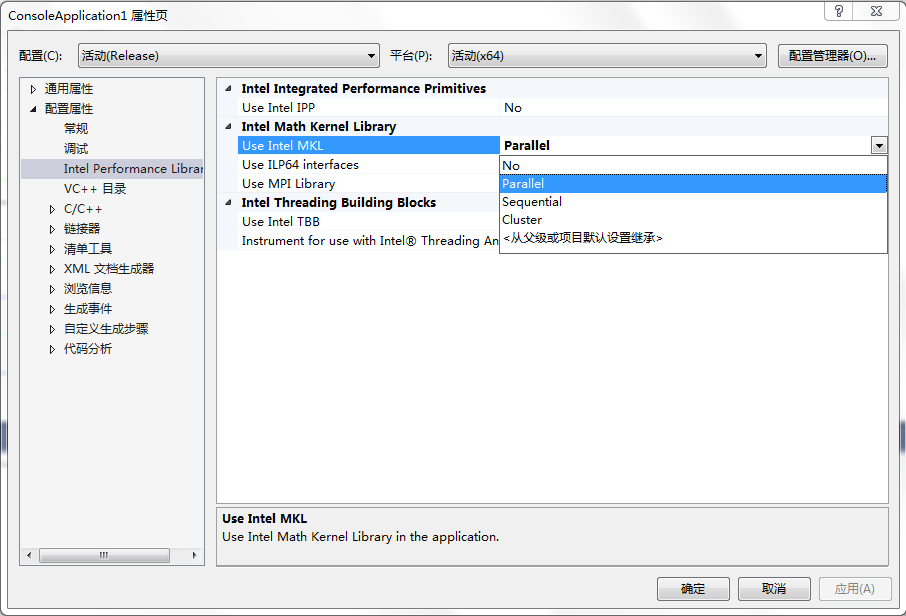
版权声明：本文为博主原创文章，未经博主允许不得转载。

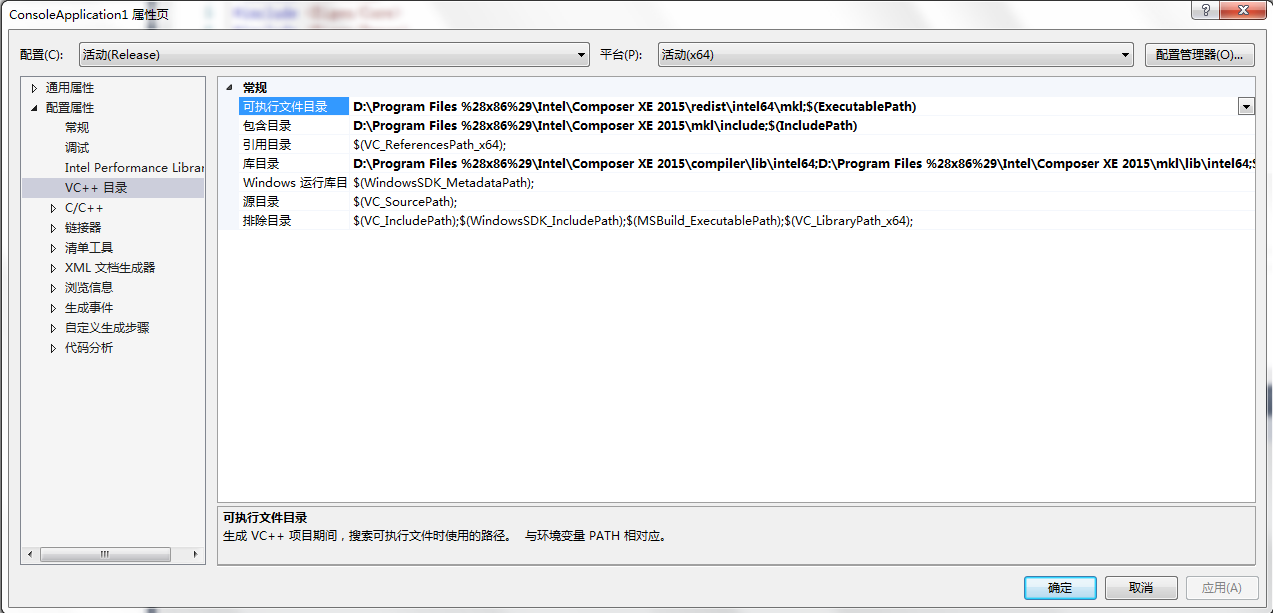
**[操作系统](http://lib.csdn.net/base/operatingsystem" \o "操作系统知识库" \t "http://blog.csdn.net/zhujiahui622/article/details/_blank)**：Windows 7 64位   
IDE：Visual Studio 2013   
MKL：安装Intel Parallel Studio XE 2015 64位

