

Performance of a New Parallel Eigensolver PDSYEVr on HPCx

Andrew Sunderland
Computational Science and Engineering Department,
CCLRC Daresbury Laboratory,
Warrington, UK

Email: a.g.sunderland@dl.ac.uk

Abstract

This report investigates the performance of a new pre-release ScaLAPACK eigensolver routine PDSYEVr. Performance comparisons are made against existing ScaLAPACK eigensolver routines for matrices with differing characteristics from different high performance computing application codes. It is expected that this technical report will be updated frequently over the coming months as new versions of the PDSYEVr code are developed and then tested on HPCx.

This is a Technical Report from the HPCx Consortium.

Report available from <http://www.hpcx.ac.uk/research/publications/HPCxTR0610.pdf>

© UoE HPCx Ltd 2006

Neither UoE HPCx Ltd nor its members separately accept any responsibility for loss or damage arising from the use of information contained in any of their reports or in any communication about their tests or investigations.

Performance of a New Parallel Eigensolver PDSYEVr on HPCX

1	Introduction	3
2	Background Information	3
2.1	Solving the Symmetric Eigenvalue Problem	3
2.1.1	Definition	4
2.1.2	Algorithms for Computing Eigenvalues and Eigenvectors	4
2.1.3	The MRRR algorithm for Solving the Symmetric Tridiagonal Eigenproblem	5
2.2	The new ScaLAPACK routine PDSYEVr	5
2.2.1	Memory Requirements of PDSYEVr	6
3	Test Matrices	6
3.1	Constructed Matrices	6
3.2	Application Matrices	7
3.3	Compilation	7
4	Results	7
4.1	Calculation of a subset of Eigenpairs	7
4.1.1	Constructed Matrices	8
4.2	Calculation of All Eigenpairs	9
4.2.1	Constructed Matrices	9
4.2.2	Application Matrices	9
5	Conclusions	12
6	Acknowledgements	12
7	References	12
	Appendix A	14
	Appendix B	16
7.1	GAMESS-UK	16
7.2	CRYSTAL	16

1 Introduction

Eigenvalue and eigenvector computations arise in a wide range of scientific and engineering applications. For example, in quantum chemistry the computation of eigenvalues may be required in order to calculate electronic energy states whilst in engineering the eigenvalues of a system may represent the natural frequencies of vibration of a structure. For large-scale application codes such as these, a highly efficient algorithm for calculating the eigenvalues and eigenvectors of a matrix in parallel is essential if fast, scalable performance is to be achieved. In the 90s, Dhillon and Parlett devised a new algorithm (Multiple Relatively Robust Representations MRRR) for computing numerically orthogonal eigenvectors of a symmetric tridiagonal matrix with $O(n^2)$ cost [1]. A performance analysis of the implementation of the MRRR algorithm in the LAPACK library was undertaken previously [2] and initial results were promising. Recently a ScaLAPACK [3] implementation of this algorithm for real (double precision) symmetric matrices named PDSYEVr has been developed [4] and it is planned that this routine will be incorporated into future releases of ScaLAPACK. The routine has a standard ScaLAPACK interface and therefore users should be able to swap eigensolvers within their code with minimum disruption. This report tests this initial pre-release implementation of PDSYEVr against the existing ScaLAPACK eigensolvers PDSYEVd, PDSYEVx and PDSYEV on a range of test matrices. It should be emphasised that this report refers to a pre-release 'snapshot' of the current PDSYEVr routine under development, and the parallel performance and some properties of the code are likely to change between now and the official release date. At the time of writing the code developers have recently provided the author with a full suite of xxYEVr routines for double precision real, single precision real, double complex Hermitian and single complex hermitian matrices. However, the focus within this report will be on the double precision real version PDSYEVr and the alternative established double precision real eigensolvers available in ScaLAPACK.

2 Background Information

2.1 Solving the Symmetric Eigenvalue Problem

Many scientific application codes under development involve the computation of the symmetric real or Hermitian eigenvalue problem. In particular this diagonalization stage in parallel quantum chemistry codes often consumes a significant percentage of overall run time. It is therefore particularly important that efficient parallel algorithms and implementations are used. Computational chemists generally require all or a subset of the spectrum and its associated eigen-subspaces. For most applications the solution of the eigenproblem is limited by available processing time and computer memory. A useful summary of methods for tackling the symmetric eigenvalue problem is given in [1].

2.1.1 Definition

The Standard Eigenvalue Problem is described as

$$\mathbf{Ax} = \lambda \mathbf{x},$$

where \mathbf{A} is a matrix and λ is the eigenvalue corresponding to eigenvector \mathbf{x} .

