Intel MKL Poisson Library for scalable and efficient solution of elliptic problems with separable variables

A. Kalinkin, A. Kuzmin

Intel Corporation

Intel®MKL Poisson Library is a collection of routines that combine discrete Fourier transforms and LU decomposition to solve Laplace, Poisson, and Helmholtz mesh problems with separable variables. The routines provide an approximate solution of some two- and three-dimensional problems with an arbitrary combination of boundary conditions, Dirichlet, Neumann, or periodic, in Cartesian or spherical coordinate systems. Intel MKL®Poisson Library is optimized for modern Intel multi-core processors and demonstrates excellent performance and scalability compared to similar solvers.

1. Introduction

Among elliptic boundary value problems, the class of problems with separable variables can be solved fast and directly. Elliptic problems with separable variables usually assume that the computational domains are simple e.g., parallelepiped or sphere, and constant coefficients [1]. This kind of problems can serve to generate preconditioners in iterative procedures for more complex methods. They can also be used in low-accuracy models similar to the ones used in Numerical Weather Simulations.

Intel® MKL Poisson Library enables solving Laplace, Poisson, and Helmholtz mesh problems with separable variables. The routines provide an approximate solution of some two- and three-dimensional problems with an arbitrary combination of boundary conditions, Dirichlet, Neumann, or periodic, in Cartesian or spherical coordinate systems.

The problems with the separable variables can be suboptimal in the sense that they require slightly more arithmetic operations (up to logarithmic factor) than the number of unknowns to compute the solution to the problem. This statement is true if, for example, the sizes of the discretized problems are powers of 2.

Computational Mathematics suggests that we take into consideration not only arithmetic operations, but also the cost of memory operations as well. Modern computers can perform computations at a very high speed, while lacking the ability to deliver data to the computational units. Keeping in mind that a memory operation can easily be dozen to hundred times slower than an arithmetic one, a computationally optimal algorithm could compute the solution slower than memory optimal algorithm.

The recent developments in processor industry resulted in multicore processors become standard processors not only for powerful clusters, but also for home computers and laptops. Therefore, the algorithm can also be non-optimal from the parallelization point of view. Optimality here can be understood in terms of the number of synchronization points and/or in terms of the amount of data that needs to be transferred between different cores.

In summary, the modern computational algorithm should focus on 3 key aspects, namely, parallelization, memory operations, and arithmetic costs.

This work extends previously published paper [2] with new routines and performance comparison.

The purpose of this article is to demonstrate on a simple 3D problem with separable variables that taking into account modern model the solution can be computed efficiently and fast. This would also help developers to compute solution to the problem with separable variables really negligible with respect to other computations. To complete our goal, we will evaluate software provided by Intel Corporation. In particular, we focus on the comparison (where possible) of NAG* SMP Library provided at [3] and Intel® MKL provided at [4].

*Other brands and names are the property of their respective owners.

2. Problem Statement

We are going to use the following notation for boundaries of a parallelepipedal domain $a_x < x < b_x$, $a_y < y < b_y$, $a_z < z < b_z$ in Cartesian space:

$bd\_a_x = \{x = a_x, a_y \le y \le b_y, a_z \le z \le b_z\}$, $bd\_b_x = \{x = b_x, a_y \le y \le b_y, a_z \le z \le b_z\}$

$bd\_a_y = \{a_x \le x \le b_x, y = a_y, a_z \le z \le b_z\}$, $bd\_b_y = \{a_x \le x \le b_x, y = b_y, a_z \le z \le b_z\}$

$bd\_a_z = \{a_x \le x \le b_x, a_y \le y \le b_y, z = a_z\}$, $bd\_b_z = \{a_x \le x \le b_x, a_y \le y \le b_{y,}, z = b_{z,}\}$.

The wildcard "*" may stand for any of the symbols a_x, b_x, a_y, b_y, a_z, b_z so that *bd_** denotes any of the above boundaries.

