DIAGONALIZATION OF SPARSE MATRICES

Peter Karpov, Max Planck Computing and Data Facility, Garching

https://github.com/karpov-peter/sparse-eigensolvers-tutorial

Meet MPCDF
02.10.2025

DIAGONALIZATION METHODS

Eigenproblem $A\mathbf{v} = \lambda\mathbf{v}$

Direct methods	**Iterative methods**
→ all or substantial part (>10%) of eigenpairs	→ small part of eigenpairs (~several hundreds)
→ relatively small matrices (up to ~10^6)	→ much larger matrices (> 10^9)
→ dense matrices	→ (typically) sparse matrices
Software: LAPACK, ScaLAPACK, ELPA, SLATE, …	Software: ARPACK, SLEPc, ChASE, …

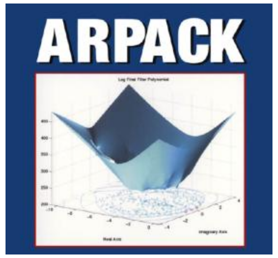
https://github.com/karpov-peter/elpa-tutorial
(Meet MPCDF, February 2024)

https://github.com/karpov-peter/sparse-eigensolvers-tutorial
(today)

OUTLINE

1. Theoretical background
- Power method
- Lanczos method (sketch)
- Matrix-free vs matrix-based methods
- Sparse matrix storage

2. ARPACK
- serial: Python/C/Fortran (“warmup”)
- MPI-parallel



3. SLEPc

4. Other tools and useful resources

POWER METHOD

Finds the eigenvalue of (complex) $n \times n$ matrix A with largest absolute value

If A is diagonalizable (i.e. there exists an invertible matrix P and diagonal matrix D: $P^{-1}AP = D$),

then eigenvectors $\mathbf{v}_i$ form a basis, so arbitrary $\mathbf{x}_0$ can be expressed as:

$$\mathbf{x}_0 = \sum_{i=1}^{n} c_i \mathbf{v}_i$$

$$\mathbf{x}_1 \equiv A\mathbf{x}_0 = A\left(\sum_{i=1}^{n} c_i \mathbf{v}_i\right) = \sum_{i=1}^{n} c_i \lambda_i \mathbf{v}_i$$

...

$$\mathbf{x}_k \equiv A\mathbf{x}_{k-1} = \ldots = A^k \mathbf{x}_0 = A^k \left(\sum_{i=1}^{n} c_i \mathbf{v}_i\right) = \sum_{i=1}^{n} c_i \lambda_i^k \mathbf{v}_i = \lambda_m^k \left(c_m \mathbf{v}_m + \sum_{i \neq m} c_i \left(\frac{\lambda_i}{\lambda_m}\right)^k \mathbf{v}_i\right)$$

POWER METHOD

Input: $n \times n$ matrix A, initial vector $\mathbf{x}_0$
Output: Approximate dominant eigenvector $\mathbf{v}$

```
$\mathbf{x} \leftarrow \mathbf{x}_0/\|\mathbf{x}_0\|$
do
    $\mathbf{y} \leftarrow A \cdot \mathbf{x}$
    $\mathbf{x} \leftarrow \mathbf{y}/\|\mathbf{y}\|$
until convergence
$\mathbf{v} \leftarrow \mathbf{x}$
```

$\lambda \approx \dfrac{\mathbf{v}^T A \mathbf{v}}{\mathbf{v}^T \mathbf{v}}$ – “Rayleigh quotient”

Improvement: at each step build orthogonal basis → “Krylov subspaces” → Arnoldi/Lanczos algorithms for non-Hermitian/Hermitian matrices

LANCZOS METHOD (SKETCH)

Let for simplicity $\lambda_1 > \lambda_2 > \ldots > \lambda_n > 0$
(we can shift the spectrum by a constant)