2.1.2 Algorithms for Computing Eigenvalues and Eigenvectors

The solution to the real or hermitian dense symmetric eigensolver problem usually takes place via three main steps

1. Reduction of the matrix to tri-diagonal form, typically using the Householder Reduction [8].
2. Solution of the real symmetric tri-diagonal eigenproblem via one of the following methods:
 - Bisection for the eigenvalues and inverse iteration for the eigenvectors [9] [10], employed within ScaLAPACK routine PDSYEVX
 - QR algorithm [11], employed within real symmetric driver ScaLAPACK driver routine PDSYEVr,
 - Divide & Conquer method (D&C) [12], employed within real symmetric driver ScaLAPACK driver routine PDSYEVd,
 - Multiple Relatively Robust Representations (MR³ algorithm) [1], employed within real symmetric driver ScaLAPACK driver routine PDSYEVr.
3. Back transformation to find the eigenvectors for the full problem from the eigenvectors of the tridiagonal problem.

Steps 1. and 3. are common to all the routines analysed here. For step 1. employing Householder transformations to reduce the original matrix to tridiagonal form costs about $4/3n^3$ operation. Back-transformation in Step 3. requires $2n^3$ operations. The cost of solving the tridiagonal eigenproblem in Step 2. varies from $O(nk)$ to $O(kn^3)$ ($4 < k < 12$) according to the method used and the numerical characteristics of the matrix, where n is the dimension of the matrix and k is the number of required eigenpairs. Moreover, Steps 1 and 3 can exploit highly-optimised matrix-multiply operations whereas Step 2 generally cannot. Therefore the tridiagonal eigensolve usually represents the computational bottleneck in the solution of eigenproblems and it is this stage of the calculation that has received the most attention from developers of eigensolver algorithms. Four alternative approaches to the tridiagonal eigensolve are listed above and the performance of the ScaLAPACK parallel implementations on HPCx for generated matrices and application matrices are investigated here.

2.1.3 The MRRR algorithm for Solving the Symmetric Tridiagonal Eigenproblem

The MR³ algorithm applies a shift σ to T near each cluster of eigenvalues and computes an LDL^T factorization of $(T - \sigma I)$. A suitable choice of σ eliminates the need for costly re-orthogonalizations when the eigenvalues are tightly clustered. The method is based upon the property that small relative changes in entries of L and D cause small relative changes in each eigenvalue of T . To summarise briefly, the MRRR approach is to factor the tridiagonal matrix into a LDL^T form where (L, D) is a 'relatively robust representation' (RRR) for its eigenvalue subset, meaning that small relative changes in L and D cause small relative changes in each eigenvalue. Given an RRR for a set of eigenvalues, relatively well-spaced eigenvalues are calculated immediately, whilst relatively clustered eigenvalues can be shifted and a new RRR formed. The process then repeats until all eigenvalues have been found.

In theory the algorithm requires $O(nk)$ operations, where k is the number of eigenpairs to be found. This computational requirement is much lower than the other methods listed. The MR³ algorithm also has the added advantage of requiring in substantially lower amounts of workspace, thereby increasing the maximum size of system that can be solved. A serial version is available as the routine xSTEGR in LAPACK, and a beta-version parallel implementation is available in PLAPACK [13] upon request. The performance of the routine in PLAPACK has already been examined in an earlier HPCx Technical Report [2].

2.2 The new ScaLAPACK routine PDSYEVX

The ScaLAPACK MRRR driver routine PDSYEVX includes the following routines for the three operations listed above

- 1) Transform full matrix into tridiagonal form (PDSYNTRD)
- 2) Compute the eigenpairs of the tridiagonal matrix:
 - a. Compute root representations and root eigenvalues (DSTEGR2A)
 - b. Share eigenvalues between processors (DGEBS2D & DGEBR2D)
 - c. Compute representation tree and eigenvectors (DSTEGR2B, DLARRV)
- 3) Compute the back transformation (PDORMTR)

As with the PDSYEVX routine, PDSYEVX is an *expert* ScaLAPACK driver, in that a user-defined subset of eigenvalues and/or eigenvectors can be computed. This is in contrast to the *simple* ScaLAPACK drivers PDSYEV and PDSYEVX, where all the eigenvalues and/or eigenvectors are always computed. The implementation and features of the new implementation are discussed in detail in [4].

2.2.1 Memory Requirements of PDSYEVr

At present the memory requirement of PDSYEVr is around $4n^2$ when all eigenpairs are required. These memory overheads are governed by the requirement that all ScaLAPACK routines use a 2D block-cyclic matrix distribution in order to maintain load balancing and optimise performance. It has been proposed that future memory requirements could be reduced by up to 50% by employing an alternative 1D block-cyclic layout for the matrix. Scalapack developers are currently testing the feasibility of this approach.

3 Test Matrices

The relative performance of the eigensolver routines is normally dependent upon the matrix characteristics, in particular the distribution of eigenvalues within the eigenvalue spectrum. Therefore PDSYEVr and other established ScaLAPACK solvers are tested here on a suite of matrices with widely differing eigenvalue spectrums. All matrices investigated are double precision real symmetric.

3.1 Constructed Matrices

A suite of test matrices is constructed in order to test the eigensolver routines for a range of eigenvalue spectrums. The real dense symmetric matrices are constructed according to the procedure outlined in [7]. A standard matrix dimension of $N=4000$ has been chosen here as this dimension is large enough to be generally representative of application matrices (see Section 3.2) yet small enough to allow many different tests to be applied within the same batch job on HPCx.