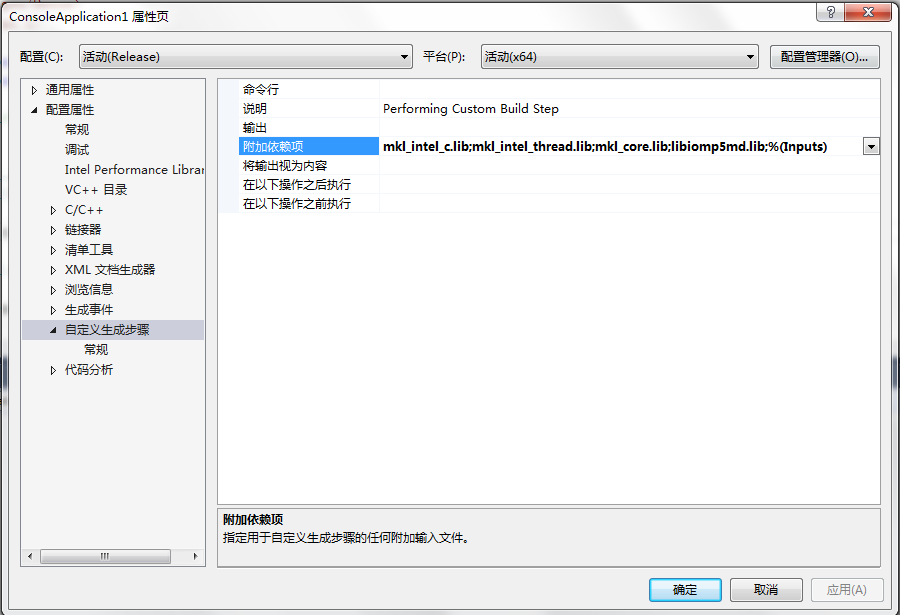
先安装Visual Studio 2013，后安装Intel Parallel Studio XE 2015。   
假设Intel Parallel Studio XE 2015安装在**D:\Program Files (x86)\Intel**下

新建一个解决方案和项目。解决方案设置为“**Release**”和“**x64**”模式。如果是32位就是”**x86**“，如果想用编译调试模式，就选择“**Debug**”。   


