Intel® Math Kernel Library

Getting Started Tutorial: Using the Intel® Math Kernel Library for Matrix Multiplication

Document Number: 327356-005US

Legal Information
Legal Information

No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document.

Intel disclaims all express and implied warranties, including without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement, as well as any warranty arising from course of performance, course of dealing, or usage in trade.

This document contains information on products, services and/or processes in development. All information provided here is subject to change without notice. Contact your Intel representative to obtain the latest forecast, schedule, specifications and roadmaps.

The products and services described may contain defects or errors which may cause deviations from published specifications.

Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products.

Cilk, Intel, the Intel logo, Intel Atom, Intel Core, Intel Inside, Intel NetBurst, Intel SpeedStep, Intel vPro, Intel Xeon Phi, Intel XScale, Itanium, MMX, Pentium, Thunderbolt, Ultrabook, VTune and Xeon are trademarks of Intel Corporation in the U.S. and/or other countries.

*Other names and brands may be claimed as the property of others.

Microsoft, Windows, and the Windows logo are trademarks, or registered trademarks of Microsoft Corporation in the United States and/or other countries.

Java is a registered trademark of Oracle and/or its affiliates.

Third Party Content

Intel® Math Kernel Library (Intel® MKL) includes content from several 3rd party sources that was originally governed by the licenses referenced below:

- Portions © Copyright 2001 Hewlett-Packard Development Company, L.P.
- Sections on the Linear Algebra PACKage (LAPACK) routines include derivative work portions that have been copyrighted:
- Intel MKL supports LAPACK 3.5 set of computational, driver, auxiliary and utility routines under the following license:
  Copyright © 1992-2011 The University of Tennessee and The University of Tennessee Research Foundation. All rights reserved.
  Copyright © 2000-2011 The University of California Berkeley. All rights reserved.
  Copyright © 2006-2012 The University of Colorado Denver. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer listed in this license in the documentation and/or other materials provided with the distribution.
- Neither the name of the copyright holders nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.
The copyright holders provide no reassurances that the source code provided does not infringe any patent, copyright, or any other intellectual property rights of third parties. The copyright holders disclaim any liability to any recipient for claims brought against recipient by any third party for infringement of that parties intellectual property rights.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

The original versions of LAPACK from which that part of Intel MKL was derived can be obtained from http://www.netlib.org/lapack/index.html. The authors of LAPACK are E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen.

• The original versions of the Basic Linear Algebra Subprograms (BLAS) from which the respective part of Intel MKL was derived can be obtained from http://www.netlib.org/blas/index.html.

• XBLAS is distributed under the following copyright:

  Copyright © 2008-2009 The University of California Berkeley. All rights reserved.

  Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:
  - Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
  - Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer listed in this license in the documentation and/or other materials provided with the distribution.
  - Neither the name of the copyright holders nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

  THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

• The original versions of the Basic Linear Algebra Communication Subprograms (BLACS) from which the respective part of Intel MKL was derived can be obtained from http://www.netlib.org/blacs/index.html. The authors of BLACS are Jack Dongarra and R. Clint Whaley.

• The original versions of Scalable LAPACK (ScaLAPACK) from which the respective part of Intel MKL was derived can be obtained from http://www.netlib.org/scalapack/index.html. The authors of ScaLAPACK are L. S. Blackford, J. Choi, A. Cleary, E. D’Azevedo, J. Demmel, I. Dhillon, J. Dongarra, S. Hammarling, G. Henry, A. Petitet, K. Stanley, D. Walker, and R. C. Whaley.

• The original versions of the Parallel Basic Linear Algebra Subprograms (PBLAS) routines from which the respective part of Intel MKL was derived can be obtained from http://www.netlib.org/scalapack/html/pblas_qref.html.

• PARDISO (PARallel DIrect SOlver)* in Intel MKL was originally developed by the Department of Computer Science at the University of Basel (http://www.unibas.ch). It can be obtained at http://www.pardiso-project.org.