Name	Description
M-even	Eigenvalues are evenly distributed along the diagonal, with values ranging from -100 to $+100$.
M-rand	Eigenvalues are randomly distributed along the diagonal, with values between -100 and $+100$. (Each solver uses the same random matrix).
M-7cluster	Eigenvalues are formed in 7 clusters. Eigenvalues within a cluster are distributed within the range $-1.e-5$ and $1.e-5$.
M-1cluster	Eigenvalues are formed in 7 clusters. Eigenvalues within a cluster are distributed within the range $-1.e-5$ and $1.e-5$.

3.2 Application Matrices

Name	Description
M-CRY3888	Three matrices of differing dimensions obtained from Hartree-Fock calculations. Eigenvalues exhibit tightly clustered properties
M-CRY7194	
M-CRY12534	
M-GAM3790	Matrix from an isocitrate lysase molecule calculation undertaken by GAMESS-UK. Eigenvalues exhibit tightly clustered properties.

The matrices and their associated applications are described in earlier HPCx Technical Reports [2], [5]. A brief description of the Crystal and GAMESS-UK packages is also given in Appendix B in this report.

3.3 Compilation

All the results reported were obtained from Phase2a of HPCx – a tightly coupled HPC system with 96 16-way IBM p5-575 compute nodes, each with 32 Gbytes of memory, and an HPS interconnect. The Fortran compiler used is `mpxlf95_r` with options `-q64 -O3 -qsuffix=f90 -qstrict=auto -qtune=auto`.

4 Results

4.1 Calculation of a subset of Eigenpairs

The ScaLAPACK routines PDSYEVr and PDSYEV are excluded from the tests below as these ‘standard’ driver routines do not have the option to calculate a subset of the eigenpairs. In order to evaluate the performance of the two expert ScaLAPACK drivers, PDSYEVr and PDSYEVX, the subset of eigenpairs to be found is varied from between 10% to 100% of the total (N). The default settings described in the ScaLAPACK documentation [3] are used throughout for PDSYEVX.

4.1.1 Constructed Matrices

Matrix	Eigenpairs Found	Time to soln. PDSYEVX (secs)		Time to soln. PDSYEVX (secs)	
		Processors		Processors	
		16	256	16	256
M-even	4000	10.14	4.67	431.2	439.3
	2000	8.08	3.60	110.3	108.8
	1000	6.81	3.33	28.94	26.68
	400	6.28	3.07	6.52	4.80
M-rand	4000	10.71	3.80	12.98	4.84
	2000	8.26	3.68	9.93	4.19
	1000	7.18	3.29	8.66	4.12
	400	6.36	3.22	8.54	3.99

Table 1. Performance comparison for calculation of a subset of eigenpairs

The timing results above demonstrate that PDSYEVX is faster in all cases when calculating a subset of eigenvalues. The PDSYEVX results for **M-even** are particularly poor, as when computing with the routine's default settings, the eigenvalues here are all regarded to lie within one large 'super-cluster'. This characteristic of this matrix requires PDSYEVX to undertake many additional operations involving much computation and communication in order to maintain orthogonality of the associated eigenvectors. In the case of **M-rand**, some re-orthogonalisation of eigenvectors is also required, but the eigenvalue spectrum is now more fragmented with multiple clusters of eigenvalues separated by relatively large gaps.

4.2 Calculation of All Eigenpairs

4.2.1 Constructed Matrices

In many applications, the full set of eigenvalues and orthogonal eigenvectors are required. The table below shows timings from HPCx for PDSYEV compared against the established ScaLAPACK eigensolver routines PDSYEV (divide-and-conquer based) and PDSYEV (QR based) for the suite of constructed matrices.

