

共有メモリ型並列計算機向け線形演算ライブラリにおける 並列化手法の評価

館野 諭司[†] 西村 成司^{††} 重原 孝臣[†] 長谷川 秀彦^{†††} 桧山 澄子[†]

[†]埼玉大学 ^{††}日本 SGI 株式会社 ^{†††}図書館情報大学

概要

本稿では、共有メモリ型並列計算機である HITACHI SR8000 1 ノード上において、連立一次方程式、固有値問題を例に NAG Fortran SMP ライブラリ、ベンダ提供の LAPACK の性能を比較し、それぞれのライブラリで採用されている並列化手法の特徴・問題点を明確にする。

Evaluation of Tuning Techniques for Parallel Processing on Shared Memory Parallel Computers

SATOSHI TATENO[†] SEIJI NISHIMURA^{††} TAKAOMI SHIGEHARA[†]
HIDEHIKO HASEGAWA^{†††} AND SUMIKO HIYAMA[†]

[†]SAITAMA UNIVERSITY. ^{††}SGI JAPAN LTD.

^{†††}UNIVERSITY OF LIBRARY AND INFORMATION SCIENCE.

Abstract

In this paper, we evaluate efficiency of tuning techniques for parallel processing by comparing the runtime performance of LAPACK with NAG Fortran SMP Library on 1 node of HITACHI SR8000, which is a representative shared memory parallel computer. For comparison, direct solution of linear systems and eigenvalue problems are considered.

1 はじめに

近年、多くのアプリケーションプログラムは演算処理・解析処理といった処理にライブラリを用いている。ライブラリ使用は、ユーザが自ら処理ルーチンを作成する手間が省けるという点だけでなく、精度の保証・汎用性といった面でもユーザの作成したルーチンより優れている。しかし、プログラムの実行に数十時間以上かかる大規模問題を扱うユーザやリアルタイム処理が必要なユーザにとっては、プログラムの実効性能が最も重要である。ライブラリの利用によりプログラムの実効性能が改善されると思われがちだが、すべての問題について実効性能が改善されるという保証はない。また、同一の処理を実現できるライブラリは一つとは限らず、施されているチューニングの差異により各ライブラリの実効性能は異なる。特に並列計算機では、ルーチン内部のどの部分をどのように並列化するかという並列化手法も性能に大きな影響を与える。

本研究では共有メモリ型並列計算機 HITACHI SR8000 1 ノード上において、並列化のアプローチの異なる線形演算ライブラリに対して性能評価を

おこない、それぞれのライブラリが採用している並列化手法の特徴・問題点を明らかにする。測定をおこなう問題は、連立一次方程式の直接解法、対称行列/非対称行列の固有値問題である。比較対象に選んだライブラリは、演算処理ルーチンの内部を重点的に並列化した LAPACK (Linear Algebra Package)[1, 2] および並列性のある部分的な処理を OpenMP[3] により並列化した The Numerical Algorithm Group 社の NAG Fortran SMP ライブラリ (以下、NAG と称す)[4] である。また、ライブラリの利用による性能向上の程度を調べるため、何もチューニングを施していないユーザ作成のプログラムとの性能比較もおこなう。

本稿の構成は以下の通りである。数値実験の計算環境および比較をおこなった各ライブラリの特徴を 2 節に示す。3 節では、まず実験方法および測定に用いたアルゴリズムを提示し、次に連立一次方程式、対称行列の固有値問題、非対称行列の固有値問題の順に実験データの提示・解析をおこない、各ライブラリの特徴を明らかにする。最後に 4 節にて本稿のまとめをおこなう。

表 1: コンパイラのバージョンとコンパイルオプション

Compiler	Compile Option	
Optimizing FORTRAN 90 Compiler V01-03-/A	-nolimit -noscope -64 -omp -save -O4 -parallel +SBTLB	
Optimizing FORTRAN 77 Compiler V01-03-/A	共通	-nolimit -noscope -64 -pvfunc=3
	8CPU	-W0,'OPT(O(4)),MP(P(3))' -procnum=8 -parallel
	1CPU	-W0,'OPT(O(4))' -noparallel
	NetlibBLAS 作成	-O4 -nosave -noparallel

