A Study of Numerical Linear Algebra Library in the Context of Adaptive Array Beamforming

Sreekanth V. P. , Sooraj K. Ambat and V. P. Felix

Naval Physical & Oceanographic Laboratory, Kochi-682 021, India
sreekanthvp@gmail.com, sooraj.k.ambat@gmail.com, v_paulfelix@yahoo.co.in

Abstract: Sonar system is an embodiment of sensor signal processing, designed to extract information from the real world, which shares much embedded computing application. The complexity of the most modern computing platforms has made it extremely difficult to write numerical code that achieves the best possible performance. In this paper, we discuss Intel Math Kernel Library (MKL) as a mathematical software package for scientific and technical computation for ease of use in Intel-architecture based COTS (commercial off- the-shelf) systems that can vary greatly. Here we discuss how the advances in computing platform can advantage the sonar signal processing system. We have taken adaptive beamforming as the case study for the comparison of improvements from direct implementation to implementation using linear algebra library for COTS systems.

Keywords: scientific computing, beam-forming, Intel MKL, MVDR

1. Introduction

Development of sonar systems requires a variety of signal processing subsystems, such as beamformer, target detector, target localization, tracking and classification [1,2,3]. The traditional way of building each of the above subsystems has been with programmable DSP hardware. The key technology hurdle the system developer faces in this approach is the development of the software that codes the specific signal processing algorithm that accomplishes each subsystems function. Modern signal processing algorithms requires use of computational intensive linear algebra operations such as Matrix inversion, eigenvalue decomposition etc.

Use of DSP hardware for embedded system development requires linear algebraic routine coding and its software validation, which is time consuming. Two technology developments help the system developer to significantly shorten the system development time. These are the advent of multi-core general purpose CPU and ready availability of linear algebraic library by manufacturer such as Intel. Leveraging these technological advances in computing platform for sonar system development has important advantages. Utilizing the tested linear algebraic library, rapid advances in general purpose CPU development allow new hardware to be inserted while preserving the software investment. This paper deals with exploratory work done in this direction utilizing Intel library and Intel multi-core architecture, Adaptive beamformer for

978-1-4244-9118-6/09/$26.00 ©2009 IEEE

broadband passive sonar is taken as a case study.

2. Beamformer

Beamformer is a method of coherently summing the outputs of spatially distributed sensors to improve the signal reception in the presence of noise. Adaptive beamforming is used to improve signal reception by maintaining a narrow beamwidth directed towards the desired signal while adaptively steering the side lobes away from the directional interference. Minimum Variance Distortionless Response (MVDR) algorithm is an optimal beamformer that chooses weights to minimize the output power subject to a unity gain constraint in the steering direction. MVDR requires the computation of the cross spectral density matrix and, inverse of cross spectral density matrix for the computation of adaptive weights, which is the computational bottleneck in the implementation. A broadband Beamformer implementation requires above computations for all frequency bins.

In this paper, we evaluate the performance of an adaptive broadband Beamformer using the Intel MKL library and a naive implementation (code without any optimized signal processing libraries) of the same in C++.

3. Intel Math Kernel Library

The Intel Math Kernel Library (MKL) is a math Library for use in scientific, engineering applications supporting a number of different mathematical areas[4, 5]. This includes optimized implementation of Basic Linear algebra Subroutines (BLAS) [6, 7], LAPACK [8], ScaLAPACK, Sparse BLAS, iterative sparse solvers, FFTs, cluster FFTs, and many others.

Intel MKL library is compiler independent, which eliminates the need for compiler specific versions and allows C language programs to link to the Fortran portions of the library without the usual Fortran runtime libraries. That is all compiler dependencies have been isolated. It also provides competitive performance on non-Intel (AMD) processors. Intel MKL library parallelizes those parts of the library where parallelization makes sense [9].

We are studying the performance of Minimum Variance Distortionless Response (MVDR) adaptive beamforming algorithm, implemented in Intel MKL and naïve code implementation, for the Direction of arrival estimation in sonar application. Adaptive beamforming provides an attractive solution for the direction of arrival estimation problem encountered in these applications. Minimum Variance Distortionless Response (MVDR) algorithm is an optimum beamformer that chooses adaptive weights to minimize the output power subject to a unity gain constraint in the steering direction

4. Adaptive Beamformer Algorithm

The aim of the adaptive beamforming is to optimize a collection of weight vectors to locate a directional source. The output of the sensor arrays would be correlated, since the directional signal would have spatial correlation across the array. A typical adaptive beamforming scheme is depicted in Figure 1.

