SLEPc

Linear Algebra Libraries for Exascale Materials Science Applications

Jose E. Roman

D. Sistemes Informàtics i Computació
Universitat Politècnica de València, Spain

2nd TIMES School, Bologna, Nov 2026

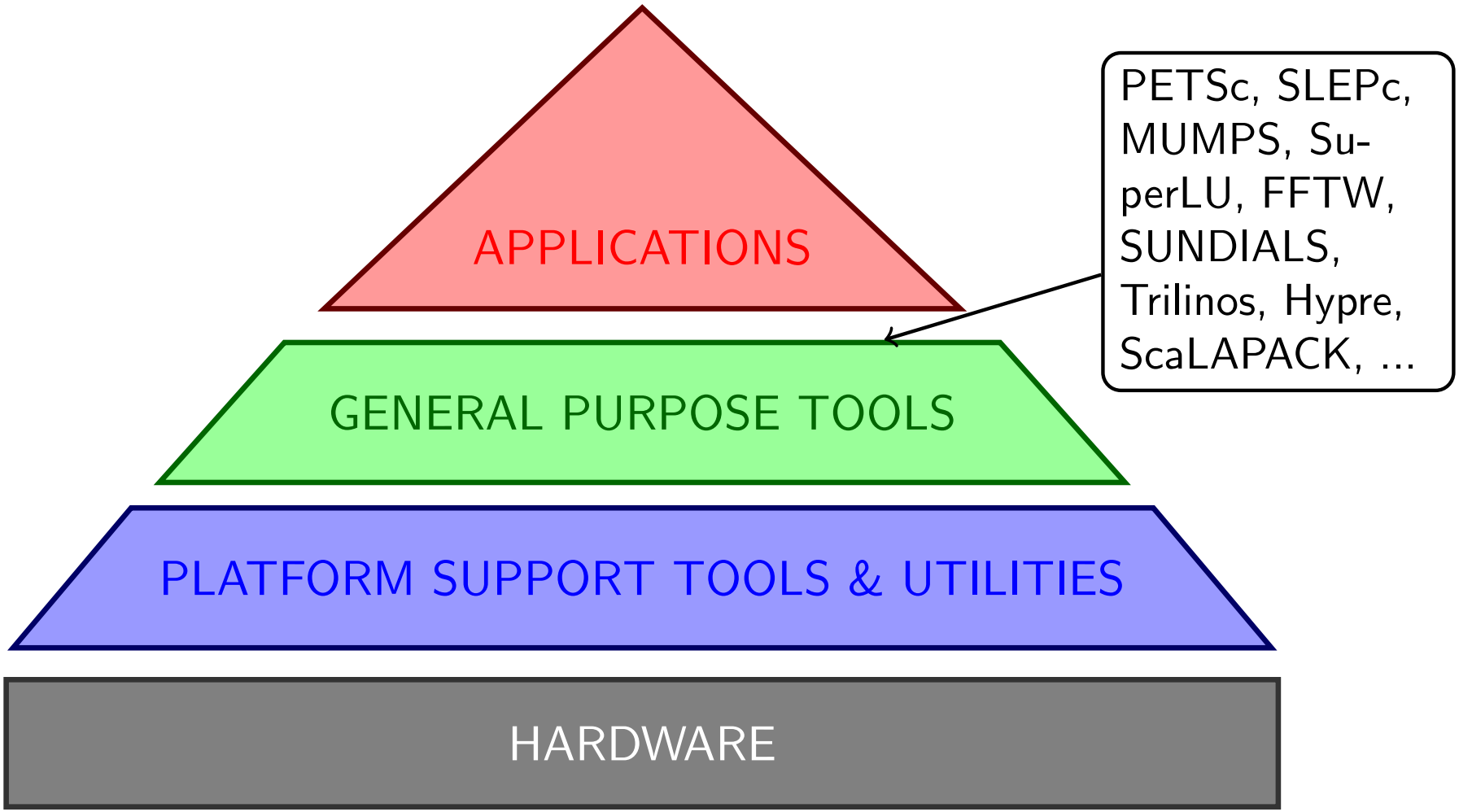
DSIC

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Outline

1. Introduction
   - Panorama of linear algebra software
   - PETSc and SLEPc
2. Eigenvalue Problems
   - Types of eigenvalue problems
   - Functionality of SLEPc
   - Exterior Eigenvalues
   - Interior Eigenvalues
3. Other Problems
   - Singular Value Decomposition
   - Matrix Functions

HPC Software Stack



Requirements of Sustainable Software

1. Robust (precision independent, error handling)
2. Scalable (across large supercomputers)
3. Extensible (new algorithms, new techniques)
4. Interoperable (programming language, installation, API)
5. User friendly interfaces, well documented
6. Periodic tests and evaluations (versions of tools, OS, compiler)
7. Portability across platforms, compiler independent
8. Long-term support
9. Training (hands-on examples)
10. Community support (developers, user groups, mailing lists)

Why so many Linear Algebra Libraries?

Different scope

- Basic codes: single problem, single method
- Libraries (e.g., LAPACK): several problems, several methods
- Toolkits (e.g., PETSc): a full environment with many tools

Different types of parallelism

- Distributed memory, shared memory, GPU
- Support for hybrid parallelism, e.g., MPI-CUDA

Different programming language

- Traditional languages: C/C++, Fortran
- Newer languages: CUDA, HIP, Python, Julia, Rust
- Toolkits usually provide bindings for several languages

Different target problem

- Dense vs sparse storage of matrices
- Direct vs iterative methods

Dense Linear Algebra (1)

Sequential dense storage

- Bidimensional array, very simple
- Pass pointer, sizes, leading dimension as arguments

BLAS: Basic Linear Algebra Subprograms

- Many implementations: OpenBLAS, BLIS, MKL, ACML, ...
- Some of them are multi-threaded

For GPU

- Bidimensional array on device memory
- Vendor implementation with slightly different API: cuBLAS, hipBLAS

Dense Linear Algebra (2)

LAPACK: Linear Algebra Package

Problem	Driver routines
Linear systems, $Ax = b$	_sv, _svx
Linear least squares, $\min \|b - Ax\|_2$	_ls, _lsy, _lsd
Generalized linear least squares	_lse, _glm
Eigenvalue problem, $Ax = \lambda x$	_ev, _evd, _evr, _es
Generalized eigenvalue, $Ax = \lambda Bx$	_gv, _gvx, _gvd, _es
Singular value decompositon, $A = U\Sigma V^*$	_svd, _sdd
Generalized SVD, $A = U\Sigma_1 Q^*, B = V\Sigma_2 Q^*$	_svd

Alternatives: GSL, Eigen

For GPU

- MAGMA: Matrix Algebra on GPU and Multicore Architectures
  - Dense linear algebra, includes batched calculations
- cuSOLVER (part of CUDA), rocSOLVER
  - Dense linear algebra, but also sparse

Parallel Dense Linear Algebra - Data Parallelism

Data parallelism (MPI)

- MPI processes execute parallel loops synchronously
- Parallel matrix storage, typically 2-D block cyclic distribution

0	1	0	1	0	1	0	1
2	3	2	3	2	3	2	3
0	1	0	1	0	1	0	1
2	3	2	3	2	3	2	3
0	1	0	1	0	1	0	1
2	3	2	3	2	3	2	3
0	1	0	1	0	1	0	1
2	3	2	3	2	3	2	3

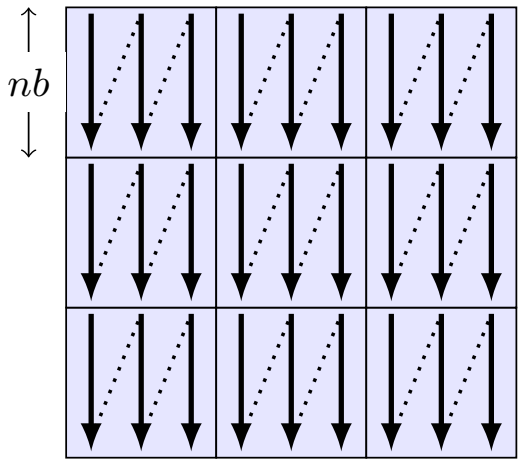
Libraries

- ScaLAPACK: a distributed-memory analog of LAPACK
- ELPA: direct Hermitian eigensolver, includes GPU support
- EigenExa: high-performance eigenvalue solver, MPI+OpenMP
- ChASE: iterative (pseudo-)Hermitian eigensolver, CPU, GPU
- KSVD: singular value decompositions, CPU-only

Parallel Dense Linear Algebra - Task Parallelism

Task parallelism (single/multi-node)

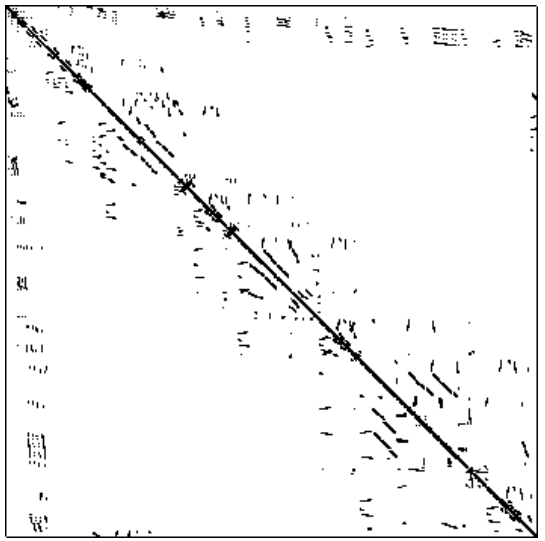
- Task dependency graph implemented explicitly
- Dynamic task scheduling, run on CPU or GPU, dependencies are enforced by the runtime
- Tiled matrix storage



Libraries

- SLATE: full coverage of LAPACK and ScaLAPACK functionality
- DLA-Future: Hermitian eigenproblems, Cholesky
- Chameleon: matrix multiplication, linear systems
- libFLAME: alternative to LAPACK (multi-thread only)

Sparse Linear Algebra



Mesh-based applications

- Small percentage of nonzeros
- Special data structures/algorithms
- In parallel usually block-row distrib.
- Generally iterative methods

Libraries (by problem type)

- Linear systems (direct): MUMPS, SuperLU, PaStiX, SuiteSparse
  (iterative, preconditioners): hypre, HPDDM, HIPS, pARMS
- Least squares: qr_mumps, SuiteSparseQR
- Nonlinear systems and ODEs: SUNDIALS
- Eigenvalues: ARPACK, PRIMME, FEAST

On GPU: cuSPARSE, cuSOLVER, AmgX, Gingko

Toolkits provide everything: PETSc/SLEPc, Trilinos

PETSc

Portable, Extensible Toolkit for Scientific Computation

- Can be used from C/C++, Fortran, Python (`petsc4py`)
- MPI parallel, can use multi-threaded BLAS
  - Support for GPUs with CUDA, HIP and Kokkos
- Linear algebra core: vectors, matrices and index sets
- `Vec` and `Mat` may be associated to a `DM`
  - Structured, unstructured, staggered, network, ...
- Automatic profiling