表 2: ライブラリのバージョンとリンクするライブラリ

		Netlib	Vendor	NAG
バージョン		Version 3.0	V01-03	Release 2
ライブラリ	8CPU	並列版	並列版	NAG Fortran SMP ライブラリ
	1CPU	シングル版	シングル版	
BLAS	8CPU	並列版 BLAS		シングル版 BLAS
	1CPU	シングル版 BLAS		

2 計算環境

2.1 計算機のスペック

数値実験はすべて HITACHI SR8000 1 ノード (以下, SR8000 と称す) 上でおこなう。SR8000 は演算 CPU 8 台と制御 CPU 1 台で構成されており, 16GByte のメインメモリを各 CPU で共有する。各 CPU は PowerPC ベースであり, 二次キャッシュメモリを利用せずに高い性能を発揮する擬似ベクトル機構 [5, 6] を備えている。また, 各 CPU の一次キャッシュメモリは命令 64KByte, データ 128KByte であるが, 並列処理時には一次キャッシュメモリも共有され, 容量が CPU 台数倍に増加する。測定に用いた CPU 台数は 1 台および 8 台である。

プログラミング言語は FORTRAN77 を用いる。使用したコンパイラのバージョンとコンパイルオプションを表 1 に示す。NAG では OpenMP の使用が前提であるが, 今回使用した FORTRAN77 コンパイラは OpenMP をサポートしていない。そのため, NAG の測定には OpenMP をサポートする FORTRAN90 コンパイラを用いた。

2.2 各ライブラリの特徴

比較対象の線形演算ライブラリは, Netlib[7] で配布されている LAPACK (以下, *Netlib* と称す), ベンダ提供の HITACHI LAPACK (以下, *Vendor* と称す) および NAG である。各ライブラリとも核演算には BLAS (Basic Linear Algebra Subprograms) [8, 9, 10, 11, 12] を用いる。BLAS のチューニング

による性能の変化も調べるため, BLAS には Netlib で配布されている未チューニングの BLAS (以下, **NetlibBLAS** と称す) およびベンダがチューニングした BLAS (以下, **VendorBLAS** と称す) を用いる。BLAS には複数の CPU 上で並列処理される並列版と単一の CPU 上で単独処理されるシングル版が存在する。性能評価をおこなったライブラリのバージョンと 1CPU/8CPU においてリンクするライブラリ・BLAS のオブジェクトを表 2 に示す。

また, ライブラリの使用による性能改善の程度を調べるため, 標準的アルゴリズムに基づくユーザ作成のプログラム (以下, *User* と称す) も比較の対象に加える。

並列版 LAPACK (*Netlib*, *Vendor*)

並列化は主に局所的な演算処理 (BLAS ルーチン内部の処理等) に対しておこなわれる。*Netlib*, *Vendor* 共に, 並列処理をおこなうかどうかでライブラリが並列版とシングル版の 2 種類に分かれている。並列版ライブラリを利用する場合には BLAS も並列版をリンクする必要がある。また, チューニングは *Vendor* のみ施されており, *Netlib* はコンパイラによる最適化・自動並列化機能のみ用いて作成されている。*Netlib*, *Vendor* 共に, バージョンは LAPACK Version 3.0 相当である。

NAG Fortran SMP ライブラリ (NAG)

並列化は OpenMP を用いて並列性のある部分的な処理 (逐次性のないルーチンを繰り返し呼び出すループ等) に対しておこなわれている。NAG を利用するプログラムは並列版・シングル版の区別はなく, 並列処理に利用する CPU 台数を環境変数

に指定することで異なるオブジェクトを用意することなく CPU 台数に応じた並列処理を実現できる。また、並列化された処理の内部で呼び出される演算処理 (BLAS) は 1 つの CPU 上で処理されるため、BLAS はシングル版のみをリンクし 2 重に並列化されることを防ぐ。NAG には独自のアルゴリズムに基づくルーチンや LAPACK では提供されていない他の科学技術演算ルーチンも含まれている。今回はアルゴリズムによる性能差をできるだけ少なくするため LAPACK 互換のルーチンを測定に用いる。

ユーザ作成のプログラム (User)

User は、ループアンローリング、ブロック化等の特別なチューニングを何も施さず、標準的アルゴリズムにしたがって作成する。配列のメモリアクセスだけは FORTRAN77 の仕様に合わせてある。また、BLAS や OpenMP 等の並列化指示文は一切使用しない。並列化にはコンパイラの自動並列化機能のみを用いる。