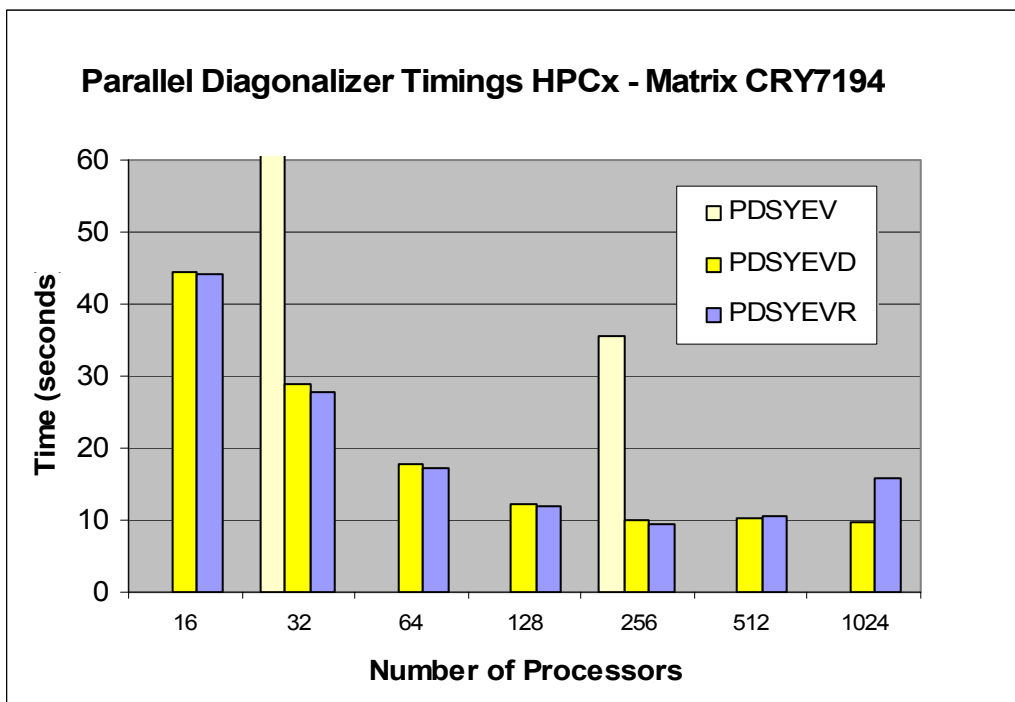
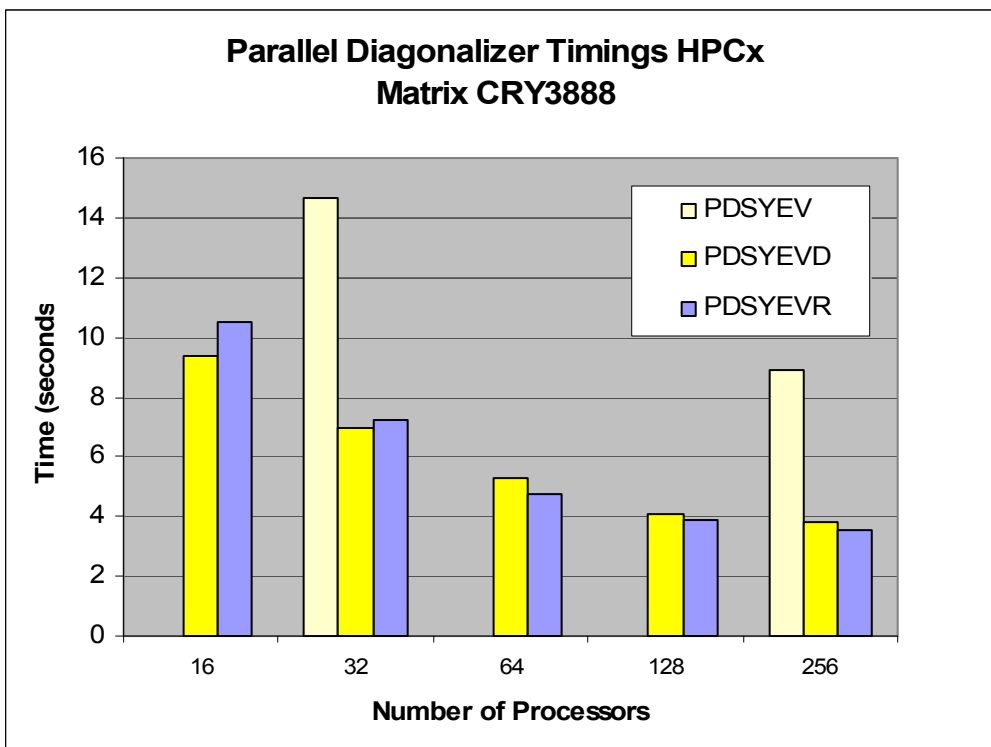
Matrix (N=4000)	Time to soln. PDSYEV (secs)				Time to soln. PDSYEV (secs)				Time to soln. PDSYEV (secs)			
	Processors				Processors				Processors			
	16	32	64	128	16	32	64	128	16	32	64	128
M-Even	10.14	6.95	4.36	3.79	10.72	7.11	4.60	3.85	26.30	15.38	9.62	7.56
M-Rand	10.71	6.84	4.32	3.90	10.88	7.15	4.56	3.83	25.53	15.11	9.78	7.49
M-7cluster	11.70	7.45	4.51	3.99	10.51	7.05	4.51	3.81	22.43	14.41	9.38	7.12
M-1cluster	11.56	7.40	4.47	3.97	10.73	7.02	4.56	3.92	21.94	12.96	8.19	6.38

Table 2. HPCx performance comparison for calculating all eigenpairs (constructed matrices)

The timings from the table above demonstrate that the performance of PDSYEV is equivalent to PDSYEV and much superior to that of PDSYEV for all the matrices tested here. For the **M-Even** and **M-Rand** matrices where eigenvalues are generally not closely bound, the performance of PDSYEV is marginally better than that of PDSYEV. However, for the **M-7cluster** and **M-1cluster** matrices where the eigenvalues are tightly clustered, PDSYEV performs better as it can take advantage of a 'deflation' algorithm specifically designed for tight clusters of eigenvalues [12]. The scaling graphs for these tests are shown in Appendix A.