The 2D Helmholtz problem is to find an approximate solution of the Helmholtz equation $-\frac{\partial^2 u}{\partial x^2} - \frac{\partial^2 u}{\partial y^2} - \frac{\partial^2 u}{\partial z^2} + qu = f(x, y, z),\ \ q = const \ge 0$, in a parallelepiped, that is, a domain $a_x < x < b_x$, $a_y < y < b_y$, $a_z < z < b_z$ with one of the following boundary conditions on each boundary *bd_**:

- The Dirichlet boundary condition: $u(x, y, z) = G(x, y, z)$
- The Neumann boundary condition: $\frac{\partial u}{\partial n}(x, y, z) = g(x, y, z)$, where

n= *-x* on bd_a_x, *n*= *x* on bd_b_x, *n*= *-y* on bd_a_y, *n*= *y* on bd_b_y, and *n*= *-z* on bd_a_z, *n*= *z* on bd_b_z.

- The periodic boundary condition: $u(a_x, y, z) = u(b_x, y, z)$, $u(x, a_y, z) = u(x, b_y, z)$ or $u(x, y, a_z) = u(x, y, b_z)$

We can see that the Poisson problem can be obtained from the Helmholtz problem by setting the Helmholtz coefficient q to zero. The Laplace problem can be obtained by setting the right-hand side f to zero in addition to Helmholtz coefficient.

To find an approximate solution for 3D problems, a uniform mesh is built in the parallelepipedal domain:

$$x_i = a_x + ih_x, x_j = x(j)$$
$$i = 0, .., n_x,\ \ h_x = \frac{b_x - a_x}{n_x},$$
$$a_x = x(0) < .. < x(j) < .. < x(n_j) = b_x$$

It is possible to use the standard five-point finite difference approximation on this mesh to compute the approximation to the solution:

We assume that the values of the approximate solution are computed in the mesh points (x_i, y_i, z_i), provided the values of the right-hand side *f(x, y, z)* at these points are given and the values of the appropriate boundary functions *G(x, y, z)* and/or *g(x, y, z)* in the mesh points laying on the boundary of the rectangular domain are known.

Discrete Fourier Transform (DFT) computations are highly dependent on the dimension. For powers of 2, the DFT requires the least possible number of operations, while for the primes the number of operations reaches its maximum value. We consider only dimensions that are powers of 2 as the difficult test case with a high data movement to operations ratio.