3 実験結果

数値実験は、連立一次方程式、対称/非対称行列の固有値問題に対しておこなう。すべての問題において行列の大きさ n は $n = 5000$ である。性能評価にはプログラムの実行時間を用いる。OS の time コマンドを用いてプログラムの経過時間 (real) と各プロセッサの CPU 時間の合計 (user) を計測し、主に real の比較により性能を評価する。表 3~5 中、real の単位は時:分:秒、user の単位は秒である。また、Speed-Up は 1CPU の real を 8CPU の real で割った値であり、並列化による性能向上の割合を示す。なお、real と user を比較することによりシステムのオーバーヘッドがわかる。

3.1 連立一次方程式

本節では、ピボットの部分選択を伴う LU 分解による連立一次方程式の直接解法について議論する。係数行列は区間 $[0, 1)$ の一様乱数を成分に持つ密行列とする。User ではまず係数行列の LU 分解をおこない、前進消去後進代入により解を求める。LAPACK ルーチンは dgetrf, dgetrs を用いる。

表 3 に実験結果を示す。LU 分解後の求解部分の演算量 $[O(n^2)]$ は分解の演算量 $[O(n^3)]$ よりはるかに少なく、実質的に LU 分解に要する時間を反映している。VendorBLAS を用いた場合の real は 8CPU の場合 NAG が 0:00:18 と最も速く、次いで Vendor(0:00:23), Netlib(0:00:28) の順となっている。これらは User(0:03:21) よりも圧倒的に速い。1CPU の場合も同様に NAG(0:01:25), Vendor(0:05:31), Netlib(0:07:15), User(0:10:54) の順

で性能が良い。1CPU の結果から NAG は Vendor よりも高度なチューニングが施されていることがわかる。NAG の測定には 1CPU と 8CPU とで同一のオブジェクトを用いているため、LU 分解ルーチンに施されたチューニングは 8CPU の性能にも影響を与えている。このことから、OpenMP により並列性のある部分に対して並列化をおこなう NAG のアプローチは基盤となるルーチン (この場合 LU 分解のルーチン) のチューニングが重要であると考えられる。また、NetlibBLAS を用いた場合には NAG と Vendor, Netlib の性能差が縮まっていることから、BLAS のチューニング次第では NAG のアプローチが演算処理部 (主に BLAS) を重点的に並列化する Vendor, Netlib のアプローチに劣る可能性もある。

NetlibBLAS を用いた場合においても 8CPU の NAG, Vendor, Netlib の real はそれぞれ 0:01:11, 0:01:15, 0:01:15 と User(0:03:21) に勝る。これは、LAPACK 互換のライブラリでは LU 分解ルーチンにブロック化アルゴリズム [1, 13] が採用されアルゴリズムの並列性が一層高められているためである。これにより NAG では OpenMP による並列化がより効果的におこなわれていると考えられる。また、ブロック化アルゴリズムでは核演算にチューニング・並列化の効果が現れやすい Level 3 BLAS [12] が用いられているため、主に局所的な演算処理を並列化している Vendor, Netlib でも十分な性能向上が望める。

なお、Vendor, Netlib では Speed-Up が CPU 台数倍以上になる場合がある。これは 1CPU, 8CPU でリンクするライブラリのオブジェクトが異なるためである。並列版とシングル版とでは、想定されるプログラムの実行環境が異なり、ライブラリ/BLAS に施すチューニングの内容も異なる。特に SR8000 の並列処理では一次キャッシュメモリが共有され容量が増えるため、ワークメモリの取り方やメモリアクセスの仕方も当然異なるものと考えられる。

3.2 対称固有値問題

本節では、フランク行列を用いて実対称行列の全固有値・固有ベクトルの計算を評価する。 n 次フランク行列の (i, j) 成分は

$$a_{ij} = n - \max(i, j) + 1, \quad i, j = 1, \dots, n$$

である。ユーザ版では、ハウスホルダー変換により行列を三重対角化し、二分法と逆反復法により全固有値・固有ベクトルを求める。LAPACK ルーチンは、Relatively Robust Representation [14] による dsyevr, 分割統治法による dsyevd, QR 法による dsyev を用いる。NAG には dsyevd と同等のルーチンを用いる。これは、今回使用した NAG では dsyevr がまだ実装されていないためである。固

表 3: SR8000 $n = 5000$ での連立一次方程式の実行時間 (単位は real 時:分:秒, user 秒)