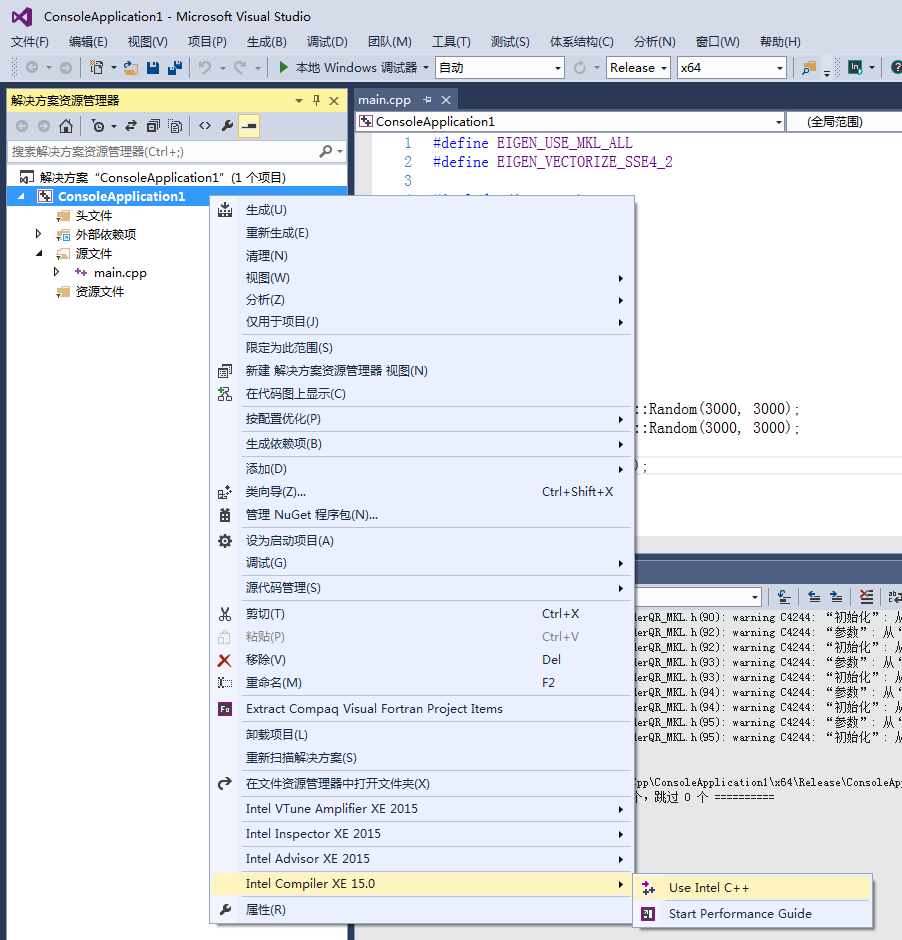
选中项目，右键选择“**属性**”   


在“**配置属性**”下的“**Intel Performance Libraries**”面板上的“**Use Intel MKL**”条目的右边的下拉框选择“**Parallel**”   


在“**VC++目录**”下设置以下：   
**可执行文件目录**为：D:\Program Files (x86)\Intel\Composer XE 2015\redist\intel64\mkl   
**包含目录**为：D:\Program Files (x86)\Intel\Composer XE 2015\mkl\include   
**库目录**为：D:\Program Files (x86)\Intel\Composer XE 2015\compiler\lib\intel64;D:\Program Files (x86)\Intel\Composer XE 2015\mkl\lib\intel64   
**注意**：如果是32位的，就不是“**intel 64**”而是“**ia32**”   


在“**自定义生成步骤**”下，设置“**附加依赖项**”，选择以下lib文件，缺少什么可能需要尝试以下，一般是这几个：   
mkl\_intel\_c.lib   
mkl\_intel\_thread.lib   
mkl\_core.lib   
libiomp5md.lib   


设置完之后确定就可以了

PS：先安装Visual Studio，后安装Intel Parallel Studio就会在Visual Studio里面安装Intel C++编译器，在生成解决方案和项目的时候可以选择使用Intel的C++编译器。这两者可以互相切换。   


对于Visual Studio 2013以后版本的IDE也是类似的。

# [Intel MKL 在VS中的配置与安装笔记](http://blog.csdn.net/caoenze/article/details/46699327)

标签： [MKL](http://www.csdn.net/tag/MKL" \t "http://blog.csdn.net/caoenze/article/details/_blank)[vs](http://www.csdn.net/tag/vs" \t "http://blog.csdn.net/caoenze/article/details/_blank)[配置与安装](http://www.csdn.net/tag/%e9%85%8d%e7%bd%ae%e4%b8%8e%e5%ae%89%e8%a3%85" \t "http://blog.csdn.net/caoenze/article/details/_blank)

2015-06-30 19:27 2415人阅读 [评论](http://blog.csdn.net/caoenze/article/details/46699327" \l "comments)(0) [收藏](http://blog.csdn.net/caoenze/article/details/javascript:void(0);" \o "收藏" \t "http://blog.csdn.net/caoenze/article/details/_blank) [举报](http://blog.csdn.net/caoenze/article/details/46699327" \l "report" \o "举报)