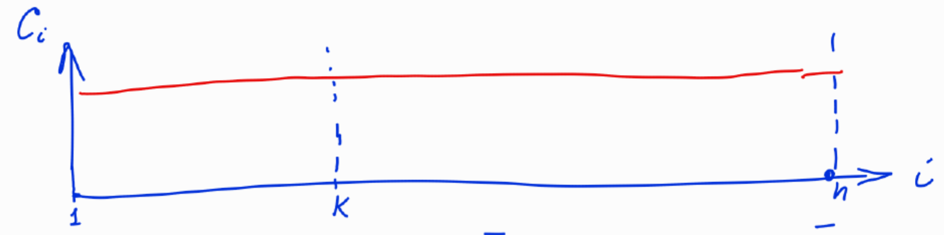
Instead of just building sequence $\mathbf{x}_0, A\mathbf{x}_0, \ldots A^k\mathbf{x}_0$
(not orthogonal), let's ortogonolize vector $\mathbf{x}_i$
to all previous $\mathbf{x}_0, \ldots, \mathbf{x}_{i-1}$ each iteration.

$$\boxed{\mathcal{K}^{k+1}(\mathbf{x}_0) = \mathrm{span}(\mathbf{x}_0, A\mathbf{x}_0, \ldots, A^k\mathbf{x}_0)}$$

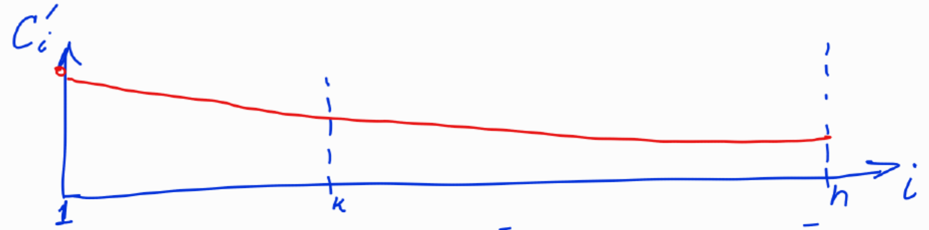
Krylov subspace of dimension $(k+1)$

Find orthonormal basis of $\mathcal{K}^{k+1}(\mathbf{x}_0)$: $\mathbf{y}_0, \mathbf{y}_1, \ldots, \mathbf{y}_k$

Diagonalize small $(k+1)\times(k+1)$ matrix
$\tilde{A}_{i,j} = (\mathbf{y}_i, A\mathbf{y}_j)$

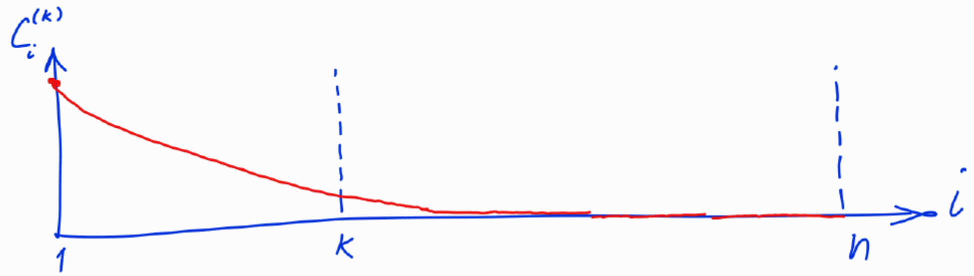


$$\bar{X}_0 = c_1\bar{v}_1 + \ldots \quad c_k\bar{v}_k + c_{k+1}\bar{v}_{k+1} + \ldots \quad + c_n\bar{v}_n$$



$$\bar{X}_1 = \underbrace{c_1\lambda_1}_{c_1'}\bar{v}_1 + \ldots + \underbrace{c_k\lambda_k}_{c_k'}\bar{v}_k + \underbrace{c_{k+1}\lambda_{k+1}}_{c_{k+1}'}\bar{v}_{k+1} + \ldots \quad + \underbrace{c_n\lambda_n}_{c_n'}\bar{v}_n$$

• • • • • •



$$\bar{X}_k = \underbrace{c_1\lambda_1^k}_{c_1^{(k)}}\bar{v}_1 + \ldots + \underbrace{c_k\lambda_k^k}_{c_k^{(k)}}\bar{v}_k + c_{k+1}\lambda_{k+1}^k\bar{v}_{k+1} + \ldots \quad + \underbrace{c_n\lambda_n^k}_{c_n^{(k)}}\bar{v}_n$$