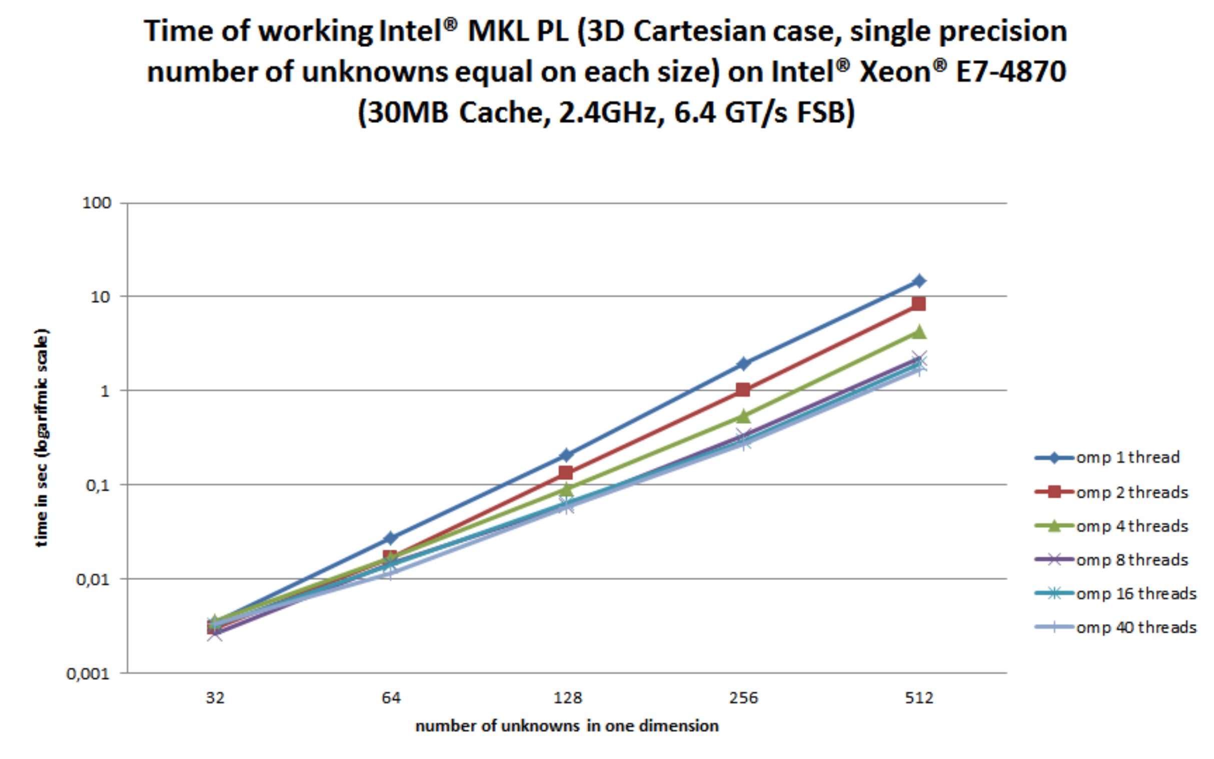
3. Single precision performance

We first look at single precision computations that are of value for Numerical Weather Simulation problems and consider the Poisson Library (PL) from Intel® MKL. PL does computations in four steps by consecutive calls to the s_init_helmholtz_2d, s_commit_helmholtz_2d, s_helmholtz_2d, and free_helmholtz_2d routines. We measure the total time spent in computations.

We consider the homogeneous Dirichlet problem with an exact solution $u(x,y,z) = \sin \pi x \sin \pi y \sin \pi z$ on the rectangular domain 0<x<1, 0<y<1, 0<z<1 as our test case.

All test cases were compiled with Intel® Fortran compilers (version 11.1). We ran each piece of code 10 times in a loop and then selected the best time out of the ten collected. Time measurements were completed using the dsecnd routine from Intel® MKL. We also used the Poisson Library from Intel® MKL 10.3 Update 7.

Figure 1 shows scalability of Intel® MKL PL routines in 3D Cartesian case. We measure computational time spent in four PL routines for different regular mesh sizes starting from 32x32x32 and ending with 512x512x512 and with different numbers of OMP threads.



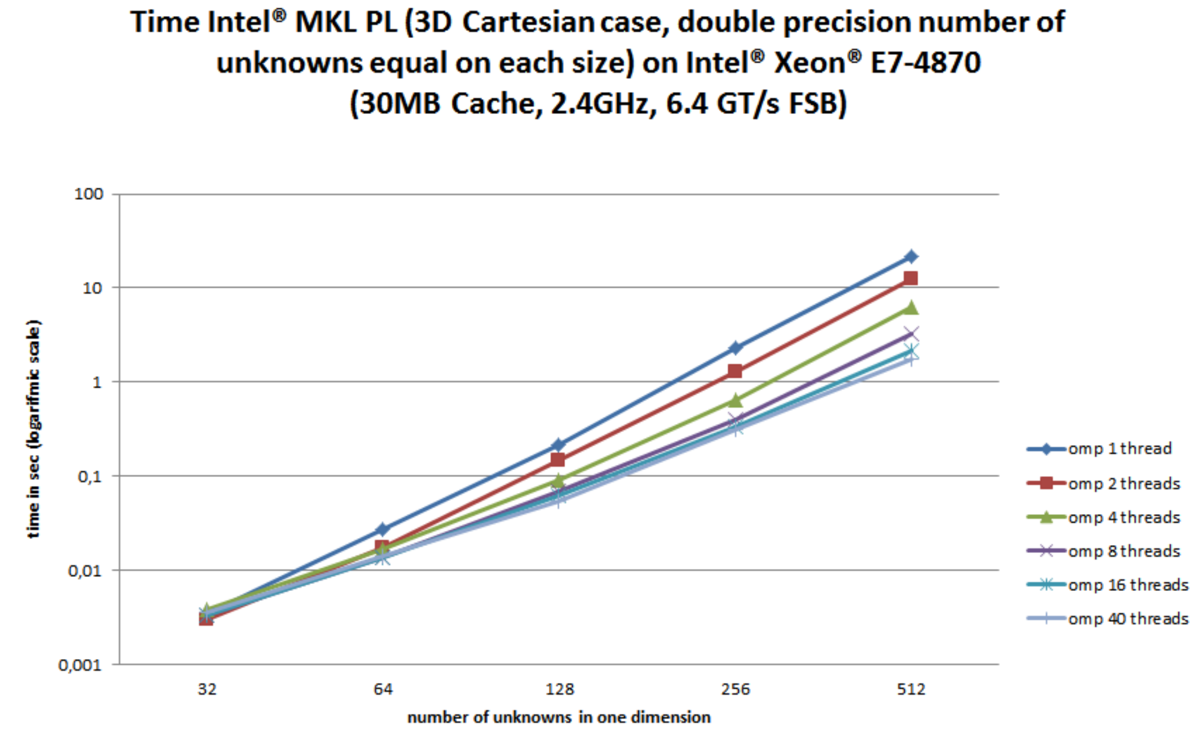
Performance tests and ratings are measured using specific computer systems and/or components and reflect the approximate performance of Intel products as measured by those tests. Any difference in system hardware or software design or configuration may affect actual performance. Buyers should consult other sources of information to evaluate the performance of systems or components they are considering purchasing. For more information on performance tests and on the performance of Intel products, refer to www.intel.com/performance/resources/benchmark_limitations.htm
Refer to our Optimization Notice for more information regarding performance and optimization choices in Intel software products at:
http://software.intel.com/en-us/articles/optimization-notice/

