

Contents

1	Introduction	1
1.1	Formal definition and problems classification	2
1.2	Subspace methods	3
1.3	Survey of freely available Davidson software	5
1.4	The SLEPc library	6
1.5	High-performance computing	8
1.6	Thesis objectives	10
1.7	Outline	11
2	Implementation of Davidson Methods in SLEPc	13
2.1	Davidson-type framework	15
2.1.1	General description of the Davidson-type methods	15
2.1.2	Subspace extractions	20
2.1.3	Subspace expansions	22
2.1.4	Deflation	23
2.1.5	Restarting	24
2.1.6	Initializing the search subspace	24
2.2	SLEPc eigensolver design	25
2.2.1	PETSc description	25
2.2.2	Internal structure of SLEPc	26
2.2.3	Comparison with the design of other parallel Davidson software	27
2.2.4	Davidson-type eigensolvers interface	28
2.3	Implementation details	31
2.3.1	Solution of the correction equation	32
2.3.2	Real arithmetic	33
2.3.3	Parallelization and memory management details	34
2.3.4	Subspace orthogonalization	35
2.3.5	Convergence criterion for the eigensolver	36
2.4	Results	36
2.4.1	Test battery	36
2.4.2	Application 1: unstable modes of turbulent plasma	39

2.4.3	Application 2: electronic configuration of atoms	40
2.5	Conclusions	41
3	A Double-Expansion Davidson Method	43
3.1	Expansions in Davidson methods	44
3.2	A double subspace expansion approach	45
3.2.1	Comparative analysis	46
3.2.2	Convergence analysis	48
3.2.3	Algorithmic details	50
3.3	Numerical results	50
3.4	Conclusions	54
4	Parallel Jacobi-Davidson in GENE	55
4.1	The Jacobi-Davidson method	57
4.2	Implementation description	60
4.2.1	Overview of SLEPc	60
4.2.2	Parallelization details	61
4.2.3	Solution of the correction equation	61
4.3	GENE: A gyrokinetic plasma simulation code	62
4.4	Computational results	63
4.4.1	Stopping criterion for the correction equation solver	65
4.4.2	Jacobi-Davidson versus Krylov-Schur	66
4.4.3	Speedup and Scalability	67
4.5	Conclusions	68
5	Use of Preconditioners and Initial Guesses in GENE	71
5.1	The GENE code	72
5.2	Fast eigenvalue computations	74
5.2.1	Eigenvalue solver	75
5.2.2	Overview of SLEPc and PETSc	76
5.2.3	Approximate explicit matrix representation for the preconditioner	77
5.2.4	ASM+ILU preconditioner	77
5.2.5	pARMS preconditioner	78
5.2.6	Results for one parameter set	79
5.3	Parameter scans	81
5.3.1	Subspace recycling	81
5.3.2	Distances in parameter space	83
5.3.3	Parallelization	85
5.4	Application	86
5.5	Conclusions	86

6	Parallel DFT with Grid Refinement and Subspace Recycling	89
6.1	Introduction	89
6.2	Theoretical background	92
6.2.1	Physical–theoretical framework	92
6.2.2	Finite element approach	93
6.3	Management of computational grids	94
6.3.1	Finite element toolkit	95
6.3.2	Grid refinement	96
6.3.3	Summary	99
6.4	Large-scale eigenvalue problems	101
6.4.1	Subspace extraction	103
6.4.2	Subspace expansion	103
6.4.3	Exploiting symmetry in definite matrix pairs	104
6.4.4	Subspace recycling	105
6.4.5	The SLEPc library	105
6.5	Numerical experiments	106
6.5.1	Performance of the combined scheme	109
6.6	Discussion and conclusions	112
7	Conclusions and Future Work	115
7.1	Publications	117
7.2	Projects	118
7.3	Software based on SLEPc	118
Bibliography		121