• The Extended Eigensolver functionality is based on the Feast solver package and is distributed under the following license:
Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. Neither the name of the University nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE AUTHOR "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

- Some Fast Fourier Transform (FFT) functions in this release of Intel® MKL have been generated by the SPIRAL software generation system (http://www.spiral.net/) under license from Carnegie Mellon University. The authors of SPIRAL are Markus Puschel, Jose Moura, Jeremy Johnson, David Padua, Manuela Veloso, Bryan Singer, Jianxin Xiong, Franz Franchetti, Aca Gacic, Yevgen Voronenko, Kang Chen, Robert W. Johnson, and Nick Rizzolo.

- Open MPI is distributed under the New BSD license, listed below.

Most files in this release are marked with the copyrights of the organizations who have edited them. The copyrights below are in no particular order and generally reflect members of the Open MPI core team who have contributed code to this release. The copyrights for code used under license from other parties are included in the corresponding files.

Copyright © 2004-2010 The Trustees of Indiana University and Indiana University Research and Technology Corporation. All rights reserved.

Copyright © 2004-2010 The University of Tennessee and The University of Tennessee Research Foundation. All rights reserved.

Copyright © 2004-2010 High Performance Computing Center Stuttgart, University of Stuttgart. All rights reserved.

Copyright © 2004-2008 The Regents of the University of California. All rights reserved.

Copyright © 2006-2010 Los Alamos National Security, LLC. All rights reserved.

Copyright © 2006-2010 Cisco Systems, Inc. All rights reserved.

Copyright © 2006-2010 Voltaire, Inc. All rights reserved.

Copyright © 2006-2011 Sandia National Laboratories. All rights reserved.

Copyright © 2006-2010 Sun Microsystems, Inc. All rights reserved. Use is subject to license terms.

Copyright © 2006-2010 The University of Houston. All rights reserved.

Copyright © 2006-2009 Myricom, Inc. All rights reserved.

Copyright © 2007-2008 UT-Battelle, LLC. All rights reserved.

Copyright © 2007-2010 IBM Corporation. All rights reserved.
Copyright © 1998-2005 Forschungszentrum Juelich, Juelich Supercomputing Centre, Federal Republic of Germany
Copyright © 2005-2008 ZIH, TU Dresden, Federal Republic of Germany
Copyright © 2007 Evergrid, Inc. All rights reserved.
Copyright © 2008 Chelsio, Inc. All rights reserved.
Copyright © 2008-2009 Institut National de Recherche en Informatique. All rights reserved.
Copyright © 2007 Lawrence Livermore National Security, LLC. All rights reserved.
Copyright © 2007-2009 Mellanox Technologies. All rights reserved.
Copyright © 2006-2010 QLogic Corporation. All rights reserved.
Copyright © 2008-2010 Oak Ridge National Labs. All rights reserved.
Copyright © 2006-2010 Oracle and/or its affiliates. All rights reserved.
Copyright © 2009 Bull SAS. All rights reserved.
Copyright © 2010 ARM ltd. All rights reserved.
Copyright © 2010-2011 Alex Brick . All rights reserved.
Copyright © 2012 The University of Wisconsin-La Crosse. All rights reserved.
Copyright © 2013-2014 Intel, Inc. All rights reserved.
Copyright © 2011-2014 NVIDIA Corporation. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer listed in this license in the documentation and/or other materials provided with the distribution.
- Neither the name of the copyright holders nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

The copyright holders provide no reassurances that the source code provided does not infringe any patent, copyright, or any other intellectual property rights of third parties. The copyright holders disclaim any liability to any recipient for claims brought against recipient by any third party for infringement of that parties intellectual property rights.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

- The Safe C Library is distributed under the following copyright:

  Copyright (c)
Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

- HPL Copyright Notice and Licensing Terms

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions, and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. All advertising materials mentioning features or use of this software must display the following acknowledgement: This product includes software developed at the University of Tennessee, Knoxville, Innovative Computing Laboratories.
4. The name of the University, the name of the Laboratory, or the names of its contributors may not be used to endorse or promote products derived from this software without specific written permission.