Figure 1. Scalability of Intel® MKL PL (3D Cartesian case) (single precision)

4. Double precision performance

We next look at double precision computations that are of value for preconditioning of elliptic problems with slightly varying coefficients and we consider the D03FAF routine from NAG* SMP Library and the Poisson Library from Intel® MKL. The D03FAF routine is able to compute the solution to the Helmholtz problem in a Cartesian coordinate system in a single step. PL does computations in four steps by consecutive calls to the double precision routines d_init_helmholtz_2d, d_commit_helmholtz_2d, d_helmholtz_2d, and free_helmholtz_2d. For fairness, we measured the total time spent in computations for both software libraries.

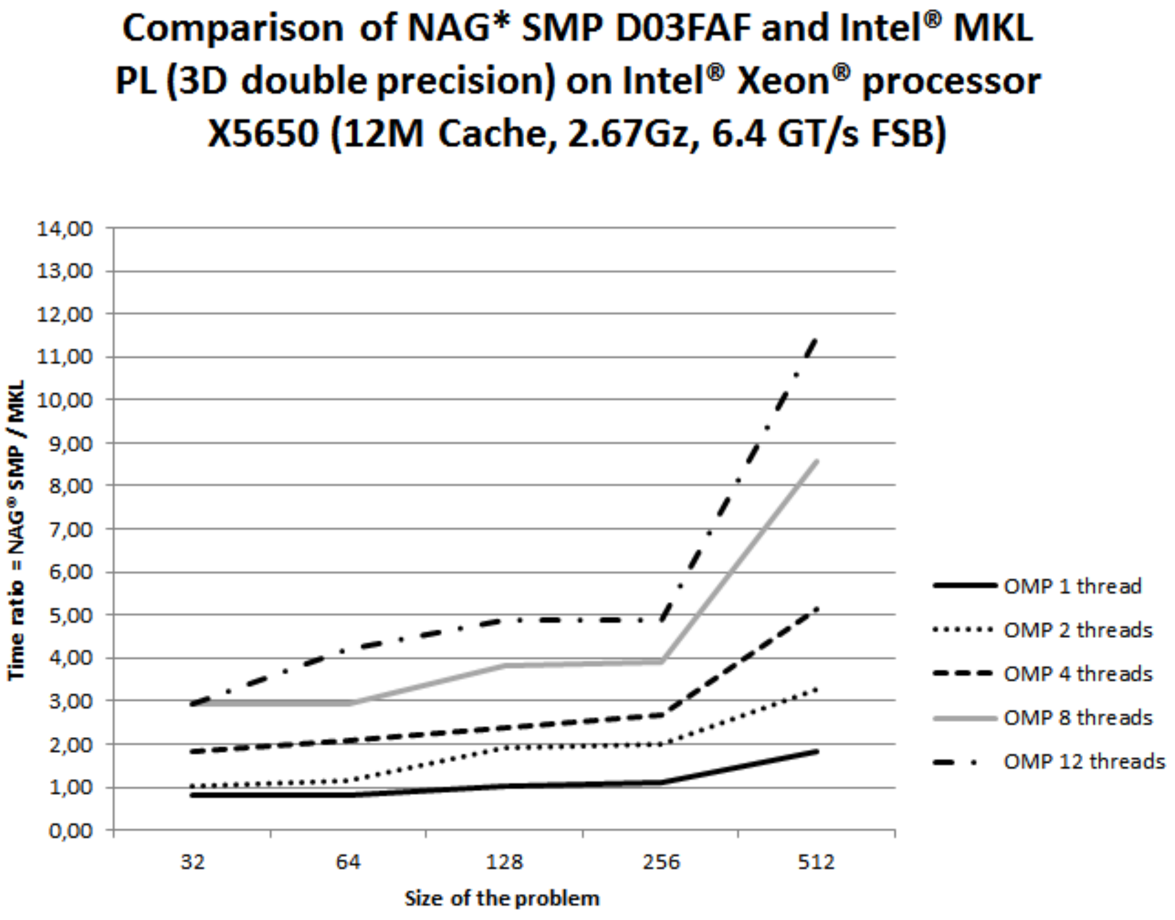
We consider the same homogeneous Dirichlet problem with the same exact solution $u(x, y, z) = \sin \pi x \sin \pi y \sin \pi z$ on the same rectangular domain 0<x<1, 0<y<1, 0<z<1 as our test case.