List of Figures

2.1	Main classes in PETSc and SLEPc.	26
2.2	The shortest SLEPc example code that compute the eigenvalues of a non-Hermitian matrix.	31
2.3	Collaboration diagram for the SLEPc's Davidson solver when the Jacobi-Davidson expansion is selected.	34
2.4	Comparison of number of matrix-vector products (left) and execution time in seconds (right) between SLEPc GD and JD computing the largest magnitude eigenvalues of generalized non-Hermitian problems. A mark above the line corresponds to an experiment with GD performing better than JD.	37
2.5	Comparison of number of matrix-vector products (left) and execution time in seconds (right) between SLEPc GD and JD computing the eigenvalues closest to a target in non-Hermitian problems. A mark above the line corresponds to an experiment with GD performing better than JD.	38
2.6	Comparison of number of matrix-vector products (left) and execution time in seconds (right) between SLEPc JD (*) and PRIMME JD (⊙), taking the results of SLEPc GD as a basis, when computing the eigenvalues closest to a target in standard Hermitian problems. A mark above the line corresponds to an experiment with GD performing better than JD.	39
2.7	Speedup examples corresponding to GENE (left) and the DFT code (right) employing the SLEPc JD and GD solvers, respectively.	40
3.1	Residual norm against preconditioner applications spent by single expansion (GD) and the new expansion (GD2) using the high quality preconditioner M_0 (left plot) and the low quality preconditioner M_α with $\alpha = 10^{4/3}$ (right plot).	51
3.2	Gain of the new expansion (GD2) over the single expansion (GD) in number of preconditioner applications (left plot) and time (right plot), versus the quality of the ILU(0) decomposition as preconditioner.	53

4.1	Spectrum of the linearized operator of a GENE problem similar to the test case I (Table 4.1). The largest magnitude (circle marks) and the rightmost (cross marks) eigenvalues are desired.	56
4.2	Time (in seconds, left axis) and number of reductions (right axis) spent by Jacobi-Davidson solving the test case I with four processes. The correction equation is solved with BiCGstab(2) and a maximum number of iterations of 25, 50 and 80. The cases labeled as “var” use a variable tolerance for the stopping criterion. The total time is split into matrix-vector products (MV) and vector operations without (Ops 0) and with (Ops 1) communication.	65
4.3	Time (in seconds) using Krylov-Schur (KS) and Jacobi-Davidson (JD) with BiCGstab(2) solving test cases I, II and III with 4 (left) and 256 (right) processes. The total time is split into matrix-vector products (MV) and vector operations without (Ops 0) and with (Ops 1) communication.	66
4.4	Speedups solving the test cases I (top left), II (top right) and III (bottom left) with Jacobi-Davidson and Krylov-Schur. Speedup of the matrix-vector product in these problems (bottom right).	68
4.5	Time (in seconds) solving versions of the test cases I (left) and II (right) with increasing resolution in the v direction, with Jacobi-Davidson and Krylov-Schur.	69
5.1	Influence of the overlap δ (left) and the shift σ of the preconditioner matrix (right) on the total time. The plots show the mean and the standard deviation (left) and the minimum time (right) spent by JD with different domain decompositions.	80
5.2	Time spent with 16 processes (left) and speedup (right) of JD solving the test case I without preconditioner (None), with ASM preconditioner using $\delta = 2$ and the local preconditioner ILU (ASM+ILU), and with pARMS using the local preconditioner ARMS (pARMS). MV stands for matrix-vector products.	82
5.3	Computation time for the eigenvalue problem with $R/L_{Ti} = 4.0$ as a function of the difference to the R/L_{Ti} value of the eigenvectors used as initial condition, normalized to the computation time with random initial vectors.	83
5.4	Number of iterations for the eigenvalue problem with the nominal parameters as function of $D' = 1.0 - C(e_i, e_{n+1})$ (left) and D (right).	84
6.1	Local refinement of cells: (a) Coarse grid on level zero; (b) Level one refinement of one cell; and (c) Level one refinement of two cells and level two refinement of one cell.	96

6.2	Projected potential plotted as an interpolated function and the resultant error estimate for the starting grid (upper panel) and after two levels of refinement (middle panel) and four levels of refinement (lower panel). The error estimate is plotted for each finite element cell, where the “height” of the finite element cell depicts the estimated error. Note the change of scale in the error estimate.	100
6.3	Two sets of successfully refined grids based on the Kelly applied to the density functional as an error indicator and using different levels of refinement. In (a) for fractional refinement $N = 1/d^2$ of the total number of active cells; and (b) The more aggressive condition $N = 1/d$.	101
6.4	The same as in Fig. 6.3 but for two sets of successfully refined grids based on the projected potential as an error indicator. The results vary slightly.	102
6.5	Evolution of the relative energy difference and total number of degrees of freedom as a function of refinement cycle using the projected potential as an error estimate. The key (0) refers to a refinement level of $N = 1/d^2$ cells, whereas (1) refers to a refinement level of $N = 1/d$ cells.	109
6.6	Wallclock times with 16 processes computing the electronic configuration of H without and with feeding (left), and the various atoms with feeding (right). The time is split between the refinement processes and the solution of the Poisson and the Schrödinger problems.	112
6.7	Speedup computing the electronic configuration of H.	113