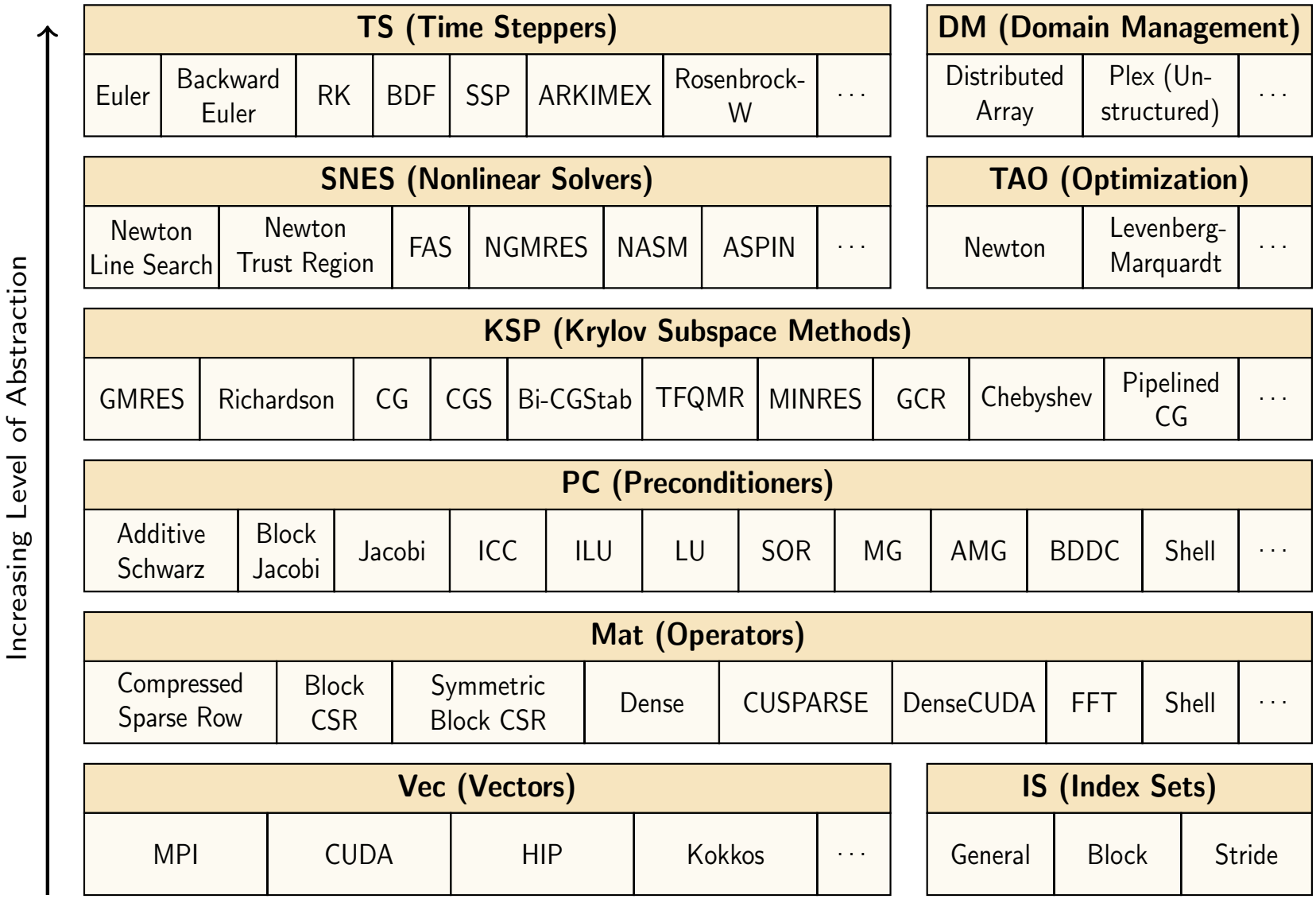
PETSc seamlessly integrates third-party packages

- MUMPS, SuperLU, SuiteSparse, hypre, pARMS, Parmetis, Elemental, ...

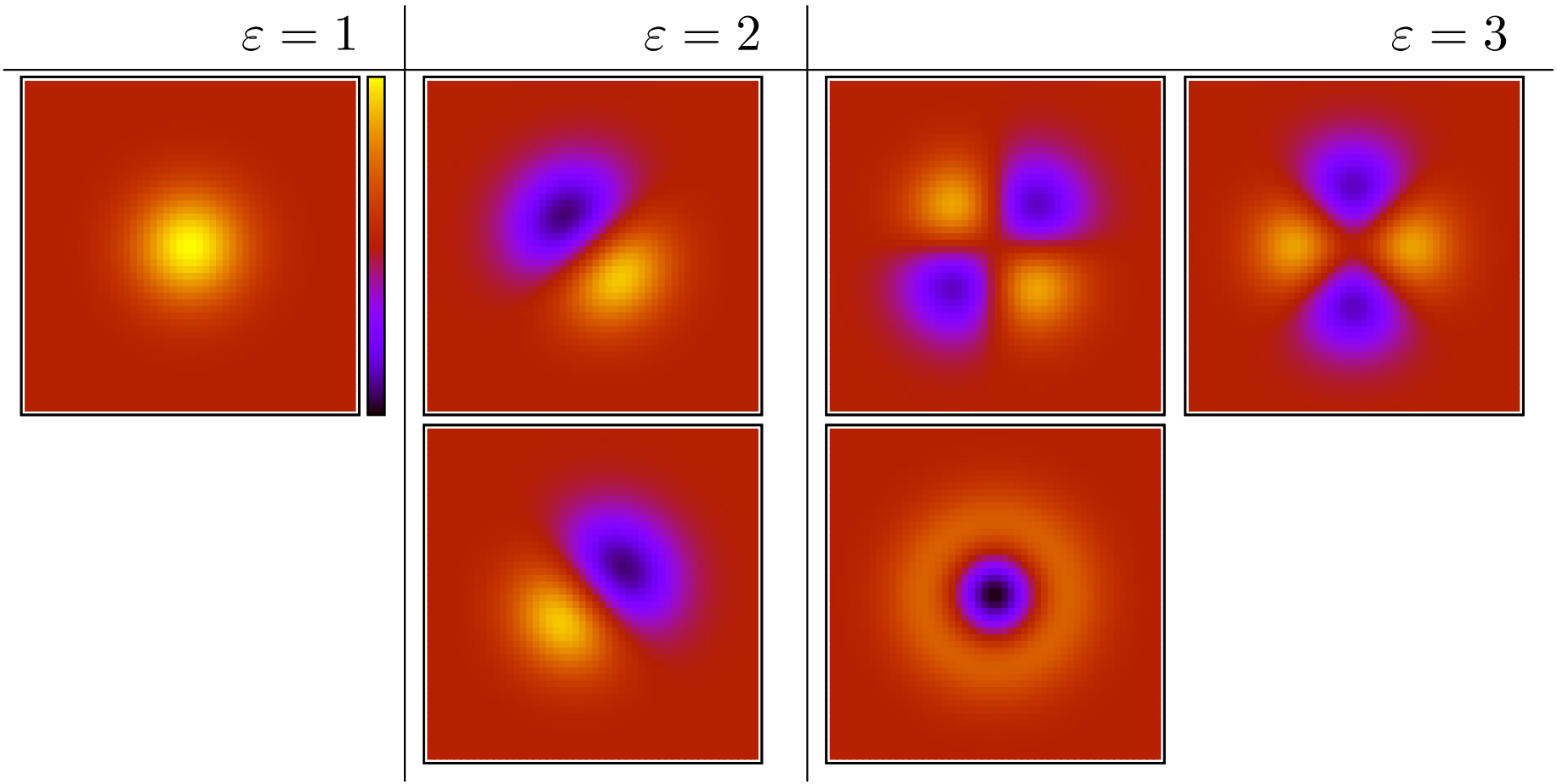
PETSc is the core of many higher-level packages

- MOOSE, libMesh, deal.II, FEniCS, Firedrake, MFEM, OpenFVM, ...