Copyright © 2015, Intel Corporation. All rights reserved.
Discover how to incorporate core math functions from the Intel® Math Kernel Library (Intel® MKL) to improve the performance of your application.

| About This Tutorial | This tutorial demonstrates how to use Intel MKL in your applications:  
|                     | • Multiplying matrices using Intel MKL routines  
|                     | • Measuring performance of matrix multiplication  
|                     | • Controlling threading |
| Estimated Duration   | 10-20 minutes. |
| Learning Objectives  | After you complete this tutorial, you should be able to:  
|                     | • Use Intel MKL routines for linear algebra  
|                     | • Compile and link your code  
|                     | • Measure performance using support functions  
|                     | • Understand the impact of threading on Intel MKL performance  
|                     | • Control threading for Intel MKL functions |
| More Resources       | This tutorial uses the Fortran language, but the concepts and procedures in this tutorial apply regardless of programming language. A similar tutorial using a sample application in another programming language may be available at http://software.intel.com/en-us/articles/intel-software-product-tutorials/. This site also offers a printable version (PDF) of tutorials.  
|                     | In addition, you can find more resources at http://software.intel.com/en-us/articles/intel-mkl/. |
Introduction to the Intel® Math Kernel Library

Use the Intel Math Kernel Library (Intel MKL) when you need to perform computations with high performance. Intel MKL offers highly-optimized and extensively threaded routines which implement many types of operations.

Optimization Notice

Intel's compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include SSE2, SSE3, and SSSE3 instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel. Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors. Certain optimizations not specific to Intel microarchitecture are reserved for Intel microprocessors. Please refer to the applicable product User and Reference Guides for more information regarding the specific instruction sets covered by this notice.

Notice revision #20110804

<table>
<thead>
<tr>
<th>Linear Algebra</th>
<th>Fast Fourier Transforms</th>
<th>Summary Statistics</th>
<th>Data Fitting</th>
<th>Other Components</th>
</tr>
</thead>
<tbody>
<tr>
<td>• BLAS</td>
<td>• Multi-dimensional (up to 7D) FFTs</td>
<td>• Kurtosis</td>
<td>• Splines</td>
<td>• Vector Math</td>
</tr>
<tr>
<td>• LAPACK/ScaLAPACK</td>
<td>• FFTW interfaces</td>
<td>• Variation coefficient</td>
<td>• Interpolation</td>
<td>• Trigonometric</td>
</tr>
<tr>
<td>• PARDISO*</td>
<td>• Cluster FFT</td>
<td>• Quantiles, order statistics</td>
<td>• Cell search</td>
<td>• Hyperbolic</td>
</tr>
<tr>
<td>• Iterative sparse solvers</td>
<td></td>
<td>• Min/max</td>
<td></td>
<td>• Exponential, Logarithmic</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Variance/ covariance</td>
<td></td>
<td>• Power/Root</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• ...</td>
<td></td>
<td>• Rounding</td>
</tr>
</tbody>
</table>

Exploring Basic Linear Algebra Subprograms (BLAS)

One key area is the Basic Linear Algebra Subprograms (BLAS), which perform a variety of vector and matrix operations. This tutorial uses the dgemm routine to demonstrate how to perform matrix multiplication as efficiently as possible.
Multiplying Matrices Using \texttt{dgemm}

Intel MKL provides several routines for multiplying matrices. The most widely used is the \texttt{dgemm} routine, which calculates the product of double precision matrices:

\[ C \leftarrow \alpha A \ast B + \beta C \]

The \texttt{dgemm} routine can perform several calculations. For example, you can perform this operation with the transpose or conjugate transpose of \( A \) and \( B \). The complete details of capabilities of the \texttt{dgemm} routine and all of its arguments can be found in the \texttt{?gemm} topic in the Intel Math Kernel Library Reference Manual.