IMG_256 分类：

论文笔记（1） IMG_257

版权声明：本文为博主原创文章，未经博主允许不得转载。

## Intel MKL 在VS中的配置与安装笔记

mkl 使用手册下载：[http://download.csdn.net/detail/caoenze/8855821](http://download.csdn.net/detail/caoenze/8855821" \t "http://blog.csdn.net/caoenze/article/details/_blank)

1. 从intel官网下载c\_studio\_xe\_2013\_sp1\_update3\_setup.exe文件（完全离线安装包）
2. 双击.exe文件，自动提取文件并进入安装引导
3. 安装完成后，配置VS2010（前提是本机已正确安装过VS2010）
4. 新建一C++项目，比如win32控制台项目：MKL\_TEST
5. 点击菜单栏 项目——》MKL\_TEST属性——》配置属性——》VC++目录：   
   可执行文件目录添加：C:\Program Files (x86)\Intel\Composer XE\mkl\bin\ia32   
   包含目录添加：C:\Program Files (x86)\Intel\Composer XE\mkl\include   
   库目录添加：C:\Program Files (x86)\Intel\Composer XE\mkl\lib\ia32   
   注意：包含目录不区分ia32和intel64   
   Bin和lib目录区分ia32和intel64根据自己的CPU架构选择。   
   IA32可以认为是X86或者X86-32   
   Intel64:intel与HP联合开发的64-bits全新架构，与X86不兼容，没有太大市场。   
   6 、连接器——》输入   
   附加依赖项：添加   
   mkl\_intel\_c.lib   
   mkl\_intel\_thread.lib   
   mkl\_core.lib   
   libiomp5mt.lib//我只添加了前三个，添加第4个，编译时提示找不到此库   
   7、配置属性——Intel Performance Library   
   右侧Use Intel MKL :   
   选择Parallel   
   其它两项可以选择性配置，不配置也可以。   
   8、至此，VS2010调用MKL已配置完毕，可在MKL\_TEST项目里添加源文件main.c 测试代码如下：

#define min(x,y) (((x) < (y)) ? (x) : (y)) #include <stdio.h> #include <stdlib.h> #include "mkl.h" int main() { double \*A, \*B, \*C; int m, n, k, i, j; double alpha, beta; printf ("\n This example computes real matrix C=alpha\*A\*B+beta\*C using \n" " Intel庐 MKL function dgemm, where A, B, and C are matrices and \n" " alpha and beta are double precision scalars\n\n"); m = 2000, k = 200, n = 1000; printf (" Initializing data for matrix multiplication C=A\*B for matrix \n" " A(%ix%i) and matrix B(%ix%i)\n\n", m, k, k, n); alpha = 1.0; beta = 0.0; printf (" Allocating memory for matrices aligned on 64-byte boundary for better \n" " performance \n\n"); A = (double \*)mkl\_malloc( m\*k\*sizeof( double ), 64 ); B = (double \*)mkl\_malloc( k\*n\*sizeof( double ), 64 ); C = (double \*)mkl\_malloc( m\*n\*sizeof( double ), 64 ); if (A == NULL || B == NULL || C == NULL) { printf( "\n ERROR: Can't allocate memory for matrices. Aborting... \n\n"); mkl\_free(A); mkl\_free(B); mkl\_free(C); return 1; } printf (" Intializing matrix data \n\n"); for (i = 0; i < (m\*k); i++) { A[i] = (double)(i+1); } for (i = 0; i < (k\*n); i++) { B[i] = (double)(-i-1); } for (i = 0; i < (m\*n); i++) { C[i] = 0.0; } printf (" Computing matrix product using Intel庐 MKL dgemm function via CBLAS interface \n\n"); cblas\_dgemm(CblasRowMajor, CblasNoTrans, CblasNoTrans, m, n, k, alpha, A, k, B, n, beta, C, n); printf ("\n Computations completed.\n\n"); printf (" Top left corner of matrix A: \n"); for (i=0; i<min(m,6); i++) { for (j=0; j<min(k,6); j++) { printf ("%12.0f", A[j+i\*k]); } printf ("\n"); } printf ("\n Top left corner of matrix B: \n"); for (i=0; i<min(k,6); i++) { for (j=0; j<min(n,6); j++) { printf ("%12.0f", B[j+i\*n]); } printf ("\n"); } printf ("\n Top left corner of matrix C: \n"); for (i=0; i<min(m,6); i++) { for (j=0; j<min(n,6); j++) { printf ("%12.5G", C[j+i\*n]); } printf ("\n"); } getchar(); printf ("\n Deallocating memory \n\n"); mkl\_free(A); mkl\_free(B); mkl\_free(C); printf (" Example completed. \n\n"); return 0; }