PETSc components



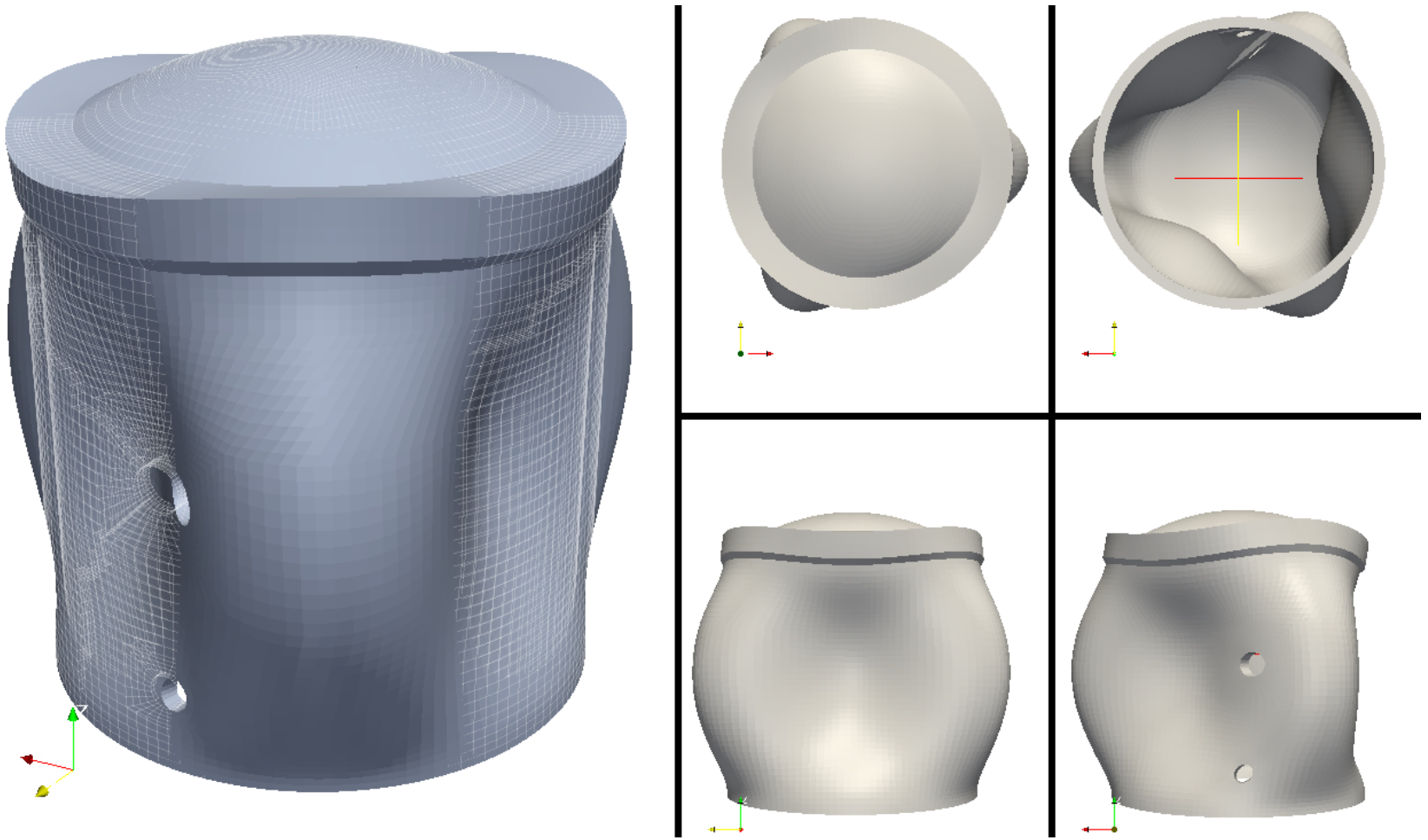
Example 1: Density Functional Theory



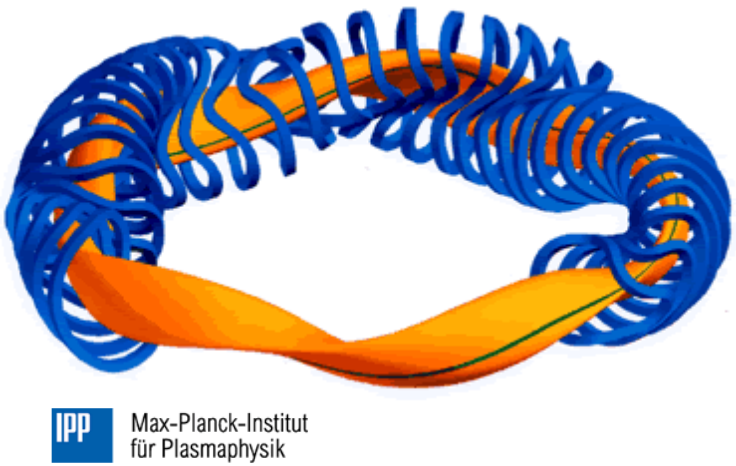
Eigenstates of quantum harmonic oscillator, $H\psi = \varepsilon\psi$

Example 2: Structural Analysis

Smallest eigenfrequencies of a real symmetric problem $Kx = \lambda Mx$

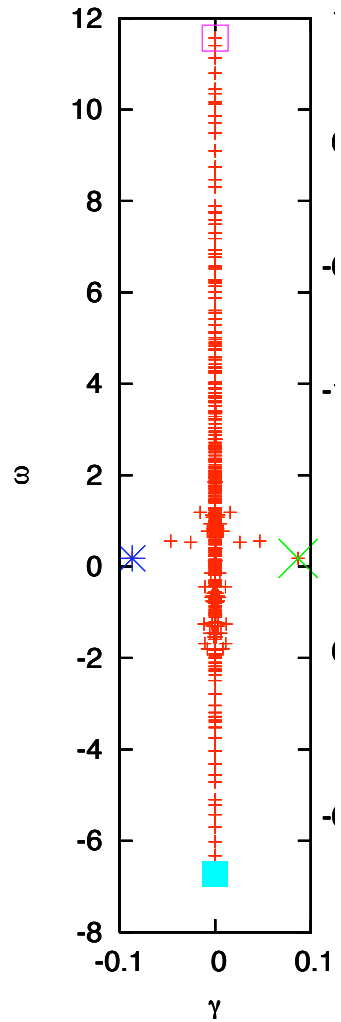


Example 3: Plasma Physics



Plasma turbulence in a tokamak determines its energy confinement

- GENE code
- Initial value solver



Knowledge of the spectrum of the linearized equation

$$Ax = \lambda x$$

- Complex, non-Hermitian, implicit A
- Estimate optimal timestep (largest eigenvalue); track sub-dominant instabilities (rightmost evals)

Example 4: Neutron Difusion Equation in Nuclear Eng.

Neutron power in nuclear reactor cores

- Commercial reactors such as PWR
- Both steady state and transient

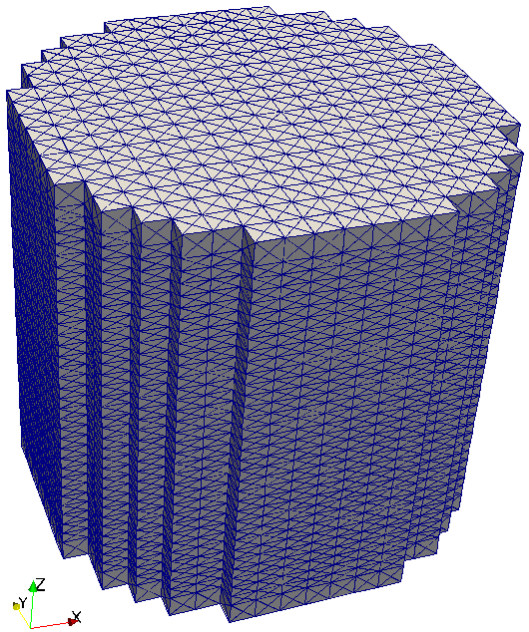
Lambda Modes Equation

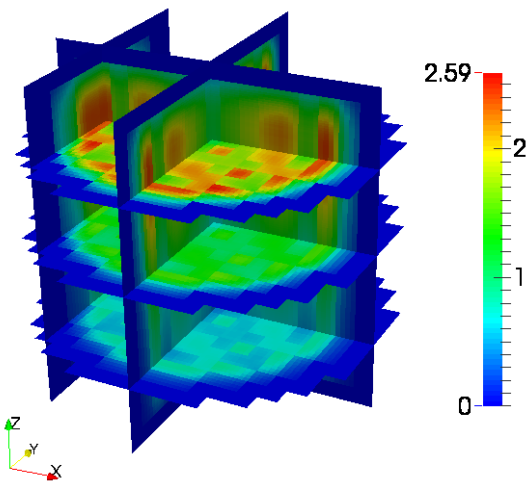
$$\mathcal{L}\phi = \frac{1}{\lambda}\mathcal{M}\phi$$

$$\begin{bmatrix} L_{11} & 0 \\ -L_{21} & L_{22} \end{bmatrix} \begin{bmatrix} \psi_1 \\ \psi_2 \end{bmatrix} = \frac{1}{\lambda} \begin{bmatrix} M_{11} & M_{12} \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \psi_1 \\ \psi_2 \end{bmatrix}$$