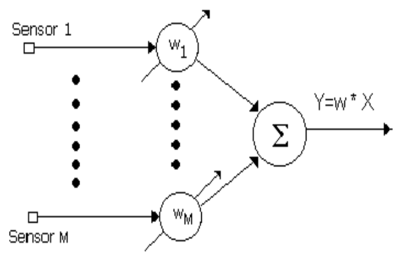

Fig. 1 A typical beamforming scheme

Complex weights for each element of the array are adaptively optimized to the received signal. There are different methods in arriving at this optimization problem.

Traditional Minimum Variance Distortionless Response (MVDR) method provides a straight forward and effective approach in solving this problem [3]. MVDR falls under the category of linearly constrained algorithm. By linearly constraining the weight we are minimizing the contribution to the output from the interference signal.

4.1. MVDR Overview

The approach of MVDR is based on optimum filtering and spectrum analysis. In this, a cost function is first defined and minimized in order to find the optimal connection weights between the input signal from the array and beam former output. Here the computation of the inverse correlation matrix and its multiplication with steering vector are the most important parts in the process of optimal weight computation. The array correlation matrix (R) is a measure of the spatial correlation of the signal and noise arriving at the array. Adaptive beam forming techniques measures the array correlation matrix instead of assuming that the noise is spatially white and Gaussian. This array correlation matrix measurement is then used to determine the spatial filter coefficients (weights).

The objective of beamforming is to resolve the direction of arrival of spatially separated signals within the same frequency band. The source of the signals is located far from sensor array. This assumption leads to plane wave propagation model. The incoming data is spatially filtered by an array of sensors. The weight by which subsequent sensor outputs are multiplied prior to summation determines the beam pattern response. The output of the spatial filter has an improved SNR than that obtained from a single sensor.

In our experiment, the frequency domain beamforming approach is chosen. Consider a linear array with *'N'* sensors uniformly spaced. Each sensor time series data is converted into frequency domain. The frequency domain data corresponding to the frequency bin of interest, k is

$$\boldsymbol{X} = \left[\boldsymbol{X}_{1k} \ \boldsymbol{X}_{2k} \ \ldots\ldots\ldots \boldsymbol{X}_{Nk} \right] \qquad (1)$$

The output $\boldsymbol{Y}$ for particular look direction can be expressed as

$$Y = w^* X \qquad (2)$$

where the * denotes complex-conjugate transposition, $\boldsymbol{X}$ and $\boldsymbol{w}$ represent the column vector of frequency-domain sensor data and the column vector of adaptive weights respectively.

The steering vector is independent of the incoming data. The equation of the steering vector $\boldsymbol{s}$ for uniform linear array is given by,

$$\boldsymbol{s}(\boldsymbol{\theta}) = \left[1, \boldsymbol{e}^{-\boldsymbol{jkd}\sin(\boldsymbol{\theta})}, \ldots, \boldsymbol{e}^{-\boldsymbol{j}(\boldsymbol{N}-1)\boldsymbol{kd}\sin(\boldsymbol{\theta})}\right] \qquad (3)$$

where $\boldsymbol{k}$ is the wave number, $\boldsymbol{N}$ is the number of sensor nodes, and $\boldsymbol{d}$ is the distance between the sensors in the array. The CSM (Cross Spectral density Matrix) $\boldsymbol{R}$ is the autocorrelation of the frequency domain sensor data vector

$$R = E\left\{X \cdot X^*\right\} \qquad (4)$$

where E is the expectation operator.

The output power for each steering direction is defined as the expected value of the squared magnitude of the beam former output, as given by

$$P = E\left\{ |Y|^2 \right\} = w^* E\left\{X \cdot X^T\right\} w = w^* R w \qquad (5)$$

MVDR is an adaptive beam former with a linear constraint, the constraint being

$$w * s = g \qquad (6)$$

Because of this constraint, signal from the steering direction is passed with gain g, which is equal to ***unity*** in the case of MVDR. Output power contributed by the interfering signal coming from other directions are minimized using a minimization criteria

$$\underset{w}{Min} \left(P = w * Rw \right)$$

Constrained to

$$w * s = 1 \tag{7}$$

Using Lagrange multipliers method, solution obtained as

$$w = \frac{R^{-1} s}{s^* R^{-1} s} \tag{8}$$

By substituting Eq (7) into the expression for power ***P*** in Eq (6), the output power for a single steering direction is found out to be

$$P(\theta) = \frac{1}{s^* R^{-1} s} \tag{9}$$

MVDR algorithm calculates the optimal set of weights based on the sampled data, resulting in a beam pattern that suppresses the response to directions in which strong source of interference are present.

MVDR requires the computation of the cross spectral density matrix and, inverse of cross spectral density matrix for the computation of adaptive weights.