MATRIX-FREE VS MATRIX-BASED ALGORITHMS

Matrix-free algorithm: matrix elements are not stored in memory, calculated on a fly
- Favorable for extremely large-scale problems
- Typically, we only need to know how matrix acts on a vector (matrix-vector product)

E.g. in Power method:

```
$\mathbf{x} \leftarrow \mathbf{x}_0/\|\mathbf{x}_0\|$
do
    $\mathbf{y} \leftarrow A \cdot \mathbf{x}$ // matrix-vector product – the only info about A
    $\mathbf{x} \leftarrow \mathbf{y}/\|\mathbf{y}\|$
until convergence
$\mathbf{v} \leftarrow \mathbf{x}$
```

Matrix-based algorithm: we store in memory only non-zero elements of the matrix
- Conceptually simpler (similar paradigm to dense e.g. LAPACK: matrix in memory, library function call to calculate its spectrum)

MATRIX-BASED METHODS: SPARSE MATRIX STORAGE FORMATS

Two classes of sparse matrix formats:

1. Allow efficient insertion of elements.
Example: “COO” format (**Coo**rdinate list): (row, column, value)
(i1, j1, val1), (i2, j2, val2), …

2. Allow efficient matrix-vector multiplication
Example: “CSR” format (**C**ompressed **S**parse **R**ow)

$$\begin{pmatrix} 10 & 20 & 0 & 0 & 0 & 0 \\ 0 & 30 & 0 & 40 & 0 & 0 \\ 0 & 0 & 50 & 60 & 70 & 0 \\ 0 & 0 & 0 & 0 & 0 & 80 \end{pmatrix}$$

```
VALUE     = [ 10  20|30  40|50  60  70|80  ]
ROW_INDEX = [   0    2      4           7   8]
COLUMN    = [   0   1   1   3   2   3   4   5  ]
```

`ROW_INDEX[i]` = how many nonzero elements are above row *i*

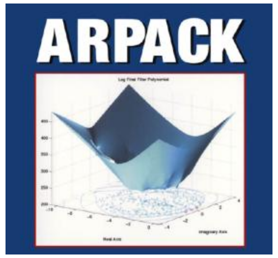
OUTLINE

1. Theoretical background
- Power method
- Lanczos method (sketch)
- Matrix-free vs matrix-based methods
- Sparse matrix storage

2. ARPACK
- serial: Python/C/Fortran (“warmup”)
- MPI-parallel

3. SLEPc

4. Other tools and useful resources



ARPACK

ARnoldi **PACK**age (uses Implicitly Restarted Arnoldi/Lanczos method)

Fortran-77 library

ARPACK-NG – community-maintained version (Debian, Octave and Scilab)
https://github.com/opencollab/arpack-ng (includes also P_ARPACK)

+ fast

+ MPI-parallelized (P_ARPACK)

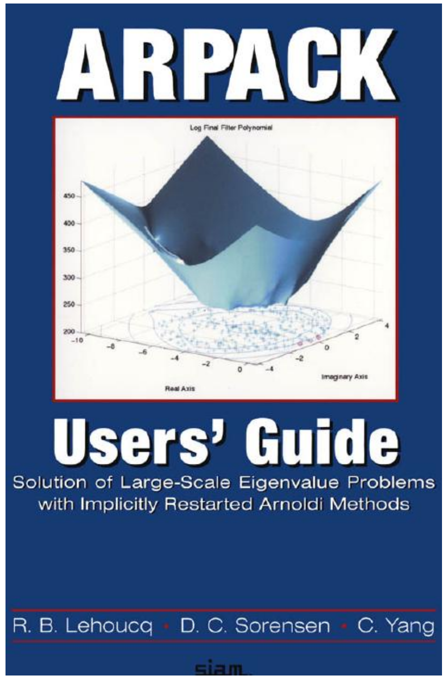
+ extensively tested, stable

+ matrix-free

– cumbersome Fortran-style interface

– not actively developed anymore (e.g. no GPU support)