Library	VendorBLAS			NetlibBLAS		
	1CPU	8CPU	Speed-Up	1CPU	8CPU	Speed-Up
<i>NAG</i>	real 0:01:25 user 670	real 0:00:18 user 140	4.72	real 0:08:07 user 3870	real 0:01:11 user 140	6.86
<i>Vendor</i>	real 0:05:31 user 180	real 0:00:23 user 176	14.39	real 0:08:25 user 503	real 0:01:15 user 592	6.73
<i>Netlib</i>	real 0:07:15 user 276	real 0:00:28 user 213	15.35	real 0:08:22 user 499	real 0:01:15 user 591	6.69
<i>User</i>	real 0:10:54 user 653	real 0:03:21 user 1600	3.25			

表 4: SR8000 $n = 5000$ での対称固有値問題の実行時間 (単位は real 時:分:秒, user 秒)

Library	VendorBLAS			NetlibBLAS		
	1CPU	8CPU	Speed-Up	1CPU	8CPU	Speed-Up
<i>NAG</i>	real 0:09:05 user 4319	real 0:02:21 user 1107	3.87	real 0:33:55 user 16205	real 0:11:41 user 5550	2.90
<i>Vendor</i> <i>dsyevr</i>	real 0:14:01 user 594	real 0:01:27 user 670	9.67	real 0:35:09 user 2097	real 0:11:20 user 5384	3.10
<i>Vendor</i> <i>dsyevd</i>	real 0:18:29 user 705	real 0:01:24 user 642	13.20	real 0:43:16 user 2583	real 0:12:04 user 5752	3.59
<i>Vendor</i> <i>dsyev</i>	real 0:27:26 user 1472	real 0:04:43 user 2237	5.82	real 0:45:31 user 2710	real 13:40 user 6507	3.33
<i>Netlib</i> <i>dsyevr</i>	real 0:21:06 user 961	real 0:01:43 user 788	12.29	real 0:34:06 user 2030	real 0:10:47 user 5130	3.16
<i>Netlib</i> <i>dsyevd</i>	real 0:24:28 user 1043	real 0:01:39 user 764	14.83	real 0:42:20 user 2521	real 0:11:32 user 5495	3.67
<i>Netlib</i> <i>dsyev</i>	real 0:31:08 user 1687	real 0:04:54 user 2309	6.35	real 0:45:09 user 2688	real 0:13:11 user 6275	3.42
<i>User</i>	real 0:27:18 user 1631	real 0:04:14 user 2016	6.45			

有値の精度は相対残差が 10^{-12} 未満となるように設定した。

表 4 に実験結果を示す。VendorBLAS を用いた場合、8CPU の real は *dsyevd* を用いた *Vendor* が 0:01:24 と最も速く、*dsyevr* の *Vendor*(0:01:27)、*dsyevd* の *Netlib*(0:01:39)、*dsyevr* の *Netlib*(0:01:43)、*NAG*(0:02:21)、*User*(0:04:14) の順に性能が良い。しかし、1CPU では *NAG* が 0:09:05 と最も速く、*dsyevr* の *Vendor*(0:14:01)、*dsyevd* の *Vendor*(0:18:29)、*dsyevr* の *Netlib*(0:21:06)、*dsyevd* の *Netlib*(0:24:28)、*User*(0:27:18) の順に性能が良いという結果になった。この結果より、ライブラリ自体のチューニングは *NAG* が優れているが、並列処理においては局所的な演算を高速に処理できる *Netlib*、*Vendor*

のアプローチのほうが高い性能を発揮できることがわかる。また、*dsyev* を用いた *Vendor*、*Netlib* の real は 8CPU でそれぞれ 0:04:43、0:04:54 であり、*User* の性能よりも劣るという結果も得られた。これは *dsyev* の演算量が *dsyevr*、*dsyevd* よりも多く、アルゴリズムの並列性も低いためである。このことから固有値問題では演算量が少なく、並列性の高いアルゴリズムを用いることが重要であるといえる。

NetlibBLAS を用いた場合には、BLAS を用いる *NAG*、*Vendor*、*Netlib* の性能は大幅に低下し、8CPU の場合は *dsyevr* の *Netlib*(0:10:47) が最速、1CPU の場合は *NAG*(0:33:55) が最速となる。これらの性能は *User* よりも劣り、8CPU の場合には *User* の半分以下の性能しか発揮できていない。こ