List of Tables

1.1	List of software with Davidson-type methods (GD: Generalized Davidson, JD: Jacobi-Davidson) for the solution of sparse eigenvalue problems, indicating the version and the year of the last release, the main language in which is written and whether distributed memory is supported.	5
2.1	Summary of parallel libraries providing Davidson-type solvers, with eigenproblem types supported by each method (HEP: standard Hermitian, GHPEP: generalized Hermitian, GNHEP: generalized non-Hermitian).	14
2.2	Correspondence between the values of α , β , γ and δ in the generic Galerkin condition (2.13) and some extraction methods.	22
2.3	Summary of differences among BLOPEX, Anasazi, PRIMME, and SLEPc's Davidson, considering the implementation of the linear algebra operations (Vectors and Matrices), the possibility to change the orthogonalization routine (Orth.), the convergence test (Conv.) and the sort function (Sort), and how the Davidson solvers are organized.	29
2.4	Number of converged cases in each experiment.	36
3.1	Iterations spent by the solvers versus the preconditioner quality . .	52
4.1	Test case I: GENE configuration for a very unstable kinetic ballooning mode with growth rate of 0.2055 and frequency of 0.2872 and another unstable mode ($0.1227 - 0.4494i$).	63
4.2	Test case II: GENE configuration similar to test case I but with a more realistic species configuration: deuterium, tritium, helium and electrons.	63
4.3	Test case III: GENE configuration corresponding to a stellarator device.	64

4.4	Total number of matrix-vector products (#MV), total time spent by them (T. MV) in seconds, average time spent by one product (T./#MV), and number of reductions (#red.) performed by both methods when solving the test cases with four processes.	67
5.1	Test case I: GENE configuration for an ITG mode with growth rate of 0.2055 and frequency of 0.2872 and a subdominant TEM ($0.1227 - 0.4494i$).	74
5.2	Time (in seconds) spent by JD with ASM+ILU solving the test case I with different distribution of processes across the directions s , z , v and μ	81
5.3	Wall clock times to compute the test parameter scan on 64 processors.	86
6.1	Evolution of the number of active cells (Act. cells), the total number of degrees of freedom (dofs), the total eigenenergy of the mean-field (eV) and the difference in the Hartree energy ($\Delta E_{\text{relative}} = E^{n-1} - E^n $), computing the electronic configuration of the Hydrogen atom. At each cycle of intermediate refinement defined by Eqn. (6.9), the number of cells flagged for refinement is $N = 1/d^2$ starting from a coarse grid globally refined $g = 2$ times.	107
6.2	Same as in table 6.1, where at each cycle the number of cells refined is $N = 1/d$. Three computations are given starting from a coarse grid globally refined $g \in \{2, 4, 6\}$	108
6.3	Evolution of the number of active cells (Act. cells), the total number of degrees of freedom (dofs), the iterations spent by the linear system solver (Poi.) and the eigensystem solver (Schr.), the total eigenenergy of the mean-field (eV) and the difference in the Hartree energy ($\Delta E_{\text{relative}} = E^{(n-1)} - E^{(n)} $), computing the electronic configuration of the Hydrogen atom performing two intermediate steps.	110
6.4	Same as in table 6.3, but performing four intermediate steps. . . .	111
6.5	Collective timings for the Schrödinger–Poisson set up and solution on 16 processors. Ref. means refinement.	111

List of Algorithms

2.1	Basic Davidson-type Method	16
2.2	Generalized Hermitian Davidson-type Method with B -orthogonalization	17
2.3	Generalized non-Hermitian Davidson-type Method	18
2.4	Optimized Iterative Classical Gram-Schmidt with B -inner product.	35
3.1	Simplified Davidson for finding the eigenvalue closest to τ	46
3.2	Inexact Inverse Iteration	48
4.1	Harmonic non-Hermitian Jacobi-Davidson	59