Many packages use it under the hood: SciPy (python), Mathematica, Armadillo (C++), MATLAB, …


ARPACK
Users' Guide
Solution of Large-Scale Eigenvalue Problems with Implicitly Restarted Arnoldi Methods
R. B. Lehoucq · D. C. Sorensen · C. Yang
siam

ARPACK: PYTHON

SciPy: `from scipy.linalg.sparse import eigs, eigsh`
https://docs.scipy.org/doc/scipy/tutorial/arpack.html

Find eigenvalues/vectors of real/complex **sparse** matrix
1. **eigs** – for general (non-hermitian/non-symmetric)
2. **eigsh** – for **hermitian** (complex) or symmetric (real) matrices

(analogous to dense-matrix functions: `from scipy.linalg import eig, eigh`)

ARPACK: FORTRAN / C

Routine naming convention: **pma**`upd`

p – precision:	`s,d`	–	real **s**ingle and **d**ouble precision
	`c,z`	–	complex single and double precision
m – matrix type:	`s`	–	**s**ymmetric
	`n`	–	**n**on-symmetric
a – algorithm:	`a`	–	**A**rnoldi (Lanczos) iteration single update
	`e`	–	**e**igenvectors postprocessing

Example: `dsaupd` – double symmetric Arnoldi (i.e. Lanczos) update

`dsaupd_c` – same, but C-version

ARPACK: FORTRAN / C, REVERSE COMMUNICATION INTERFACE

Matrix-free MatVec multiplication: input x, output y = Ax

```
void myMatVec(double* x, double* y){ // Example: “1D Laplace matrix“
  int i;
  for (i = 0; i < n;   i++) y[i]   +=  2.0*x[i]; // Main diagonal
  for (i = 0; i < n-1; i++) y[i+1] += -1.0*x[i]; // Subdiagonal
  for (i = 1; i < n;   i++) y[i-1] += -1.0*x[i]; // Superdiagonal
}
```

$$\begin{bmatrix} +2 & -1 & 0 & 0 \\ -1 & +2 & -1 & 0 \\ 0 & -1 & +2 & -1 \\ 0 & 0 & -1 & +2 \end{bmatrix}$$

Reverse communication interface:

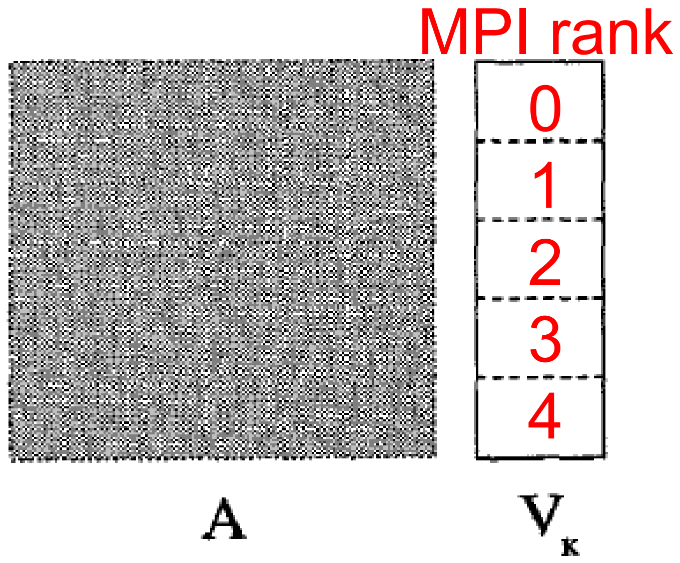
```
int ido = 0; // reverse communication flag; should be initialized with 0
do {
    // "double symmetric Arnoldi update"
    dsaupd_c(&ido, bmat, N, which, nev, tol, resid, ncv, V, ldv, iparam, ipntr,
             workd, workl, lworkl, &info);

    // calculate y=Ax, where for x, y are parts of workd array
    myMatVec(&(workd[ipntr[0] - 1]), &(workd[ipntr[1] - 1]));
} while (ido == 1 || ido == -1);
```