4.2.2 Application Matrices

The parallel scaling curves from HPCx for PDSYEV, PDSYEV and PDSYEV are shown below for the eigensolves of the Crystal and Gamess-UK matrices. Here eigenvalues are very tightly bound – typical spacing is around 10^{-5} within clusters – therefore PDSYEVX is unsuitable for these cases and is not tested here.



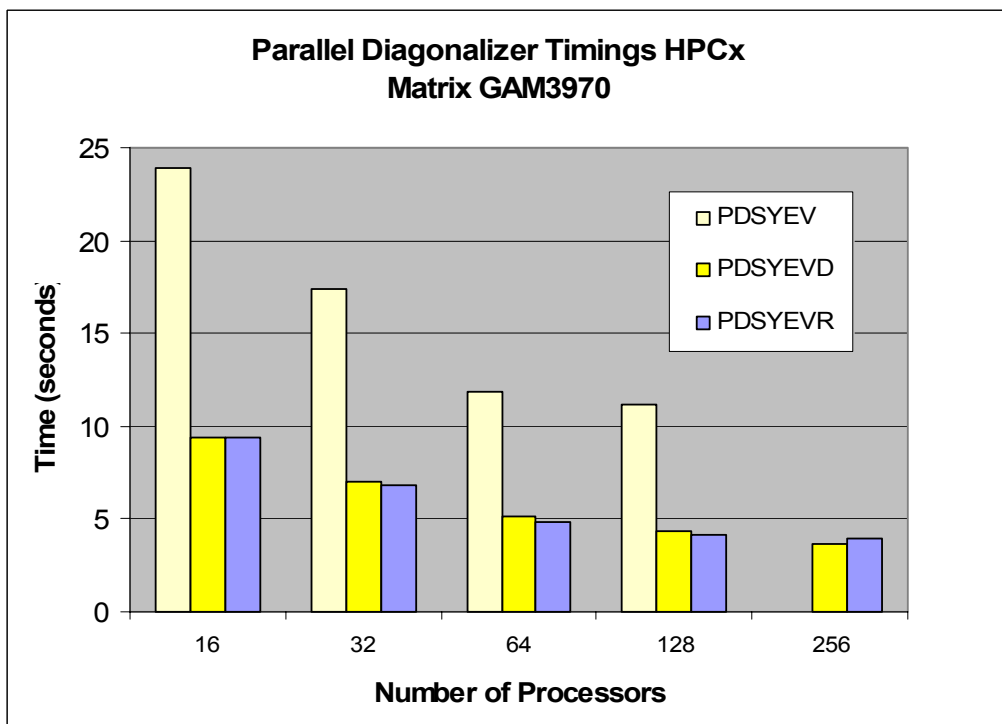
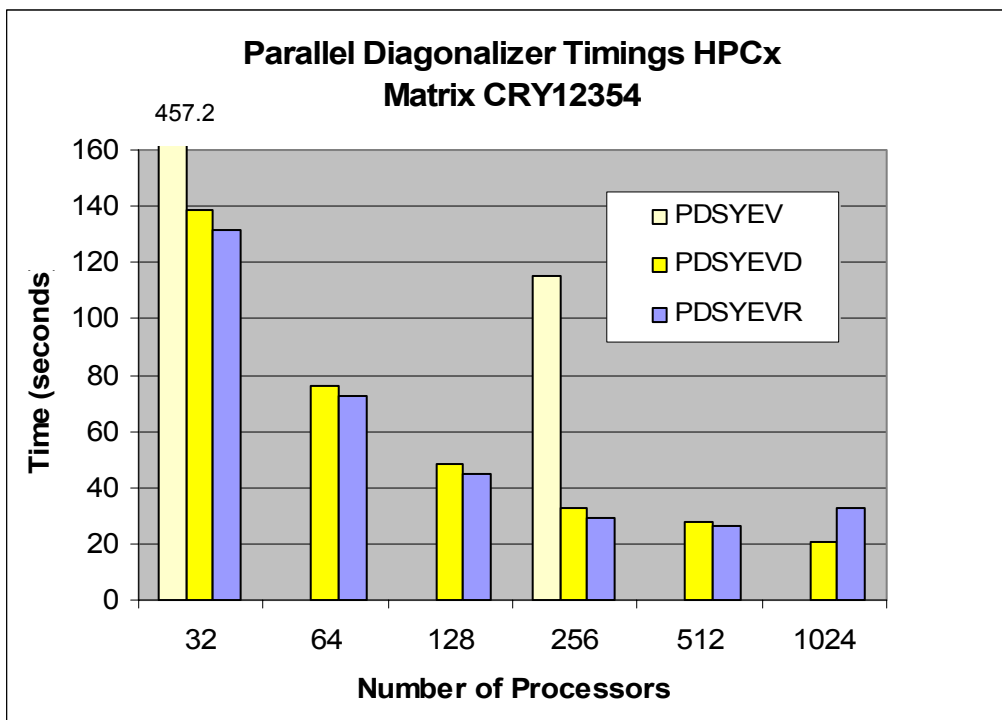


Figure 1. Performance comparison of ScaLAPACK eigensolvers for the application matrices

The results above demonstrate that PDSYEVR performance is usually superior to PDSYEVD, albeit by a relatively small margin, for all the application matrices tested here. The only exceptions are runs undertaken on the very largest numbers of processors where some degradation in performance is observed for PDSYEVR. The PDSYEVR timings are impressive, as the characteristic of tightly clustered eigenvalues, common to the application matrices tested here, is particularly suited to the *deflation* algorithm in PDSYEVD [12].