Solved as $G\psi_1 = \lambda\psi_1$ with

$$G = L_{11}^{-1}(M_{11} + M_{12}L_{22}^{-1}L_{21})$$

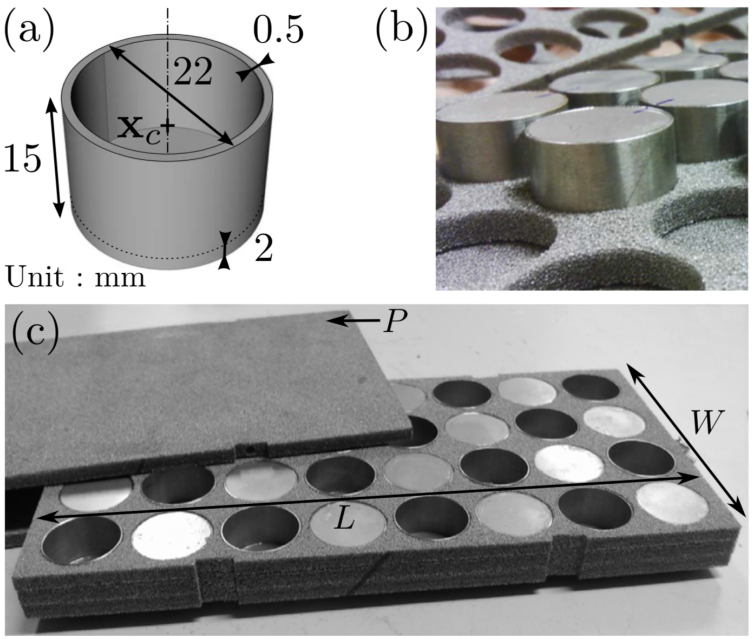




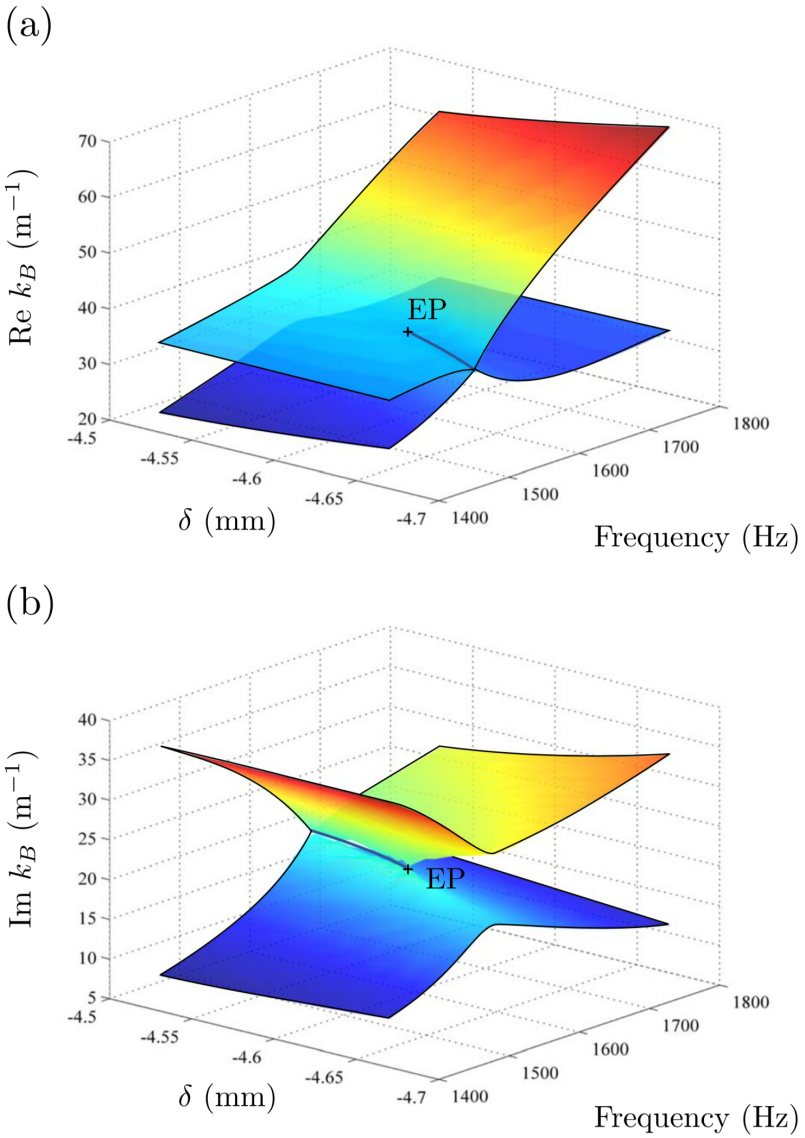
Example 5: Acoustic Metamaterials

Periodic domain; objective: minimize attenuation

$$(A_0 + A_1\lambda + A_2\lambda^2)x = 0$$



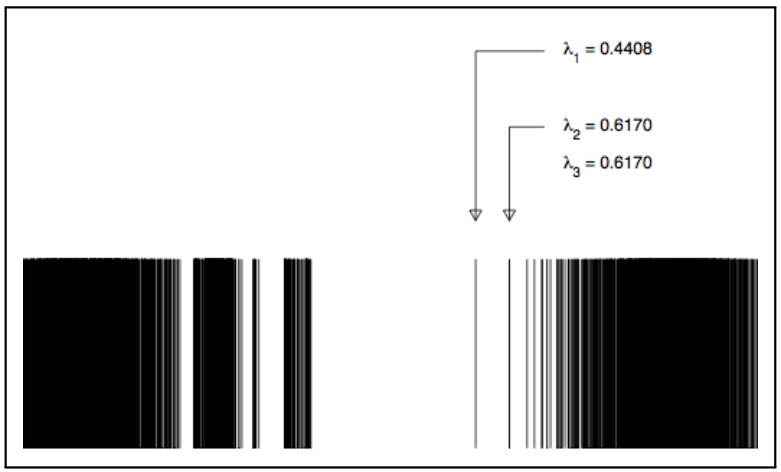
L. Xiong et al. *J. Acoust. Soc. Am.* 142 (2017)

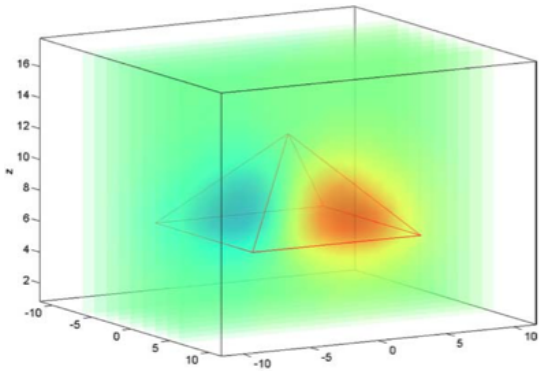


Example 6: Quantum Dot Simulation

3D pyramidal quantum dot discretized with finite volumes

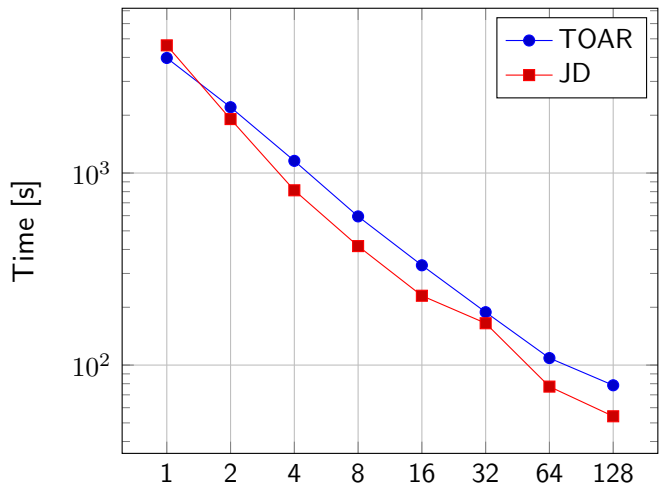
T.-M. Hwang et al. (2004), "Numerical Simulation of Three Dimensional Pyramid Quantum Dot," J. Comput. Phys. 196(1):208–232.





Quintic polynomial, $n \approx 12$ mill.

Scaling for tol=10^{-8}, nev=4, ncv=30 with inexact shift-and-invert (`bcgs+bjacobi`)

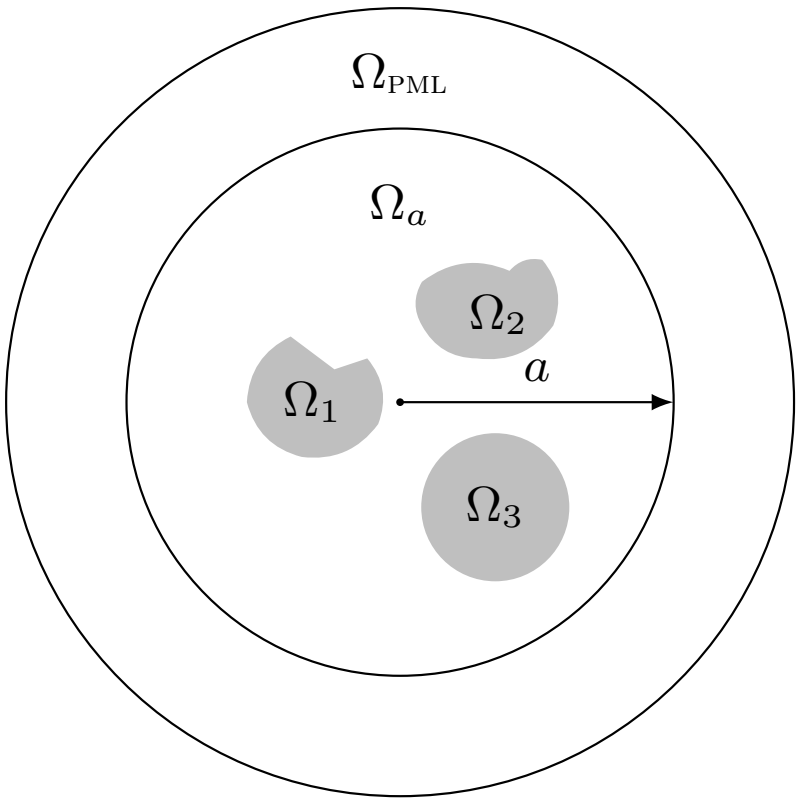


Example 7: Exterior Helmholtz Resonance Problem

J. C. Araujo, C. Campos, C. Engström, J. E. Roman (2020), "Computation of scattering resonances in absorptive and dispersive media with applications to metal-dielectric nano-structures," J. Comput. Phys. 407:109220

Goal: scattering resonances of metallic nano-structures

- Main interest: surface plasmons
- Applications: nano-antennas, sensors, cancer treatment with gold nanoparticles



Model problem: transverse scattering

- Planar wave perpendicular to infinite scatterers
- Piecewise constant permittivities

Example 7: Exterior Helmholtz (Equations)

Two polarizations are of interest: TM and TE

Case TM:

$$\Delta u + \varepsilon(x,\omega)\omega^2 u = 0, \quad x \in \mathbb{R}^2$$

Case TE:

$$\nabla \cdot \left(\frac{1}{\varepsilon(x,\omega)} \nabla u \right) + \omega^2 u = 0, \quad x \in \mathbb{R}^2$$

Drude-Lorentz model:

$$\varepsilon(\omega) := \varepsilon_\infty + \sum_{j=0}^{N_p} \frac{f_j \omega_p^2}{\omega_j^2 - \omega^2 - i\omega\gamma_j}$$

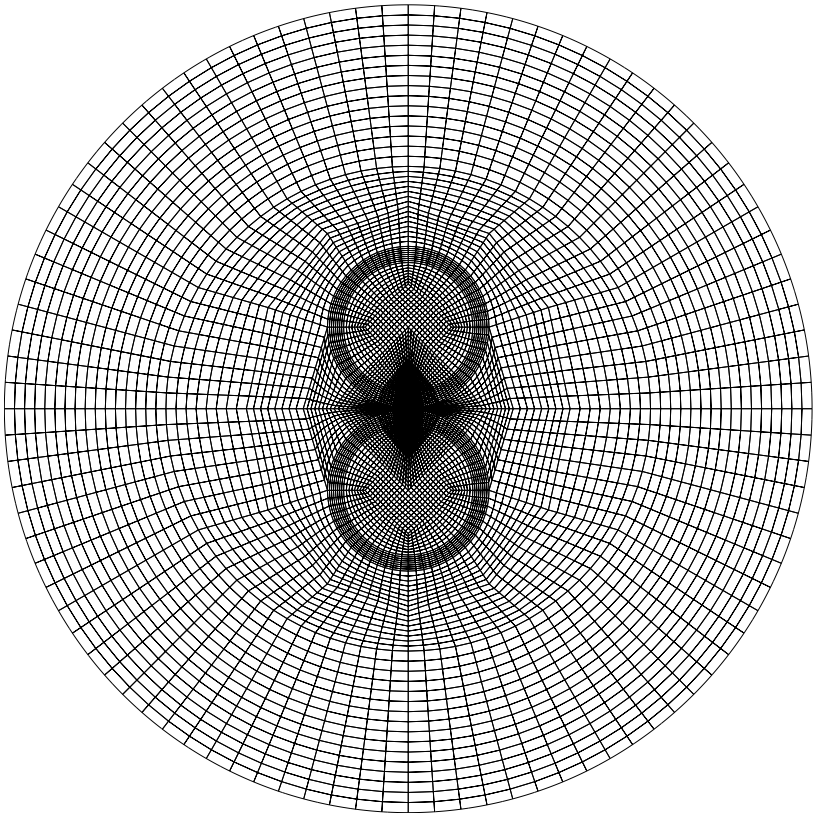
In the case of gold: six rational terms ($N_p = 5$)

Example 7: Exterior Helmholtz (Dimer Problem)

Two coated disks separated vertically by a distance $s = 0.02$

FEM discretization with deal.II

- complex matrices
- problem size $n = 922,081$
- nonzeros: 62 mill.