P_ARPACK

Parallel **ARPACK** – MPI parallel version of ARPACK

- The parallel storage the matrix (or calculation on the fly) is up to user
- Eigenvalues are global (same on all MPI processes)
- Eigenvectors are distributed by rows



Routine naming:	ARPACK	→	P_ARPACK
For example:	`dsaupd`	→	**`p`**`dsaupd`

K. J. Maschhoff and D. C. Sorensen, P_ARPACK: An Efficient Portable Large Scale Eigenvalue Package for Distributed Memory Parallel Architectures (1996)

OUTLINE

1. Theoretical background
- Power method
- Lanczos method (sketch)
- Matrix-free vs matrix-based methods
- Sparse matrix storage

2. ARPACK
- serial: Python/C/Fortran (“warmup”)
- MPI-parallel

3. SLEPc

4. Other tools and useful resources

SLEPc

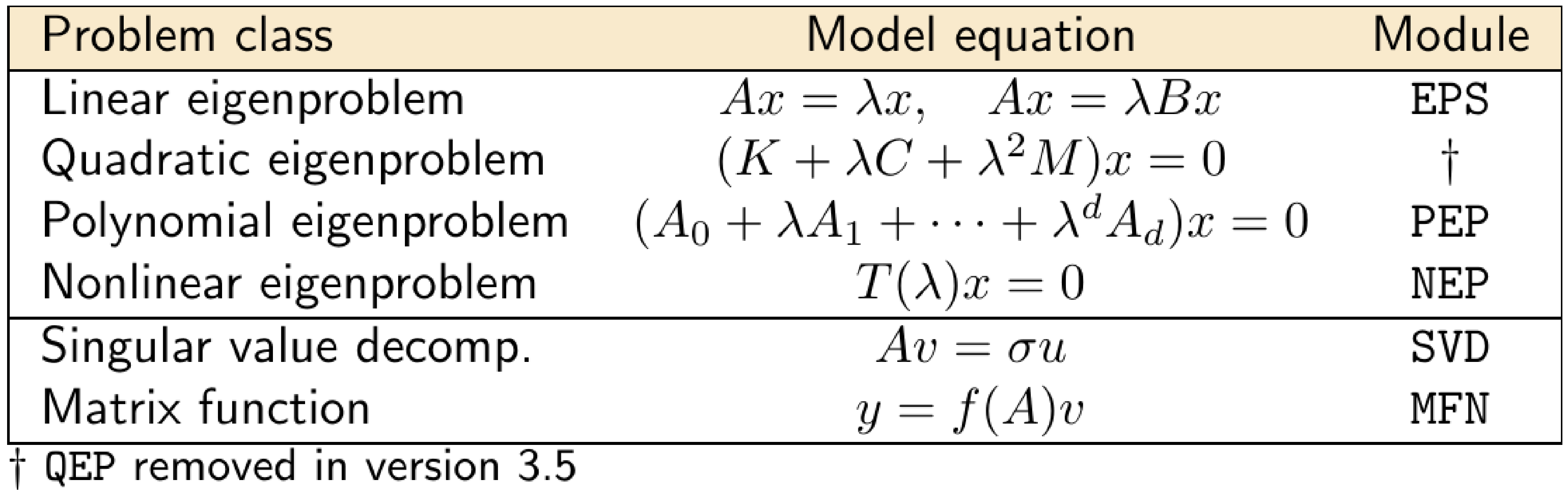
Scalable Library for Eigenvalue Problem computations

Built on top of PETSc (Portable, Extensible Toolkit for Scientific computation)

+ fast
+ MPI-parallelized
+ GPU-support (not for all routines)
+ both matrix-free and matrix-based
+ many solvers under the hood

– harder to learn and incorporate in your project (but a lot of great materials for study!)