5 Conclusions

Initial testing of the new ScaLAPACK routine PDSYEVR routine on HPCx has shown that its performance is very good for a range of test matrices. The routine exhibited robustness throughout and consistently produced accurate results for all the test cases investigated here. The performance of the new routine matched and sometimes exceeded that of PDSYEVD, even for cases evidently advantageous to PDSYEVD's deflation algorithm (Table 2. & Figure 1.) , and there are potentially large savings with associated memory overheads. PDSYEVR also is shown to be highly suited to applications where only a subset of eigenpairs is required, particularly for matrices with relatively tightly clustered eigenvalues (Table 1.). It is these cases where the parallel performance of PDSYEVX is often very poor due to its reliance on reorthogonalization methods for the eigenvectors. Further testing of xxYEVR routines on matrices from several more HPCx applications is currently underway and results and conclusions from these tests will be added to this report shortly.

6 Acknowledgements

The author is indebted to Christof Voemel from the University of Berkeley for providing up-to-date pre-release versions of the PDSYEVR routine for testing. Many thanks also goes to Stanko Tomic and Ian Bush at CCLRC Daresbury Laboratory for providing several of the application matrices.

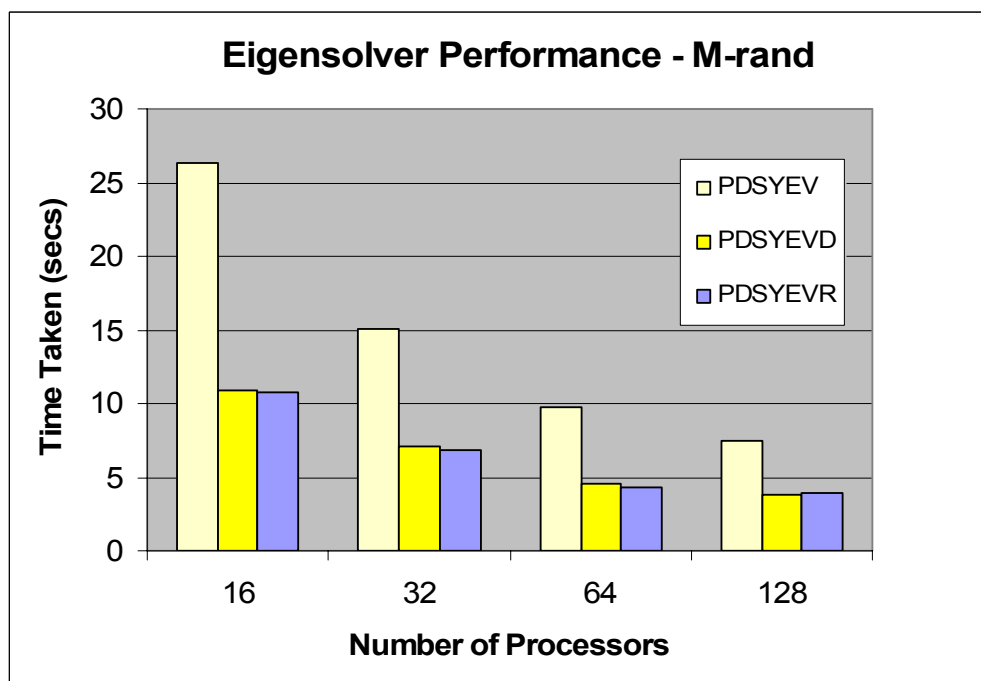
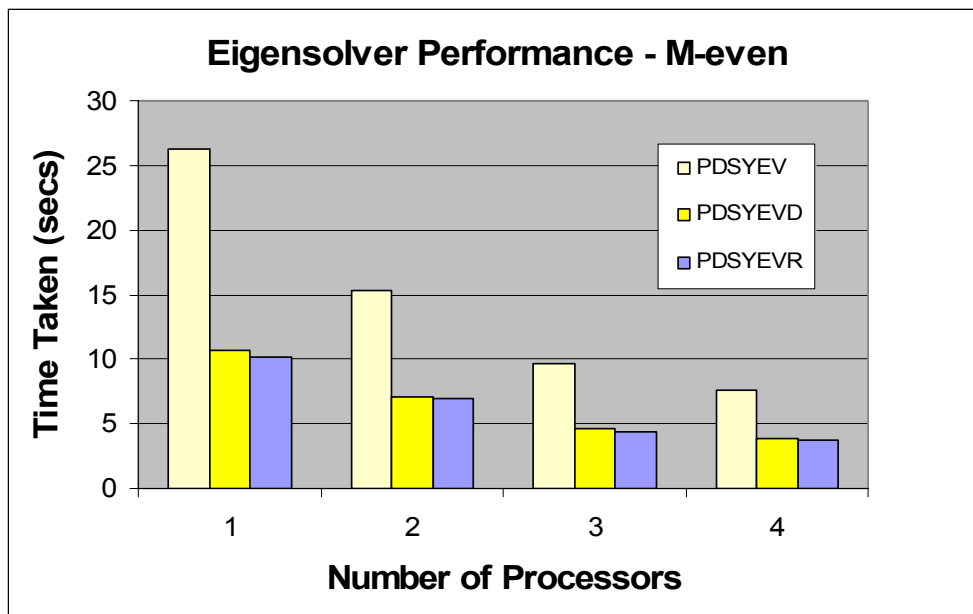
7 References