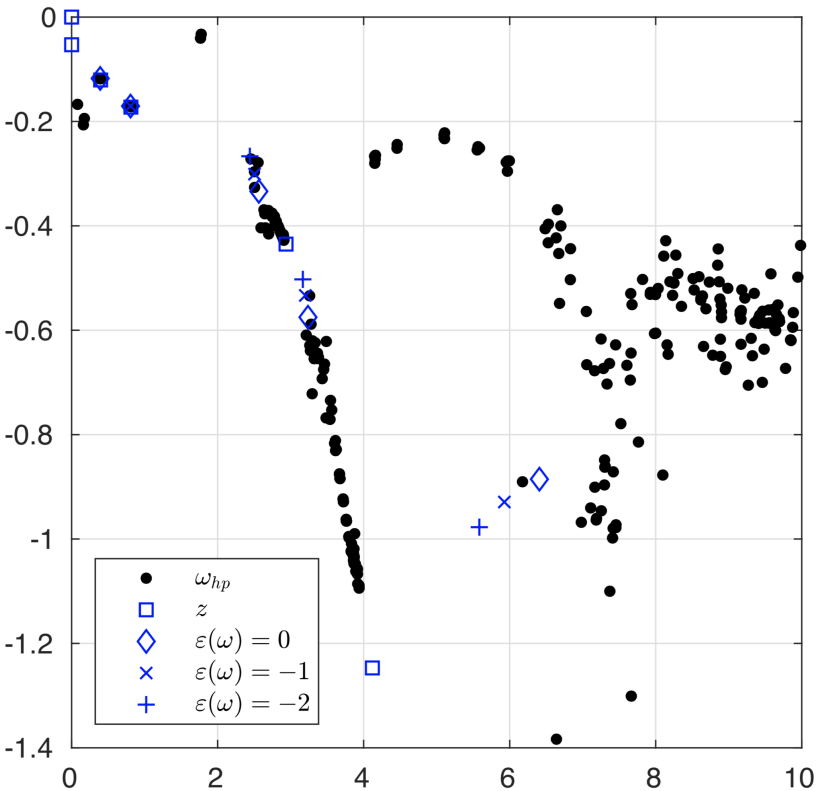
TM:

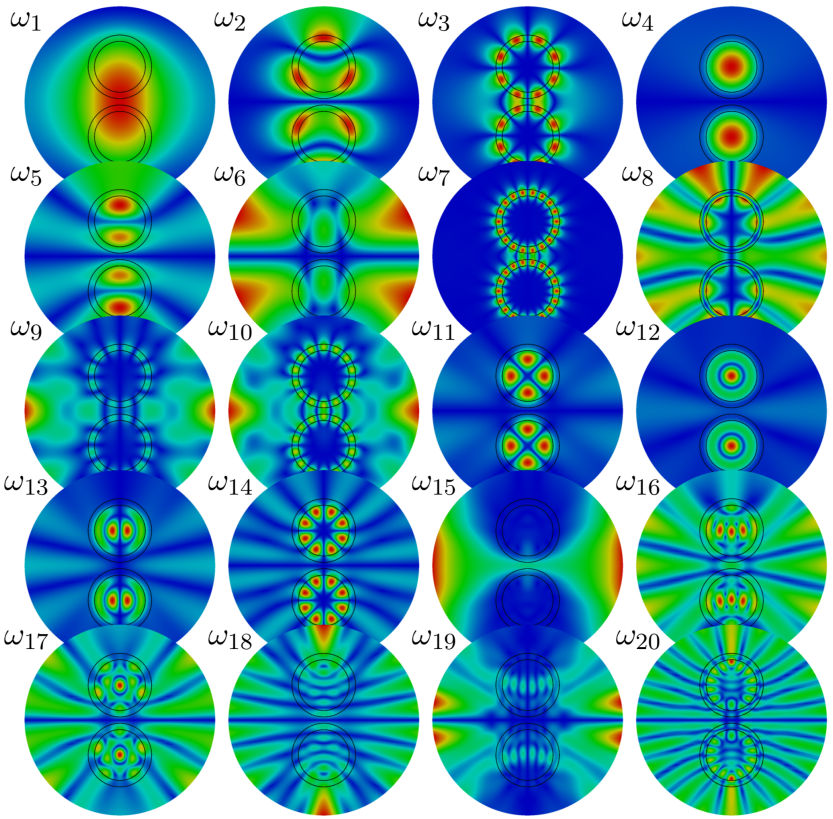
$$(\hat{A}_0 - \omega^2 \hat{A}_1 - \omega^2 \varepsilon(c,\omega)\hat{A}_2)x = 0$$

TE:

$$(\tilde{A}_0 - \omega^2 \tilde{A}_1 + \frac{1}{\varepsilon(c,\omega)}\tilde{A}_2)x = 0$$

Example 7: Exterior Helmholtz (Computed Solution)





Types of Eigenvalue Problems

HEP
$Ax = \lambda x$
$A = A^*$

SVD
$Av = \sigma u$
A rectangular

NHEP
$Ax = \lambda x$
$A \neq A^*$

GHEP
$Ax = \lambda Bx$
$A = A^*, B = B^*$ pos.-def.

GNHEP
$Ax = \lambda Bx$
$A \neq A^*$, or B not SPD

Linear

PEP
$P(\lambda)x = 0$
$P(\lambda) = A_0 + A_1\lambda + \cdots + A_d\lambda^d$

NEP
$T(\lambda)x = 0$
$T : \Omega \to \mathbb{C}^{n\times n}$

Nonlinear

SLEPc: Scalable Library for Eigenvalue Problem Computations

A general library for solving large-scale sparse eigenproblems on parallel computers

- Linear eigenproblems (standard or generalized, real or complex, Hermitian or non-Hermitian)
- Also support for SVD, PEP, NEP and more

$$Ax = \lambda x \qquad Ax = \lambda Bx \qquad Av_i = \sigma_i u_i \qquad T(\lambda)x = 0$$

Authors: J. E. Roman, C. Campos, L. Dalcin, E. Romero, A. Tomas

`http://slepc.upv.es`

Current version: 3.26 (released September 2026)

PETSc

Nonlinear Systems		
Line Search	Trust Region	...

Time Steppers				
Euler	Backward Euler	RK	BDF	...

Krylov Subspace Methods							
GMRES	CG	CGS	Bi-CGStab	TFQMR	Richardson	Chebychev	...

Preconditioners						
Additive Schwarz	Block Jacobi	Jacobi	ILU	ICC	LU	...

Matrices					
Compressed Sparse Row	Block CSR	Symmetric Block CSR	Dense	CUSPARSE	...

Vectors		
Standard	CUDA	ViennaCL

Index Sets		
General	Block	Stride

SLEPc

Nonlinear Eigensolver					
SLP	RII	N-Arnoldi	Interp.	CISS	NLEIGS

M. Function	
Krylov	Expokit

Polynomial Eigensolver			
TOAR	Q-Arnoldi	Linear-ization	JD

SVD Solver		
Cross Product	Cyclic Matrix	Thick R. Lanczos

Linear Eigensolver						
Krylov-Schur	Subspace	GD	JD	LOBPCG	CISS	...

Spectral Transformation			
Shift	Shift-invert	Filter	Precond.

BV	DS	RG	FN
...	...	...	...

PETSc $\sim$ 1.500.000 lines of code

SLEPc $\sim$ 200.000 lines of code

Application Areas

Google Scholar: 1500+ citations of main paper (ACM TOMS 2005)

Computational Physics, Materials Science, Electronic Structure	27 %
Computational Fluid Dynamics	18 %
PDE's, Numerical Methods	16 %
Computational Electromagnetics, Electronics, Photonics	7 %
Nuclear Engineering	6 %
Structural Analysis, Mechanical Engineering	6 %
Earth Sciences, Oceanology, Hydrology, Geophysics	4 %
Plasma Physics	4 %
Information Retrieval, Machine Learning, Graph Algorithms	4 %
Acoustics	3 %
Bioengineering, Computational Neuroscience	3 %
Dynamical Systems, Model Reduction, Inverse Problems	2 %
Visualization, Computer Graphics, Image Processing	1 %

Higher-level Software Packages

Software using SLEPc

General purpose:

- Automated modeling: GetDP, FEniCS, Firedrake
- Finite elements: libMesh, deal.II, MFEM, FreeFEM

Domain specific:

- Materials science: yambo, ELSI, DFT-FE, pyCTQW, dynamite
- Plasma physics: GENE, GYRO, PB3D, BOUT++
- Nuclear engineering: VERA, MOOSE, FEMFFUSION
- CFD: FELiCS, BROADCAST, helmholtz-x
- Bioengineering: OpenCMISS

Linear Eigenvalue Problems

Compute a few eigenpairs (x, λ) of $Ax = \lambda x$ or $Ax = \lambda Bx$

`EPS` is the solver class for linear eigenproblems

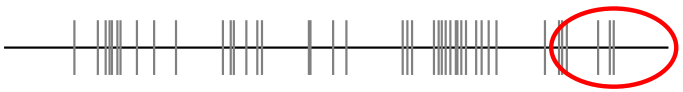
- Default solver is Krylov-Schur
  - Thick-restart Lanczos in the symmetric case
  - Mathematically equivalent to ARPACK's method
- User specifies number of eigenpairs (`nev`), subspace dimension (`ncv`), tolerance, selected part of spectrum
- Also: initial guesses, deflation space

$Ax = \lambda Bx$ solved as $B^{-1}Ax = \lambda x$

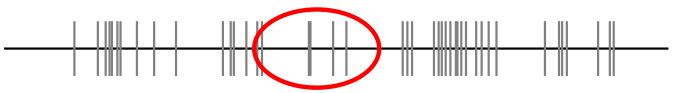
- Linear solves for B done via PETSc linear solvers (including external packages such as MUMPS, etc.)

Selected Part of the Spectrum

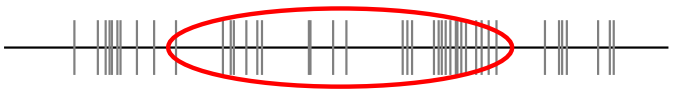
A few eigenvalues in the periphery of the spectrum



A few eigenvalues in the interior of the spectrum



A large percentage of eigenvalues (usually interior)



Families of Eigensolvers

Krylov methods - Arnoldi and variants

- Good to approximate several exterior eigenvalues
- For interior eigenvalues, need to factorize a matrix

Preconditioned methods - Davidson and variants

- Use a preconditioner instead of a factorization
- Compute one eigenvalue at a time

Conjugate gradient methods - LOBPCG and variants

- Specific for leftmost eigenvalues
- Can also use a preconditioner

Contour integral methods - FEAST and variants

- Returns all eigenvalues contained in a region
- Need one factorization per integration point

Interior Eigenvalues with Spectral Transformation

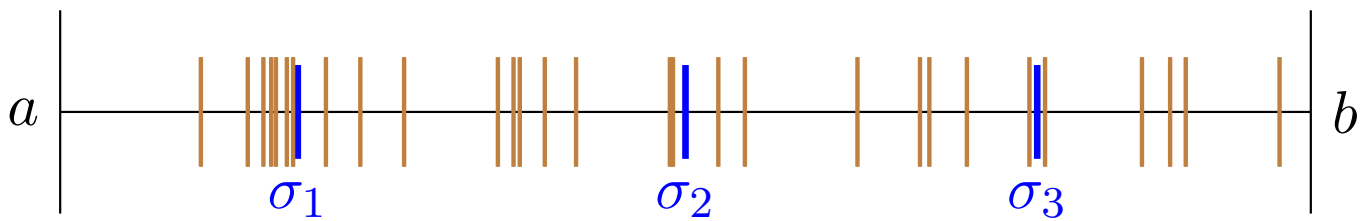
Shift-and-invert is used to compute interior eigenvalues

$$Ax = \lambda Bx \quad \Longrightarrow \quad (A - \sigma B)^{-1} Bx = \theta x$$