これは、BLAS のチューニングが重要であることを示すと同時に、単に BLAS を用いるだけではなく、並列性の高い適切な計算アルゴリズムの利用およびアルゴリズムの改良も重要であることを示している。なお、NetlibBLAS を用いた場合の各ルーチンの性能は Netlib の性能が同じルーチンを用いた Vendor を上回るという点を除き VendorBLAS を用いた場合と同様の傾向を示した。

3.3 非対称固有値問題

本節では、非対称行列の固有値問題について検討する。固有値は全て求めるが、固有ベクトルは求めない。非対称固有値問題の一例として、実験には

$$b_{ij} = \text{mod}(n + j - i, n) + 1, \quad i, j = 1, \dots, n$$

を (i, j) 成分とする巡回行列を用いた。User では、ピボット選択つきガウス消去法と同等の手順により係数行列をヘッセンベルグ行列へ変換し、シフト付き QR 法により全固有値を求める [15]。Vendor, Netlib の LAPACK ルーチンは dgeev を使用する。NAG には dgeev に相当するルーチンが実装されていないため、dgeev 内部で呼び出されるヘッセンベルグ行列への変換ルーチン (dgehrd 相当)、ヘッセンベルグ行列に対して QR 法により固有値を求めるルーチン (dhseqr 相当) を組み合わせて測定をおこなう。dgeev では、ヘッセンベルグ行列への変換にハウスホルダー変換を用いているため、変換には User の約 2 倍の演算量が必要となる [15]。NAG のヘッセンベルグ行列への変換ルーチン内でどのようなアルゴリズムが用いられているかは不明である。また、NAG では非対称行列の固有値問題に対するルーチンは並列化されていない [4]。固有値の精度は相対残差が 10^{-12} 未満となるように設定した。なお dgeev では、固有値の収束を速めるため、ヘッセンベルグ型に変換する前にバランシング [1] をおこなうが、実験に用いた巡回行列ではバランシングは効かない。

表 5 に実験結果を示す。VendorBLAS, NetlibBLAS とともに NAG の Speed-Up が 1.00 となっているが、これは NAG のルーチンが並列化されておらず、8CPU の並列処理時においても 1CPU と同じ処理をおこなってしまうためである。NAG を除いた VendorBLAS 8CPU の real は、User の性能が 0:08:16 と最も良く、Vendor が 0:13:12 と、Netlib(0:10:01) に劣るという結果を得た。NAG は VendorBLAS 1CPU において 0:24:31 であり、User(0:25:13) よりわずかに優れる。VendorBLAS 1CPU における Vendor, Netlib の real はそれぞれ 1:00:07, 0:47:05 であり、User よりも劣る。ヘッセンベルグ行列への変換に要する演算量の違いを考慮に入れても、非対称固有値問題においてはライブラリの使用により性能が改善されるとはいえない。

また、NetlibBLAS を用いた場合の real は、8CPU の場合 Vendor は 0:22:41, Netlib は 0:20:02, 1CPU の場合 NAG は 0:48:35, Vendor は 1:24:47, Netlib は 1:12:39 となり、VendorBLAS を用いた場合よりも 1.5 倍～2 倍程度遅くなる。連立一次方程式や対称固有値問題の場合、VendorBLAS を用いることで 8CPU の場合 2.5 倍以上、1CPU の場合 1.5 倍以上性能が向上することを考慮に入れると、非対称固有値問題では BLAS のチューニングだけでは大幅な性能向上は期待できないといえる。

4 まとめ

本稿では並列化をおこなう手法の違いに着目し、SR8000 1 ノード上で NAG Fortran SMP ライブラリ、ベンダ提供の LAPACK, Netlib 上の LAPACK および自動並列化機能を利用したユーザ作成プログラムについて性能比較をおこない、各ライブラリで用いられている並列化手法の特徴・問題点を明らかにした。数値実験の結果より、NAG Fortran SMP ライブラリのように OpenMP により並列性がある部分の処理を重点的に並列化する方法は、連立一次方程式の場合のように基盤となるルーチンが十分にチューニングされ適度な並列性を備えているようなアルゴリズムに対しては有効であるが、固有値問題のように並列性が低いアルゴリズムでは並列化の効果が十分に得られないことが明らかとなった。また、ベンダ提供の LAPACK のように局所的な演算処理を重点的に並列化する方法は、アルゴリズムの並列性が低い場合でも十分な効果を得られるが、性能の大部分が演算処理すなわち BLAS のチューニングによるものであるため、アルゴリズムを改良しなければ大幅な性能改善が期待できないことが明らかとなった。非対称行列の固有値問題の結果は、並列性の低いアルゴリズムはプログラムのチューニング・並列化による性能改善が困難であり、アルゴリズムの改良もしくは新しいアルゴリズムの開発が必要であるということを示している。