SLEPc: TYPES OF EIGENPROBLEMS

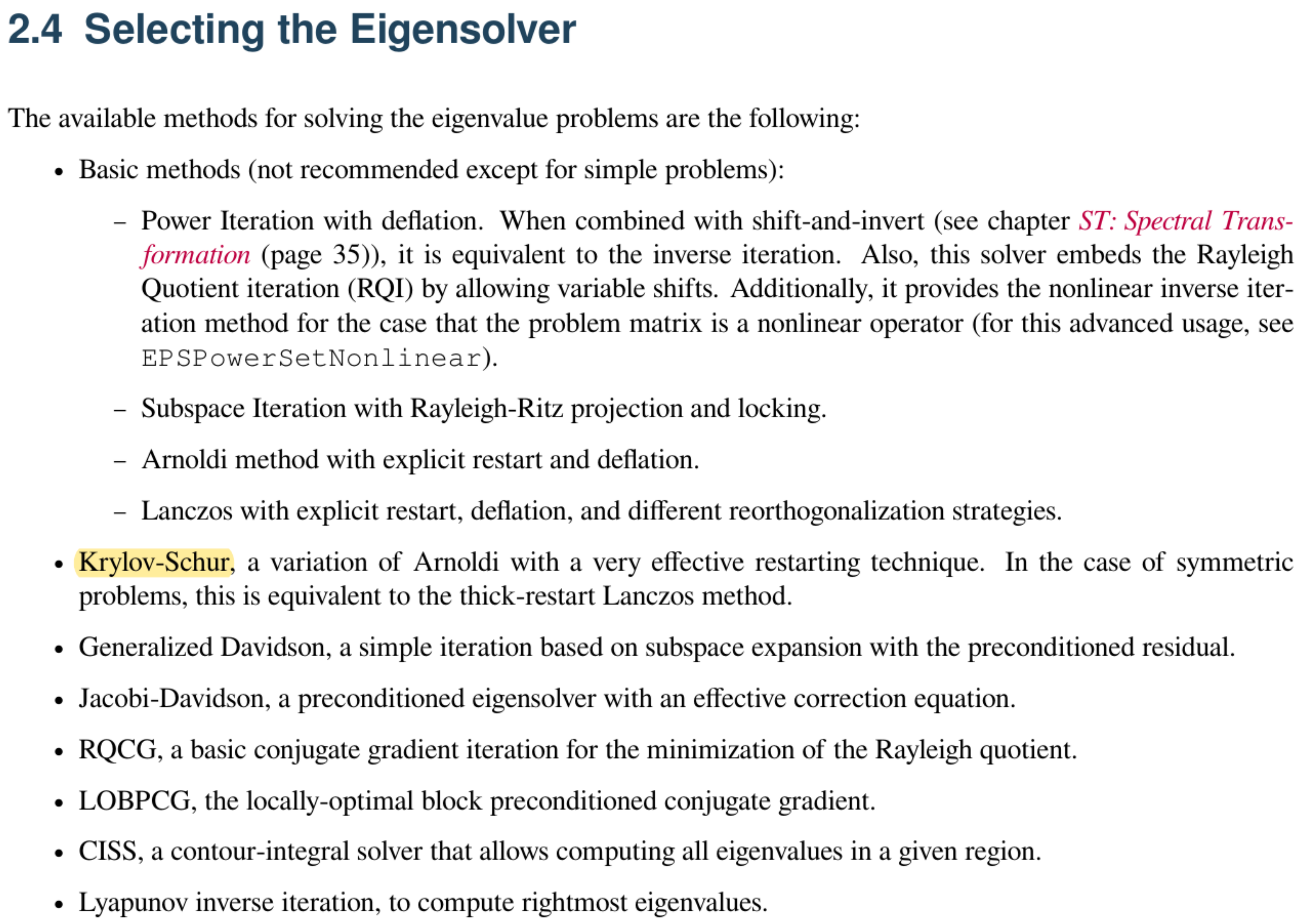


Problem class	Model equation	Module
Linear eigenproblem	$Ax = \lambda x, \quad Ax = \lambda Bx$	EPS
Quadratic eigenproblem	$(K + \lambda C + \lambda^2 M)x = 0$	†
Polynomial eigenproblem	$(A_0 + \lambda A_1 + \cdots + \lambda^d A_d)x = 0$	PEP
Nonlinear eigenproblem	$T(\lambda)x = 0$	NEP
Singular value decomp.	$Av = \sigma u$	SVD
Matrix function	$y = f(A)v$	MFN