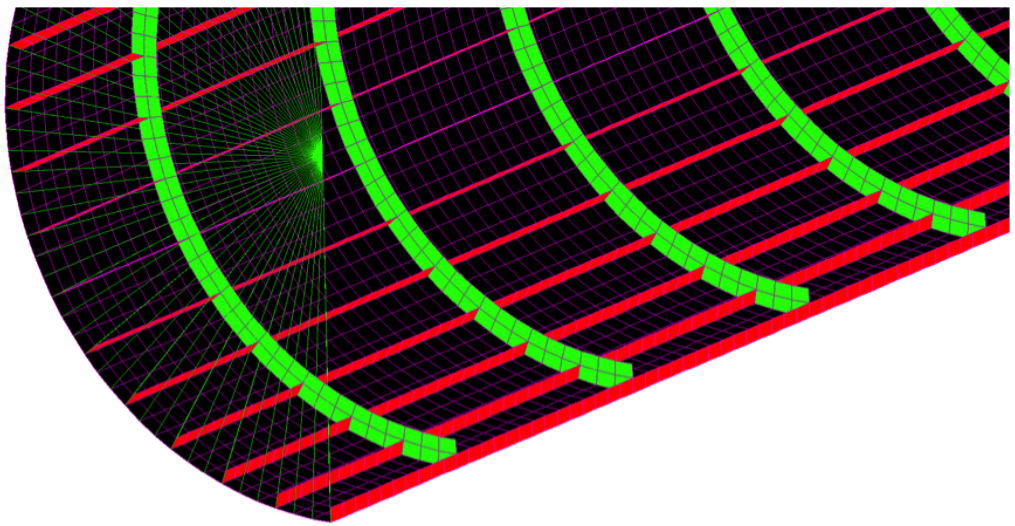
- Trivial mapping of eigenvalues: $\theta = (\lambda - \sigma)^{-1}$
- Eigenvectors are not modified
- Implicit inverse $(A - \sigma B)^{-1}$ via linear solves

For many eigenvalues in HEP/GHEP, use several shifts σ

- *Spectrum slicing*: smart sweep of the interval

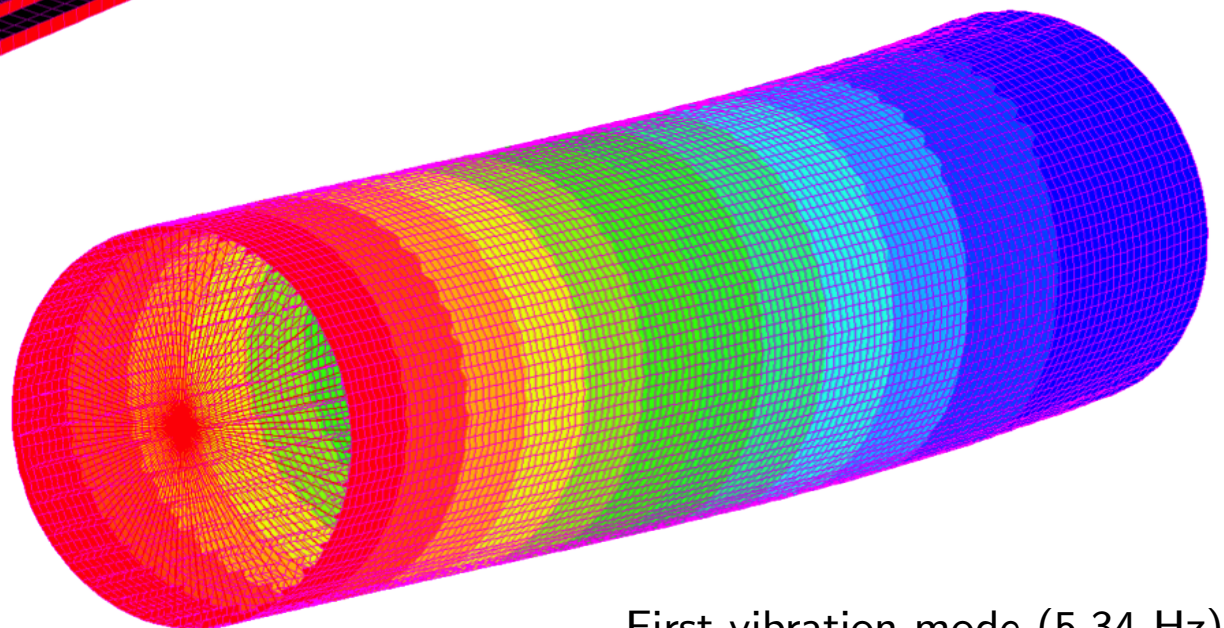


Example 8: Aircraft Fuselage



Simplified but realistic model: cylinder with skin, frames, and stringers

Parametric, "scalable"



First vibration mode (5.34 Hz)

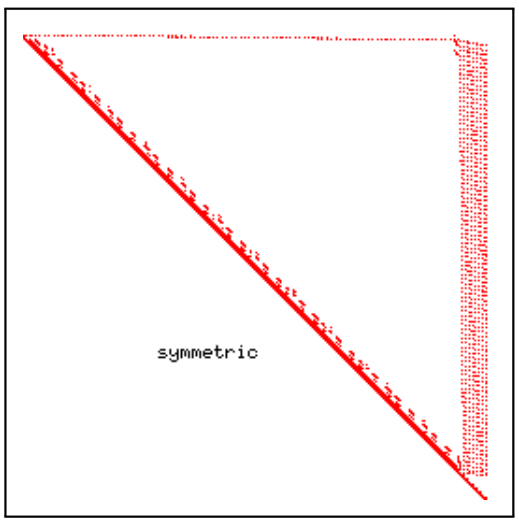
SAMTECH

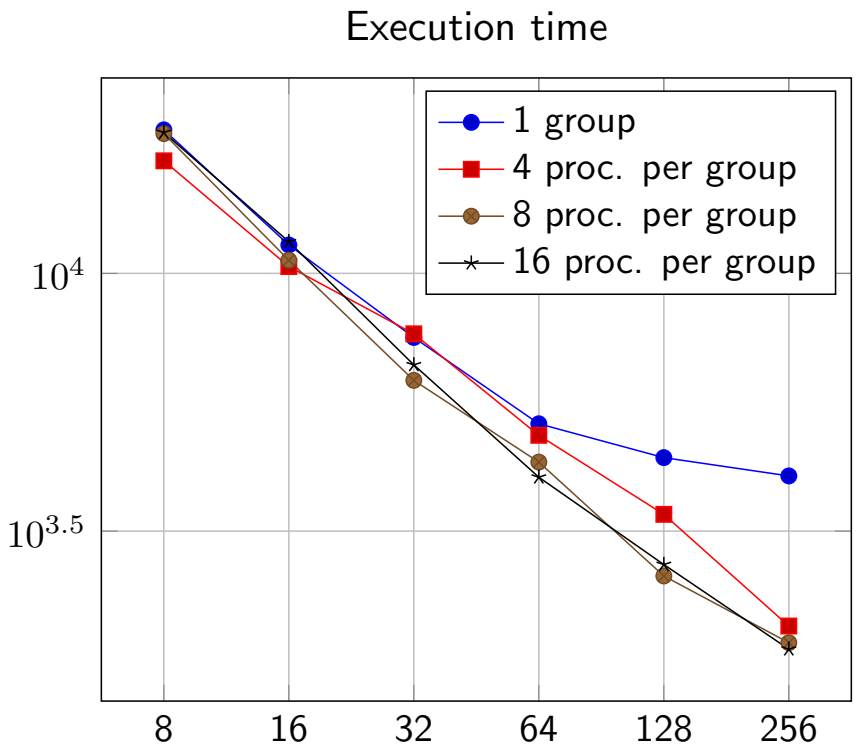
Example 8: Aircraft Fuselage (Results)

Analysis of frequency range [0–60] Hz

Problem dimension: $n = 1,036,698$ (∼29 million nonzeros)

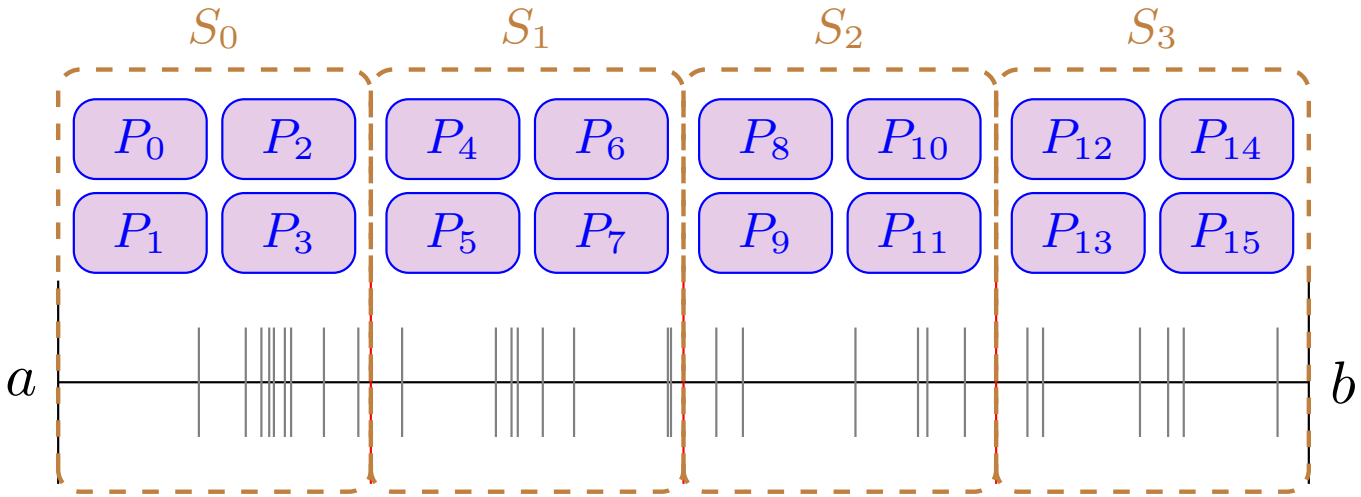
- Eigenvalues in interval: 1989
- Max multiplicity: 2
- B singular