- [1] *A Parallel Eigensolver for Dense Symmetric Matrices based on Multiple Relatively Robust Representations*, P.Bientinesi, I.S.Dhillon, R.A.van de Geijn, UT CS Technical Report #TR-03026, 2003, <http://www.cs.utexas.edu/users/plapack/papers/pareig.ps>.

-
- [2] *A Performance Study of the PLAPACK and ScaLAPACK Eigensolvers on HPCx for the Standard Problem*, E. Breitmoser, A.G. Sunderland, HPCx Technical Report HPCxTR0406, 2004, <http://www.hpcx.ac.uk/research/hpc/HPCxTR0406.pdf>.
- [3] www.netlib.org/scalapack/scalapack_home.html
- [4] *"PDSYEVr. ScaLAPACK's parallel MRRR algorithm for the symmetric eigenvalue problem"*, D. Antonelli, C. Vomel, Lapack working note 168, (2005). <http://www.netlib.org/lapack/lawnspdf/lawn168.pdf>
- [5] *Using Scalable Eigensolvers on HPCx: Case Studies*, I.J.Bush, A.G.Sunderland, G. Pringle, HPCx Technical Report HPCxTR0510, 2005, <http://www.hpcx.ac.uk/research/hpc/HPCxTR0510.pdf>.
- [6] *An Overview of Eigensolvers for HPCx*, A.G.Sunderland, E. Breitmoser, HPCx Technical Report HPCxTR0312, 2003, <http://www.hpcx.ac.uk/research/hpc/HPCxTR0312.pdf>.
- [7] *Performance of Software for the Full Symmetric Eigenproblem on CRAY T3E and T90 Systems*, R. Gutheil, R. Zimmerman, Zentralinstitut für Angewandte Mathematik (ZAM) Technical Report FZJ-ZAM-IB-2000-07, July 2000, <http://www.fz-juelich.de/zam/docs/autoren2000/Reports2000.html>.
- [8] *"Numerical Recipes in Fortran 77, 2nd Edition"*, B. P. Flannery, W. H. Press, Saul A. Teukolsky, W. T. Vetterling, pages 449-490, Cambridge University Press (1996)
- [9] *"The numerical computation of the characteristic values of a real symmetric matrix"*, Wallace J. Givens, Technical Report ORNL-1574, Oak Ridge National Laboratory, Oak Ridge, TN, USA, (1954)
- [10] *"The calculation of specified eigenvectors by inverse iteration, contribution II/18"*, volume II of Handbook of Automatic Computation, pages 418-439, Springer-Verlag, New York, Heidelberg, Berlin, (1971).
- [11] *"The QR transformation, parts I and II"*, Computer J., 4:265-271, 332-345, (1961-62)
- [12] *"A parallel divide and conquer algorithm for the symmetric eigenvalue problem on distributed memory architectures"*, F.Tisseur and Jack Dongarra, SIAM J. SCI. COMPUT, Vol.20, No. 6, pp. 2223-2236 (1999)
- [13] *"Plapack: Parallel Linear Algebra Package"*, The University of Texas at Austin, <http://www.cs.utexas.edu/users/plapack/>

Appendix A

Scaling graphs for data from Table 1.