† QEP removed in version 3.5

Source: SLEPc tutorial, Jose E. Roman

SLEPc: SOLVERS FOR LINEAR EIGENPROBLEM (“EPS MODULE”)



2.4 Selecting the Eigensolver

The available methods for solving the eigenvalue problems are the following:

- Basic methods (not recommended except for simple problems):
  - Power Iteration with deflation. When combined with shift-and-invert (see chapter *ST: Spectral Transformation* (page 35)), it is equivalent to the inverse iteration. Also, this solver embeds the Rayleigh Quotient iteration (RQI) by allowing variable shifts. Additionally, it provides the nonlinear inverse iteration method for the case that the problem matrix is a nonlinear operator (for this advanced usage, see `EPSPowerSetNonlinear`).
  - Subspace Iteration with Rayleigh-Ritz projection and locking.
  - Arnoldi method with explicit restart and deflation.
  - Lanczos with explicit restart, deflation, and different reorthogonalization strategies.
- Krylov-Schur, a variation of Arnoldi with a very effective restarting technique. In the case of symmetric problems, this is equivalent to the thick-restart Lanczos method.
- Generalized Davidson, a simple iteration based on subspace expansion with the preconditioned residual.
- Jacobi-Davidson, a preconditioned eigensolver with an effective correction equation.
- RQCG, a basic conjugate gradient iteration for the minimization of the Rayleigh quotient.
- LOBPCG, the locally-optimal block preconditioned conjugate gradient.
- CISS, a contour-integral solver that allows computing all eigenvalues in a given region.
- Lyapunov inverse iteration, to compute rightmost eigenvalues.

Source:
Jose E. Roman et al,
SLEPc Users Manual
v.3.24 (2025)

OTHER SPARSE EIGENSOLVERS (NON COMPREHENSIVE)

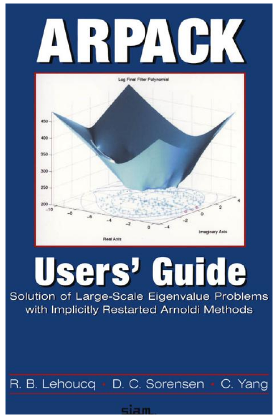
Anasazi – a C++ package within the Trilinos framework. Supports Hermitian and non-Hermitian problems, and includes block Krylov–Schur, block Davidson, LOBPCG. MPI and GPU support (through Kokkos).

ChASE (**C**hebyshev **A**ccelerated **S**ubspace **E**igensolver) – Chebyshev filtering and subspace iteration. Favourable when larger part of eigenspectrum is needed. MPI and GPU support.

FEAST – Contour-integration method**.** Integrates the resolvent along a contour in the complex plane to compute eigenpairs within that contour. MPI+OpenMP support.

Review of pre-2009 solvers:
V. Hernandez et al, A Survey of Software for Sparse Eigenvalue Problems, SLEPc Technical Report-6 (2009)

RESOURCES: ARPACK


ARPACK

Users' Guide

Solution of Large-Scale Eigenvalue Problems with Implicitly Restarted Arnoldi Methods

R. B. Lehoucq D. C. Sorensen C. Yang

siam

ARPACK-NG repo (many examples, both ARPACK and P_ARPACK):
https://github.com/opencollab/arpack-ng

ARPACK User's guide
http://li.mit.edu/Archive/Activities/Archive/CourseWork/Ju_Li/MITCourses/18.335/Doc/ARPACK/Lehoucq97.pdf

P_ARPACK paper
http://softlib.rice.edu/pub/CRPC-TRs/reports/CRPC-TR96659.pdf

RESOURCES: SLEPc

Official website (Manual, hands-on, …)
https://slepc.upv.es/

Set of 10-min videotutorials
https://slepc.upv.es/release/documentation/index.html#video-tutorials

Need help ? https://helpdesk.mpcdf.mpg.de/
petr.karpov@mpcdf.mpg.de