## [（原）使用mkl中函数LAPACKE\_sgesv计算矩阵的逆矩阵](http://www.cnblogs.com/darkknightzh/p/5578027.html)

转载请注明出处：

[http://www.cnblogs.com/darkknightzh/p/5578027.html](http://www.cnblogs.com/darkknightzh/p/5578027.html" \t "http://www.cnblogs.com/darkknightzh/p/_blank)

参考文档：mkl的说明文档

lapack\_int LAPACKE\_sgesv(int matrix\_layout, lapack\_int n, lapack\_int nrhs, float \* a, lapack\_int lda, lapack\_int \* ipiv, float \* b, lapack\_int ldb);

该函数计算AX=B的解。简单来说，当B为单位矩阵时，X即为inv(A)。

**输入：**

matrix\_layout：矩阵存储顺序，C++中为行优先LAPACK\_ROW\_MAJOR

n：线性方程的个数，n≥0n≥0

nrhs：矩阵B的列数，nrhs≥0nrhs≥0

a：大小max(1, lda\*n)，包含n\*n的系数矩阵A

b：列优先存储时，大小max(1, ldb\* nrhs)；行优先存储时，大小max(1, ldb\*n)；包含n\* nrhs的矩阵B

lda：矩阵a的leading dimension，lda≥max(1,n)lda≥max(1,n)

ldb：矩阵b的leading dimension；列优先存储时，ldb≥max(1,n)ldb≥max(1,n)；行优先存储时，ldb≥max(1,nrhs)ldb≥max(1,nrhs)

**输出：**

a：（具体看说明文档）可能会被覆盖

b：（具体看说明文档）调用此函数时被覆盖

ippv：（具体看说明文档）大小最小是max(1, n)

**返回值：**0成功，其他不成功

This function returns a value info.

If info=0, the execution is successful.

If info = -i, parameter i had an illegal value.

If info = i, Ui,iUi,i (computed in double precision for mixed precision subroutines) is exactly zero. The factorization has been completed, but the factor U is exactly singular, so the solution could not be computed.

**程序：**

int SCalcInverseMatrix(float\* pDst, const float\* pSrc, int dim)

{

int nRetVal = 0;

if (pSrc == pDst)

{

return -1;

}

int\* ipiv = new int[dim \* dim];

float\* pSrcBak = new float[dim \* dim]; // LAPACKE\_sgesv会覆盖A矩阵，因而将pSrc备份

memcpy(pSrcBak, pSrc, sizeof(float)\* dim \* dim);

memset(pDst, 0, sizeof(float)\* dim \* dim);

for (int i = 0; i < dim; ++i)

{

// LAPACKE\_sgesv函数计算AX=B，当B为单位矩阵时，X为inv(A)

pDst[i\*(dim + 1)] = 1.0f;

}

// 调用LAPACKE\_sgesv后，会将inv(A)覆盖到X（即pDst）中

nRetVal = LAPACKE\_sgesv(LAPACK\_ROW\_MAJOR, dim, dim, pSrcBak, dim, ipiv, pDst, dim);

delete[] ipiv;

ipiv = nullptr;

delete[] pSrcBak;

pSrcBak = nullptr;

return nRetVal;

}