Performance tests and ratings are measured using specific computer systems and/or components and reflect the approximate performance of Intel products as measured by those tests. Any difference in system hardware or software design or configuration may affect actual performance. Buyers should consult other sources of information to evaluate the performance of systems or components they are considering purchasing. For more information on performance tests and on the performance of Intel products, refer to www.intel.com/performance/resources/benchmark_limitations.htm
Refer to our Optimization Notice for more information regarding performance and optimization choices in Intel software products at: http://software.intel.com/en-us/articles/optimization-notice/

Figure 2. Scalability of Intel® MKL PL (3D Cartesian case) (double precision)

Figure 2 shows scalability of Intel® MKL PL routines in 3D Cartesian case in double precision arithmetic. Figure 3 shows the ratio of computational time spent in the D03FAF routine and computational time spent in four PL routines for different regular mesh sizes starting from 32 and ending with 512 mesh points in each dimension. When the ratio curve is above 1, PL Helmholtz 2D solver is faster than D03FAF.



Performance tests and ratings are measured using specific computer systems and/or components and reflect the approximate performance of Intel products as measured by those tests. Any difference in system hardware or software design or configuration may affect actual performance. Buyers should consult other sources of information to evaluate the performance of systems or components they are considering purchasing. For more information on performance tests and on the performance of Intel products, refer to www.intel.com/performance/resources/benchmark_limitations.htm

Refer to our Optimization Notice for more information regarding performance and optimization choices in Intel software products at: http://software.intel.com/en-us/articles/optimization-notice/

*Other brands and names are the property of their respective owners.

Figure 3. Comparison of NAG* **SMP** D03FAF and Intel® MKL PL (3D double precision)

From Figure 3, it can be seen that even on a single thread, the PL routines can be roughly two times faster than the D03FAF routine. When threaded, the advantage grows up to 11 times on 12 threads. It is also clear that PL has heavier interface that results in essentially slower performance for small problems (sizes up to 100). We think that small size problems are not of great interest for HPC area, so the lower performance of PL in this range should not be considered as a problem. However, PL can regain some performance in the case when the solution of problems different in right-hand side only as it can do pre- (init and commit) and post-computational (free) step only once unlike the D03FAF routine.

5. Conclusions

We have investigated the performance of software available for solving 3D Helmholtz problems with separable variables in parallelepipedal domains. We found that the implementation of the solver from

NAG SMP Library shows good serial performance for small size problems (up to 64 mesh points in one dimension). The optimized implementation of Intel® MKL PL shows better performance and scalability on larger problem sizes. Performance gains are up to 10 times for some dimensions. Both libraries produce solutions with the same level of accuracy.

References

1. A. A. Samarskii and E. S. Nikolaev, Methods of Solution of Grid Problems, Nauka, Moscow (1978) (in Russian)

2. A. A. Kalinkin, Y.M. Laevsky, S.V. Gololobov, 2D Fast Poisson Solver for High-Performance Computing, Parallel Computing Technologies, Lecture Notes in Computer Science 2009, Vol. 5698/2009

3. NAG® SMP Library, http://www.nag.co.uk/numeric/fl/fsdescription.asp

4. Intel® Math Kernel Library, http://www.intel.com/software/products/mkl