The computational bottleneck is in finding the inverse of the cross spectral density matrix. Computation of inverse of a matrix is of degree *$O(N^3)$*.

The order N of the cross spectral density matrix vary with the specification and method of beamforming is done. We implemented the computation of R^{-1} in naive code and using Intel MKL library, timings are computed for different orders of R. Timings for computing the inverse are given in Table1 for naive code implementation and Intel MKL implementation.

Table1: Timings in finding the inverse of R for different orders in naive code and using Intel-MKL in seconds

Order N	Naive code	MKL
2	$5.01x10^{-5}$	$1.50x10^{-5}$
4	$3.42x10^{-4}$	$1.62x10^{-5}$
8	$3.52x10^{-3}$	$2.00x10^{-5}$
16	$4.58x10^{-2}$	$3.41x10^{-5}$
32	$6.64x10^{-1}$	$9.92x10^{-5}$
64	1.01x10	$5.40x10^{-4}$
128	$1.59x10^{2}$	$2.98x10^{-3}$
256	$2.52x10^{3}$	$1.47x10^{-2}$

5. Simulation and Results

The simulations were carried out using the following software in the hardware platform with specifications given below.

Software:

Operating System: Linux (Fedora 8)

Compiler: gcc version 4.2.4

Hardware:

Processor: Intel Xeon Core2Duo

CPU MHz: 2000

L1 I cache: 32K, L1 D cache: 32K

L2 cache: 4096 K

Measuring the execution time for code can be done in multiple ways in C++. Except for time resolution issue, different timing methods worked relatively the same in single processor environment. As multi-core processors become more prevalent however we need to be careful at choosing the correct timing mechanism as not all such routines measure the wall time elapsed. The method we used is the *rdstc (read time stamp counter)* instruction available on most Intel processors since Pentium. In Intel processors, the *rdtsc* instruction reads the the time stamp counter which allows near-cycle accurate timing. Cross spectral density matrix R is a positive definite hermitian and in MKL we use ~potrf function and ~potri function of LAPACK routines[4,7,8], to factorize and to find inverse respectively. In naïve code we found the inverse of the matrix by doing QR factorization using Householder transformation.

Moderate sonar application requires the cross-spectral density matrix up to order 256. Present multi-core processor can meet the computational demands of the sonar.

6. Conclusions

The Intel MKL is part of a suite of tools offered by Intel to help developers create software efficiently and to achieve high performance in COTS systems. Here we discussed how the advent of multi-core systems and MKL will help high performance computing applications in sonar. An adaptive beamformer (MVDR) for broadband passive sonar is taken as a case study. The improvement on inverse computation of matrices implemented in Intel MKL library, for different orders, over a naive code implementation has been studied, which is the main computational bottleneck in the MVDR implementation.

ACKNOWLEDGEMENTS

Authors are immensely grateful to Director, NPOL and Dr. A. Unnikrishnan, Sc ‘G for their guidance and encouragement. We acknowledge the support provided by our division for the successful completion of the work.

References

[1]. A.D. Waite. SONAR for Practising Engineers, John Wiley & Sons, Ltd, 3rd Edition, 2002.

[2]. Richard O. Nielsen, Sonar Signal Processing, Artech House, 1991

[3]. Harry L. Van Trees, Optimum Array Processing. Part IV of Detection Estimation and Modulation Theory, John Wiley & Sons, 2002

[4]. http://www.developer.intel.com

[5]. Max Domeika, Software Development for Embedded Multi-core Systems – A Practical Guide Using Embedded Intel Architecture, Elsevier Inc. 2008

[6]. R. Hanson, F. Krogh, and C. Lawson. “A Proposal for Standard Linear Algebra Subprograms”. ACM SIGNUM Newsl., 8(16), 1973.

[7]. C. Lawson, R. Hanson, D. Kincaid, and F. Krogh “Basic Linear Algebra Subprograms for Fortran Usage”. ACM Transactions on Mathematical Software, 5(3):308–323, 1979.

[8]. Anderson, E., Bai, Z., Bischof, C., Blackford, L. S.,Demmel, J., Dongarra, J., DuCroz, J., Greenbaum, A., Hammarling, S., McKenny, A., and Sorenson, D. “LAPACK User’s Guide”. 3rd ed., SIAM, Philadelphia, PA., 1999.

[9]. Buttari1, A., Dongarra, J., Kurzak, J., Langou, J., Luszczek, P. and Tomov, S., “The Impact of Multicore on Math Software,” http://icl.cs.utk.edu/projectsfiles/sans/pubs/paramulticore-2006.pdf