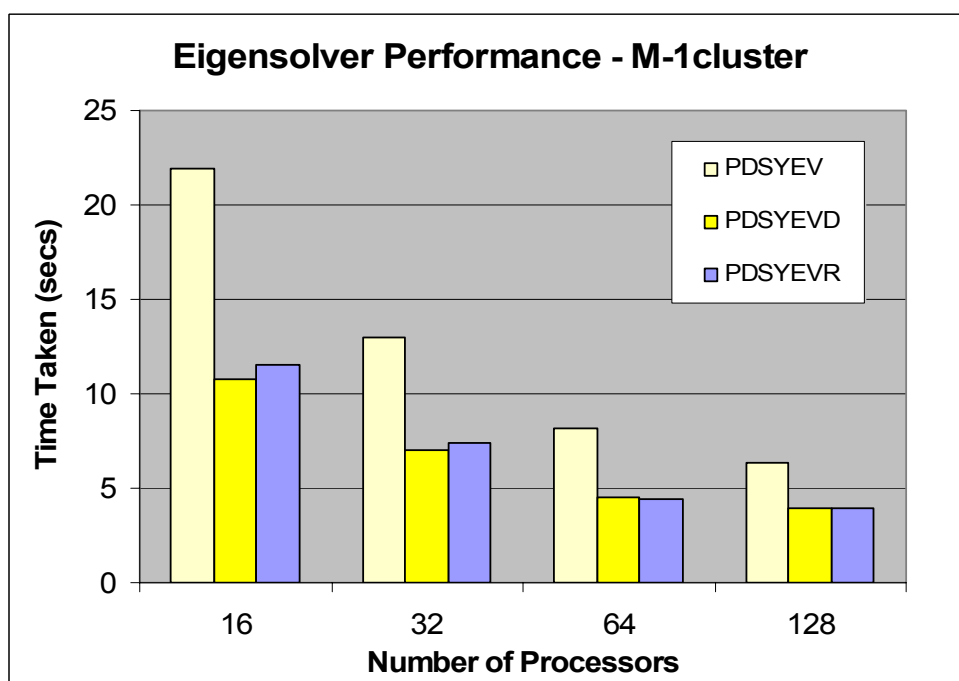
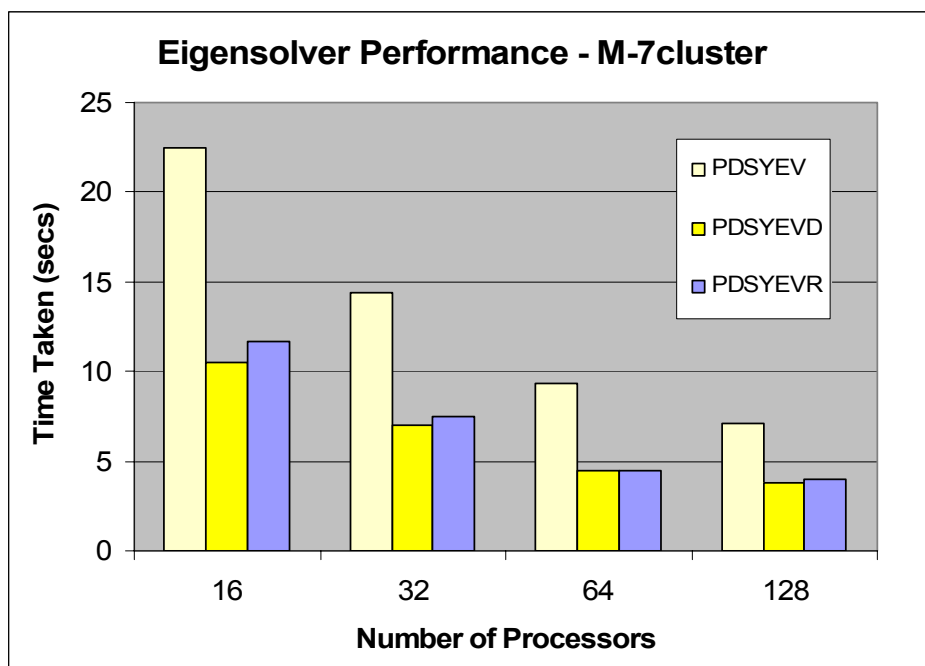


Figure 2. Performance of the ScaLAPACK eigensolvers for the constructed matrices

Appendix B

7.1 GAMESS-UK

GAMESS-UK represents a typical established electronic structure code, comprising some 800K lines of Fortran that permits a wide range of computational methodology to be applied to molecular systems. Two of the most commonly used methods, Hartree-Fock (HF) and Density Functional Theory (DFT), are iterative methods that require the diagonalization of at least one matrix at each step of the algorithm. These diagonalizations are the limiting factor in the scalability of the method.

GAMESS-UK has two parallel implementations of HF and DFT. The first, and earlier, implementation uses the Global Array (GA) toolkit from the Environmental Molecular Sciences Laboratory at PNNL, in Richland, Washington. This provides a portable "shared-memory" programming interface for distributed-memory computers. Within this implementation the PeIGS diagonalizer is used, and Lapi is used for communication internally. The alternative is a distributed memory implementation that uses ScaLAPACK to diagonalize the matrices, and MPI for communication.

7.2 CRYSTAL

CRYSTAL performs ab initio calculations of the ground state energy, electronic wave function and properties of periodic systems. The CRYSTAL software was jointly developed by the Theoretical Chemistry Group at the University of Torino and the Computational Materials Science Group at CCLRC Daresbury Laboratory (UK).

The CRYSTAL package performs the computation of the electronic structure using either Hartree-Fock or Density Functional theory. In each case the fundamental approximation made is the expansion of the single particle wave functions as a linear combination of atom centred atomic orbitals (LCAO) based on Gaussian functions. Powerful screening techniques are used to exploit real space locality. The code may be used to perform consistent studies of the physical, electronic and magnetic structure of molecules, polymers, surfaces and crystalline solids. CRYSTAL has been applied to studies of defects in ionic materials, the stability of minerals and oxide surface chemistry.