参考文献

- [1] Anderson, E., Bai, Z., Bischof, C., Blackford, S., Demmel, J., Dongarra, J. J., DuCroz, J., Greenbaum, A., Hammarling, S., McKenney, A. and Sorensen, D.: *LAPACK Users' Guide (3rd Ed.)*, SIAM (2000).
- [2] 山本喜一, 榊原進, 野寺隆志, 長谷川秀彦: これだけは知っておきたい数学ツール, 共立出版 (1999).
- [3] OpenMP Architecture Review Board: <http://www.OpenMP.org/>.

表 5: SR8000 $n = 5000$ での非対称固有値問題の実行時間 (単位は real 時:分:秒, user 秒)

Library	VendorBLAS			NetlibBLAS		
	1CPU	8CPU	Speed-Up	1CPU	8CPU	Speed-Up
NAG	real 0:24:31 user 1464	real 0:24:33 user 1464	1.00	real 0:48:35 user 2903	real 0:48:43 user 2903	1.00
Vendor	real 1:00:07 user 3598	real 0:13:12 user 6271	4.55	real 1:24:47 user 5074	real 0:22:41 user 10830	3.74
Netlib	real 0:47:05 user 2817	real 0:10:01 user 4765	4.70	real 1:12:39 user 4347	real 0:20:02 user 9566	3.63
User	real 0:25:13 user 1507	real 0:08:16 user 3942	3.05			

- [4] NAG Fortran SMP Library Documentation: <http://www.nag.co.uk/numeric/FL/manual/html/FSlibrarymanual.asp> .
- [5] Nishiyama, H., Motokawa, K., Kyushima, I. and Kikuchi, S.: Pseudo-vectorizing Compiler for the SR8000, *Proc. Euro-Par 2000, LNCS 1900*, Springer-Verlag, pp. 1023–1027 (2000).
- [6] Brehm, M., Bader, R., Heller, H. and Ebner, R.: Pseudovectorization, SMP, and Message Passing on the Hitachi SR8000-F1, *Proc. Euro-Par 2000, LNCS 1900*, Springer-Verlag, pp. 1351–1361 (2000).
- [7] Netlib: <http://www.netlib.org/>.
- [8] 小国力, 村田健郎, 三好俊郎, Dongarra, J. J., 長谷川秀彦: 行列計算ソフトウェア, 丸善 (1991).
- [9] Dongarra, J. J., Duff, I. S., Sorensen, S. C. and van der Vorst, H. A.: *Solving Linear Systems on Vector and Shared Memory Computers*, SIAM (1991).
- [10] Lawson, C., Hanson, R., Kincaid, D. and Krogh, F.: Basic Linear Algebra Subprograms for FORTRAN Usage, *ACM Trans. Math. Software*, Vol. 5, pp. 308–325 (1979).
- [11] Dongarra, J. J., DuCroz, J., Hammarling, S. and Hanson, R.: An Extended Set of FORTRAN Basic Linear Algebra Subprograms, *ACM Trans. Math. Software*, Vol. 14, pp. 1–17 (1988).
- [12] Dongarra, J. J., DuCroz, J., Duff, I. S. and Hammarling, S.: A Set of Level 3 Basic Linear Algebra Subprograms, *ACM Trans. Math. Software*, Vol. 16, pp. 1–17 (1990).
- [13] Dongarra, J. J., Duff, I. S., Sorensen, D. C. and van der Vorst, H. A.: *Numerical Linear Algebra for High-Performance Computers*, SIAM (1998).
- [14] Dhillon, I. S.: *A New $O(n^2)$ Algorithm for Symmetric Tridiagonal Eigenvalue/Eigenvector Problem*, PhD Thesis, University of California at Berkeley (1997).
- [15] Press, W. H., Teukolsky, S. A., Vetterling, W. T. and Flannery, B. P.: *Numerical Recipes in FORTRAN Second Edition*, Cambridge University Press (1992).