\textbf{Use \texttt{dgemm} to Multiply Matrices}

This exercise demonstrates declaring variables, storing matrix values in the arrays, and calling \texttt{dgemm} to compute the product of the matrices. The arrays are used to store these matrices:

\[
A = \begin{bmatrix}
1.0 & 1001.0 & 2001.0 & \cdots & 999001.0 \\
2.0 & 1002.0 & 2002.0 & \cdots & 999002.0 \\
3.0 & 1003.0 & 2003.0 & \cdots & 999003.0 \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
1000.0 & 2000.0 & 3000.0 & \cdots & 1000000.0 \\
\end{bmatrix}
\]

\[
B = \begin{bmatrix}
-1.0 & -1001.0 & -2001.0 & \cdots & -999001.0 \\
-2.0 & -1002.0 & -2002.0 & \cdots & -999002.0 \\
-3.0 & -1003.0 & -2003.0 & \cdots & -999003.0 \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
-1000.0 & -2000.0 & -3000.0 & \cdots & -1000000.0 \\
\end{bmatrix}
\]

The one-dimensional arrays in the exercises store the matrices by placing the elements of each column in successive cells of the arrays.

\textbf{NOTE}

The Fortran source code for the exercises in this tutorial is found in `<install-dir>\Samples\en-US\mkl\tutorials.zip` (Windows* OS), or `<install-dir>/Samples/en-US/mkl/tutorials.zip` (Linux* OS/OS X*).

Although Intel MKL supports Fortran 90 and later, the exercises in this tutorial use FORTRAN 77 for compatibility with as many versions of Fortran as possible.

After you unzip the tutorials.zip file, the Fortran source code can be found in the mkl_mmx_f directory, and the C source code can be found in the mkl_mmx_c directory.

* Fortran source code is found in dgemm_example.f

\[
\text{PROGRAM MAIN}
\]
\[
\text{IMPLICIT NONE}
\]
\[
\text{DOUBLE PRECISION ALPHA, BETA}
\]
\[
\text{INTEGER M, K, N, I, J}
\]
\[
\text{PARAMETER (M=2000, K=200, N=1000)}
\]
\[
\text{DOUBLE PRECISION A(M,K), B(K,N), C(M,N)}
\]
\[
\text{PRINT *, "This example computes real matrix C=alpha*A*B+beta*C"}
\]
\[
\text{PRINT *, "using Intel(R) MKL function dgemm, where A, B, and C"}
\]
\[
\text{PRINT *, "are matrices and alpha and beta are double precision "}
\]
\[
\text{PRINT *, "scalars"}
\]
\[
\text{PRINT *, ""}
\]
PRINT *, "Initializing data for matrix multiplication C=A*B for ", " matrix A(M, K) and matrix B(K, N)"
10 FORMAT(a,i5,a,i5,a,i5,a,i5,a)
PRINT *, ""
ALPHA = 1.0
BETA = 0.0
PRINT *, "Initializing matrix data"
PRINT *, ""
DO I = 1, M
  DO J = 1, K
    A(I,J) = (I-1) * K + J
  END DO
END DO
DO I = 1, K
  DO J = 1, N
    B(I,J) = -((I-1) * N + J)
  END DO
END DO
DO I = 1, M
  DO J = 1, N
    C(I,J) = 0.0
  END DO
END DO
PRINT *, "Computing matrix product using Intel(R) MKL DGEMM"
PRINT *, "subroutine"
PRINT *, "Computations completed."
PRINT *, ""
PRINT *, "Top left corner of matrix A:"
PRINT 20, ((A(I,J), J = 1,MIN(K,6)), I = 1,MIN(M,6))
PRINT *, ""
PRINT *, "Top left corner of matrix B:"
PRINT 20, ((B(I,J), J = 1,MIN(N,6)), I = 1,MIN(K,6))
PRINT *, ""
20 FORMAT(6(F12.0,1x))
PRINT *, "Top left corner of matrix C:"
PRINT 30, ((C(I,J), J = 1,MIN(N,6)), I = 1,MIN(M,6))
PRINT *, ""
30 FORMAT(6(ES12.4,1x))
PRINT *, "Example completed."
STOP
END

NOTE
This exercise illustrates how to call the dgemm routine. An actual application would make use of the result of the matrix multiplication.
This call to the `dgemm` routine multiplies the matrices:

```fortran
```

The arguments provide options for how Intel MKL performs the operation. In this case:

- `'N'` Character indicating that the matrices $A$ and $B$ should not be transposed or conjugate transposed before multiplication.
- $M$, $N$, $K$ Integers indicating the size of the matrices:
  - $A$: $M$ rows by $K$ columns
  - $B$: $K$ rows by $N$ columns
  - $C$: $M$ rows by $N$ columns
- $ALPHA$ Real value used to scale the product of matrices $A$ and $B$.
- $A$ Array used to store matrix $A$.
- $M$ Leading dimension of array $A$, or the number of elements between successive columns (for column major storage) in memory. In the case of this exercise the leading dimension is the same as the number of rows.
- $B$ Array used to store matrix $B$.
- $K$ Leading dimension of array $B$, or the number of elements between successive columns (for column major storage) in memory. In the case of this exercise the leading dimension is the same as the number of rows.
- $BETA$ Real value used to scale matrix $C$.
- $C$ Array used to store matrix $C$.
- $M$ Leading dimension of array $C$, or the number of elements between successive columns (for column major storage) in memory. In the case of this exercise the leading dimension is the same as the number of rows.

**Compile and Link Your Code**

Intel MKL provides many options for creating code for multiple processors and operating systems, compatible with different compilers and third-party libraries, and with different interfaces. To compile and link the exercises in this tutorial with Intel® Parallel Studio XE Composer Edition, type

- **Windows* OS:** `ifort /Qmkl dgemm_example.f`
- **Linux* OS, OS X*: `ifort -mkl dgemm_example.f`

**NOTE**

This assumes that you have installed Intel MKL and set environment variables as described in .


After compiling and linking, execute the resulting executable file, named `dgemm_example.exe` on Windows* OS or `a.out` on Linux* OS and OS X*.
Optimization Notice

Intel's compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include SSE2, SSE3, and SSSE3 instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel. Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors. Certain optimizations not specific to Intel microarchitecture are reserved for Intel microprocessors. Please refer to the applicable product User and Reference Guides for more information regarding the specific instruction sets covered by this notice.

Notice revision #20110804

See Also


Intel Math Kernel Library Knowledge Base for articles describing usage of Intel MKL functionality.
Measuring Performance with Intel® MKL Support Functions

Intel MKL provides functions to measure performance. This provides a way of quantifying the performance improvement resulting from using Intel MKL routines in this tutorial.

Measure Performance of dgemm

Use the dsecnd routine to return the elapsed CPU time in seconds.

NOTE
The quick execution of the dgemm routine makes it difficult to measure its speed, even for an operation on a large matrix. For this reason, the exercises perform the multiplication multiple times. You should set the value of the LOOP_COUNT constant so that the total execution time is about one second.

Measure Performance Without Using dgemm

In order to show the improvement resulting from using dgemm, perform the same measurement, but use a triply-nested loop to multiply the matrices.
END DO

PRINT *, "Measuring performance of matrix product using ", "triple nested loop"
PRINT *, ""
S_INITIAL = DSECND()
DO R = 1, LOOP_COUNT
   DO I = 1, M
      DO J = 1, N
         TEMP = 0.0
         DO L = 1, K
            TEMP = TEMP + A(I,L) * B(L,J)
         END DO
         C(I,J) = TEMP
      END DO
   END DO
END DO
S_ELAPSED = (DSECND() - S_INITIAL) / LOOP_COUNT
PRINT *, "== Matrix multiplication using triple nested loop ==";
PRINT 50, " == completed at ", S_ELAPSED*1000, " milliseconds =="
50 FORMAT(A,F12.5,A)
PRINT *, ""

Compare the results in the first exercise using \texttt{dgemm} to the results of the second exercise without using \texttt{dgemm}.

You can find more information about measuring Intel MKL performance from the article "A simple example to measure the performance of an Intel MKL function" in the Intel Math Kernel Library Knowledge Base.

\textbf{See Also}

\texttt{Intel MKL Documentation} for additional Intel MKL documentation, including the \texttt{Intel MKL Reference Manual} and the \texttt{Intel MKL User's Guide}.

\texttt{Intel Math Kernel Library Knowledge Base} for articles describing usage of Intel MKL functionality.
Measuring Effect of Threading on dgemm

By default, Intel MKL uses \( n \) threads, where \( n \) is the number of physical cores on the system. By restricting the number of threads and measuring the change in performance of dgemm, this exercise shows how threading impacts performance.

**Limit the Number of Cores Used for dgemm**

This exercise uses the mkl_set_num_threads routine to override the default number of threads, and mkl_get_max_threads to determine the maximum number of threads.

* Fortran source code is found in dgemm_threading_effect_example.f

```fortran
PRINT *, "Finding max number of threads Intel(R) MKL can use for"  
PRINT *, "parallel runs"
PRINT *, ""
MAX_THREADS = MKL_GET_MAX_THREADS()

PRINT 20," Running Intel(R) MKL from 1 to ",MAX_THREADS," threads"
20   FORMAT(A,I2,A)
PRINT *, ""
DO L = 1, MAX_THREADS
   DO I = 1, M
      DO J = 1, N
         C(I,J) = 0.0
      ENDDO
   ENDDO
PRINT *, " Requesting Intel(R) MKL to use ",L," thread(s)"
30   FORMAT(A,I2,A)
   CALL MKL_SET_NUM_THREADS(L)
   PRINT *, "Making the first run of matrix product using ",L," thread(s)"
   PRINT *, "Measuring performance of matrix product using ",L," thread(s)"
40   FORMAT(A,I2,A)
   S_INITIAL = DSECND()
   DO R = 1, LOOP_COUNT
   END DO
   S_ELAPSED = (DSECND() - S_INITIAL) / LOOP_COUNT
   PRINT *, " == Matrix multiplication using Intel(R) MKL DGEMM =="
50   FORMAT(A,F12.5,A)
   PRINT *, " == completed at ",S_ELAPSED*1000," milliseconds =="
50   PRINT *, " == using ",L," thread(s) =="
   FORMAT(A,I2,A)
   PRINT *, ""
END DO
```
Examine the results shown and notice that time to multiply the matrices decreases as the number of threads increases. If you try to run this exercise with more than the number of threads returned by `mkl_get_max_threads`, you might see performance degrade when you use more threads than physical cores.

**NOTE**


**See Also**


Intel Math Kernel Library Knowledge Base for articles describing usage of Intel MKL functionality.
Other Areas to Explore

The exercises so far have given the basic ideas needed to get started with Intel MKL, but there are plenty of other areas to explore. The following are some controls, interfaces, and topics which you might find worth investigating further.

Support functions

The second exercise shows how to use the timing functions and the third exercise shows the use of threading control functions. Acquaint yourself with other support functions by referring to the "Support functions" chapter of the Intel MKL Reference Manual:

- Support functions for Conditional Numerical Reproducibility (CNR)
  These functions provide the means to balance reproducibility with performance in certain conditions.
- Memory functions
  These functions provide support for allocating and freeing memory. The allocation functions allow proper alignment of memory to ensure reproducibility when used together with CBWR functions.
- Error handling functions
  The xerbla function is used by BLAS, LAPACK, VML, and VSL to report errors.

Linking and interfaces

- The ILP64 interface
  Most users call the interface of Intel MKL that takes 32-bit integers for size parameters, but increased memory and also some legacy code requires 64-bit integers. Read more about the ILP64 interface and the libraries and functions supporting it in the Intel MKL User's Guide.
- Single Dynamic Library (SDL) linking model
  Intel MKL has two ways to link to dynamic libraries. The newest of these models is the best option for those calling Intel MKL from managed runtime libraries and is easy to link, but requires some functions calls to use non-default interfaces (for example, ILP64). See the Intel MKL User's Guide for more information on Intel MKL linking models.

Miscellaneous

- Environment variables
  Many controls in Intel MKL have both environment variables and functional versions. In all cases the function overrides the behavior of the environment variable. If you do not want the behavior to change based on an environment variable in a particular case, use the function call to ensure the desired setting. See the Intel MKL User’s Guide for descriptions of the environment variables used by Intel MKL.

See Also


Intel Math Kernel Library Knowledge Base for articles describing usage of Intel MKL functionality.