Multi-Communicator Version

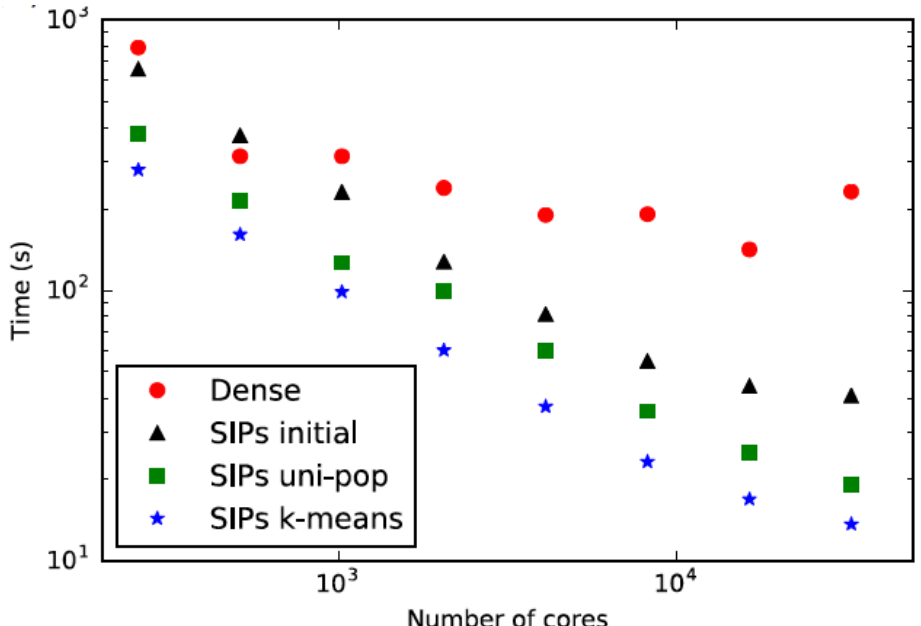
Idea: split in several sub-intervals, assign to process groups



DFTB calculation for metallic carbon nanotube

n=64,000

50% eigenpairs



Interior Eigenvalues with Filters

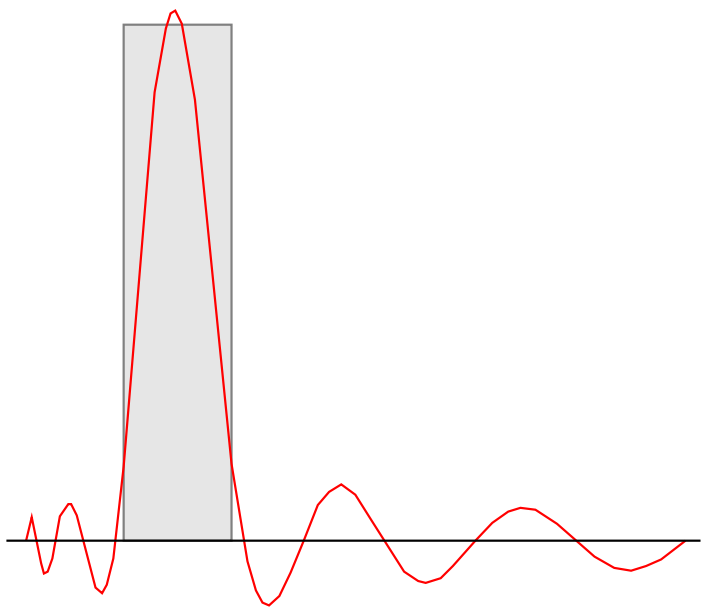
Pitfalls of shift-and-invert:

- Direct solvers have high cost, limited scalability
- *Inexact* shift-and-invert (i.e., with iterative solver) not robust

Alternatives: preconditioned eigensolvers or filters

Polynomial filter: $\rho(A)x = \theta x$

- $\rho(\cdot)$ is a high-degree Chebyshev polynomial
- Acts as a band-pass filter
- Eigenvalues outside the wanted interval are damped



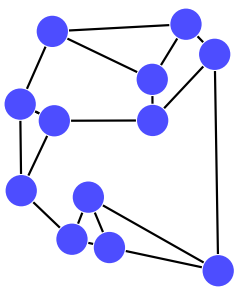
Anderson Localization

Collaboration with Manuel Pino (IFF-CSIC, Madrid)

M. Pino, J. E. Roman (2024), "Correlated volumes for extended wavefunctions on a random-regular graph," PRB 109:184204

Non-interacting models

- A single particle hopping in a disordered random regular graph, RRG



matrix size = number of nodes

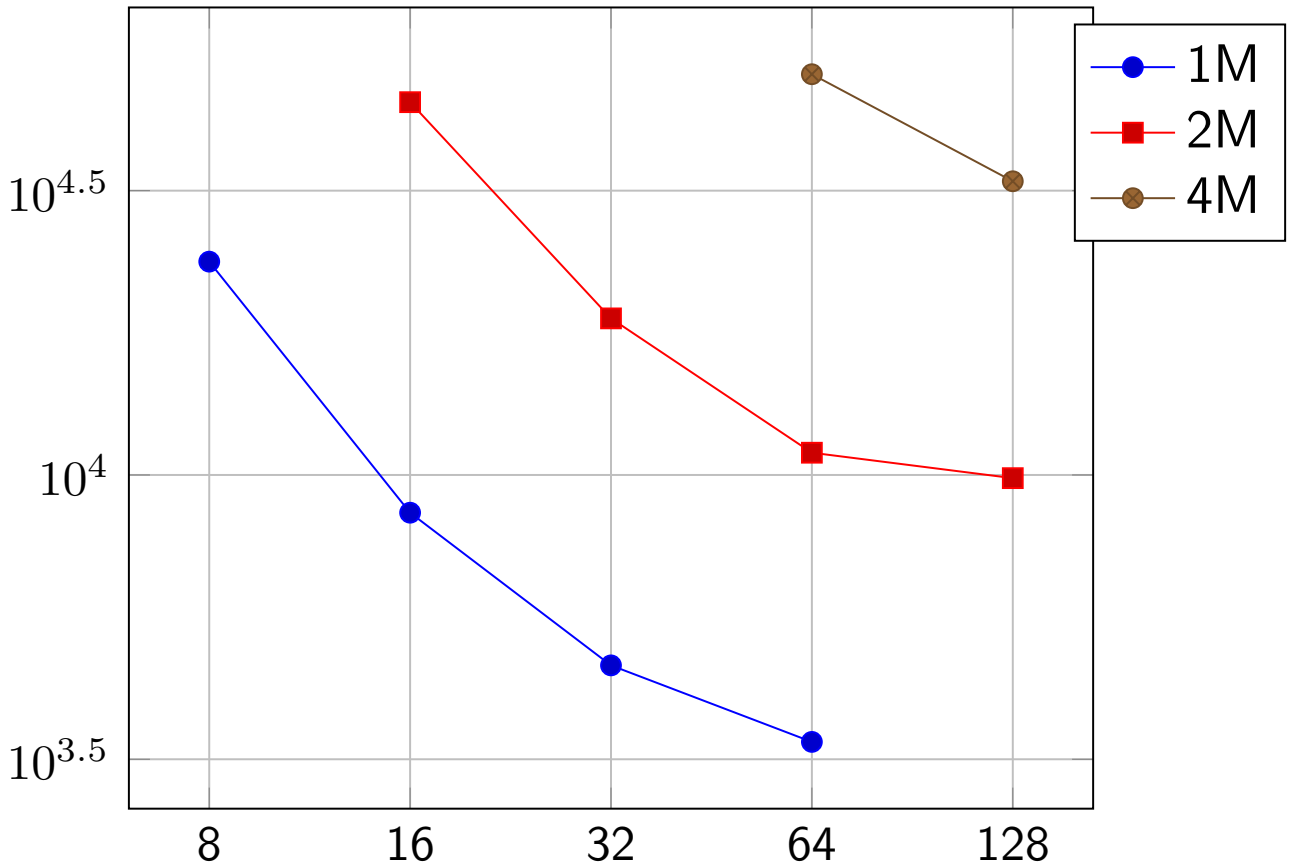
Matrix has random diagonal, 3 off-diagonal entries per row at random locations

Compute eigenvalues closest to $\sigma = 0$



Anderson Localization: MPI Scaling for RRG Problems

For 1M and 2M $\rightarrow d = 3000$, `ncv`=120, $\tau = 0.0001$
For 4M $\rightarrow d = 4000$, `ncv`=120, $\tau = 0.00005$



Structured Eigensolvers

Structured matrices: n^2 entries depend on less than n^2 parameters
Examples: skew-symmetric, Hamiltonian, orthogonal, symplectic

Preserving structure...

- helps preserve physically relevant symmetries in eigenvalues
- may improve the accuracy and efficiency of the eigensolver

Example: QEP for gyroscopic systems, eigenvalues in quadruples $\{\lambda, -\lambda, \bar{\lambda}, -\bar{\lambda}\}$

In Krylov methods, this implies

- Build subspace such that projected problem has same structure
- Do not ruin this structure when restarting

Recent interesting structures: Bethe-Salpeter, linear response

Bethe-Salpether Equation

Block-structured Hamiltonian matrix:

$$H = \begin{bmatrix} R & C \\ -C^H & -R^T \end{bmatrix} = \begin{bmatrix} R & C \\ -\bar{C} & -\bar{R} \end{bmatrix}$$

R Hermitian, C complex symmetric

Helper function `MatCreateBSE()` creates a `MATNEST` from R, C

$$H = \Omega\tilde{H}, \quad \Omega = \begin{bmatrix} I & 0 \\ 0 & -I \end{bmatrix}, \quad \tilde{H} = \begin{bmatrix} R & C \\ C^H & R^T \end{bmatrix} \text{ (HPD)}$$

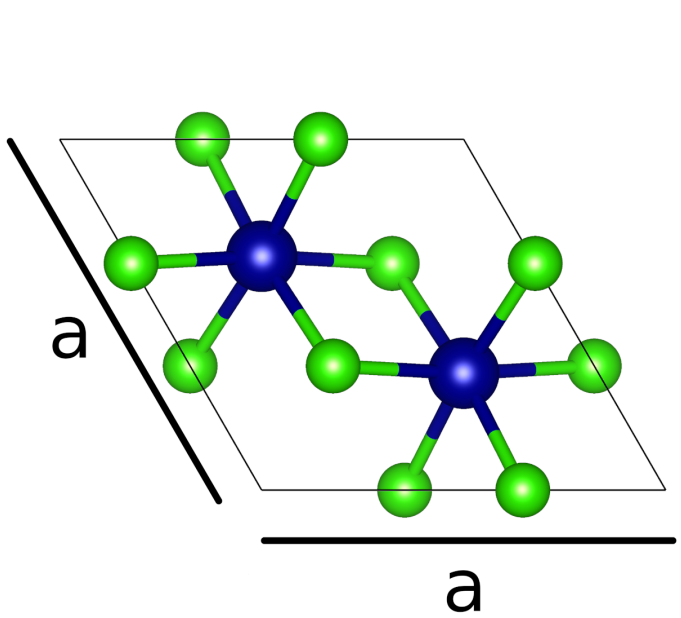
Properties:

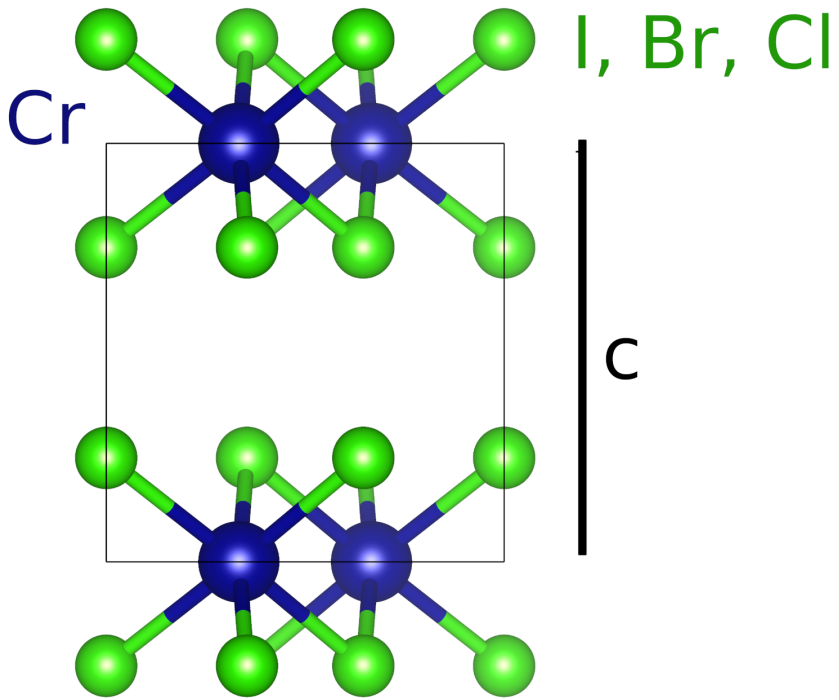
- H pseudo-Hermitian with respect to both $\tilde{H}$ and Ω
- Eigenvalues appear in pairs $\{\lambda_i, -\lambda_i\}$
- Trivial relations between x_i^+ and x_i^-, and between x_i and y_i

BSE - Test Problem

CrI_3 monolayer — a 2D magnetic material

Structure of the crystal (unit cell):

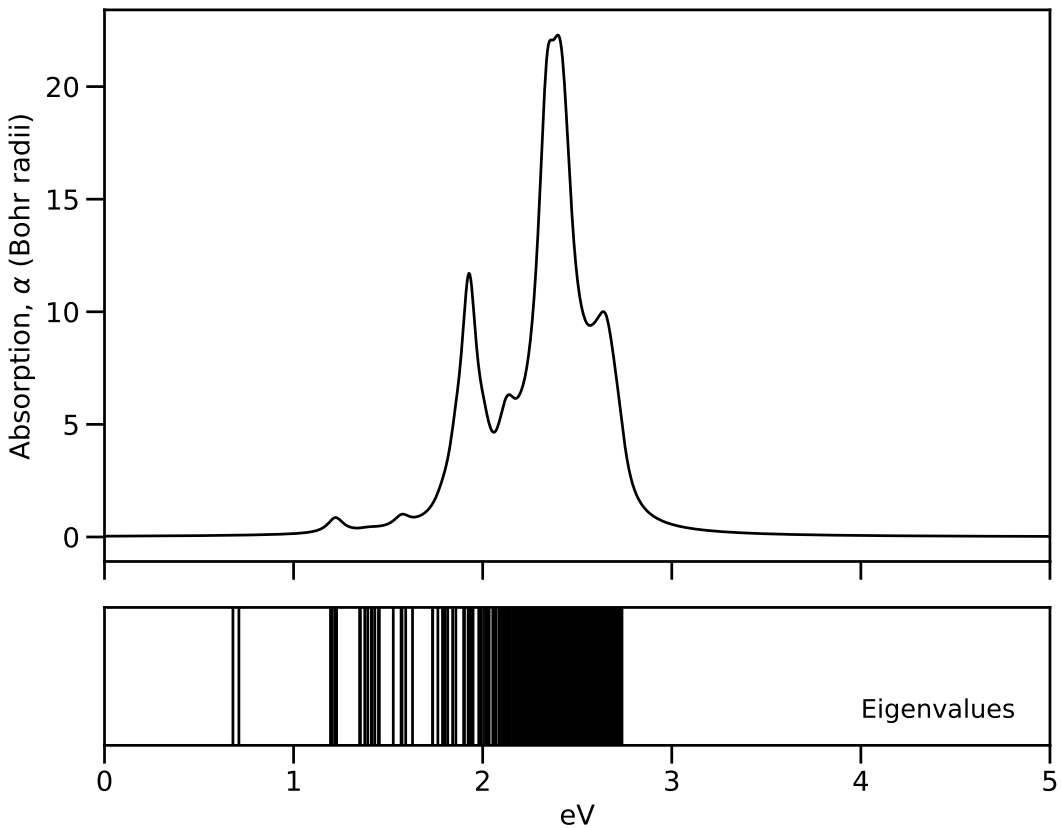




Iodine can be replaced by Br or Cl

BSE - Absortion Spectrum

Solution of Bethe-Salpeter equation with spin orbit coupling



← Absortion spectrum

← Computed eigenvalues

BSE - Computational Times

Single precision

Size of matrix: 23040 Number of eigenvalues: 100

CPU (Intel Core i9-10940X 16GB)

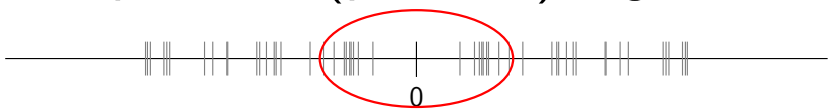
solver	procs	iter	time
two-sided	1	58	6,071 s
non-hermitian	1	49	2,560 s
bse	1	50	629 s
two-sided	8	59	1,964 s
non-hermitian	8	50	838 s
bse	8	50	413 s

GPU (NVIDIA RTX A2000 12GB)

solver	iter	time
two-sided	60	248 s
non-hermitian	49	105 s
bse	49	48 s

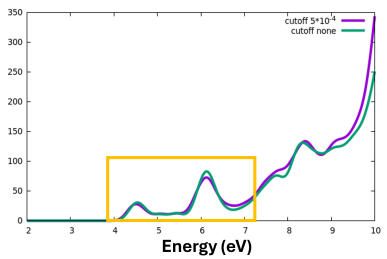
Linear Response - Test cases

Compute 50 (positive) eigenvalues

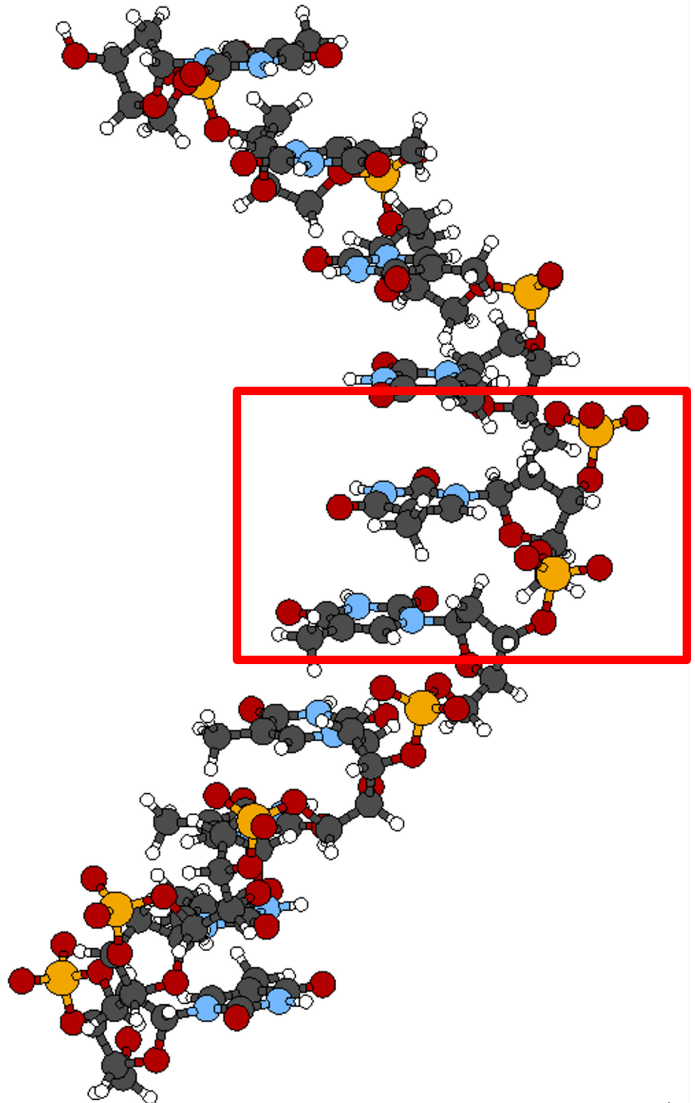


- **Na_2** molecule
  Zhaojun Bai (turboTDDFT code in QE)
  $n = 1862,\ \lambda_1 = 0.64,\ \lambda_n = 11.19$
- Single-strand **DNA** helix
  Luca Sementa (TDDFT via CP2K)
  $n = 12000,\ \lambda_1 = 0.12,\ \lambda_n = 1.47$

  Sparsify: discard $|h_{ij}| < 5 \cdot 10^{-4}$, 3.2% nnz, minor deviations in absorption diagram



- **Pentadiag**
  Synthetic problem
  $n = 50000,\ \lambda_1 = 2.30,\ \lambda_n = 4.89$



Linear Response - Results

name	solver	its	time	residual	bi-orthog
Na_2	teng	18	3.25	$2.81 \cdot 10^{-9}$	$1.80 \cdot 10^{-15}$
Na_2	zhong	18	3.27	$3.45 \cdot 10^{-9}$	$4.13 \cdot 10^{-10}$
Na_2	shao	18	3.30	$2.81 \cdot 10^{-9}$	$2.25 \cdot 10^{-15}$
Na_2	gruning	18	3.34	$3.45 \cdot 10^{-9}$	$4.13 \cdot 10^{-10}$
Na_2	non-Hermitian	19	7.61	$9.23 \cdot 10^{-9}$	$5.81 \cdot 10^{-08}$
DNA	teng	36	28.11	$2.05 \cdot 10^{-8}$	$4.65 \cdot 10^{-15}$
DNA	zhong	36	28.34	$3.64 \cdot 10^{-9}$	$5.57 \cdot 10^{-10}$
DNA	shao	36	28.39	$2.05 \cdot 10^{-8}$	$4.58 \cdot 10^{-15}$
DNA	gruning	36	28.88	$3.64 \cdot 10^{-9}$	$5.57 \cdot 10^{-10}$
DNA	non-Hermitian	38	65.45	$9.47 \cdot 10^{-9}$	$1.66 \cdot 10^{-08}$
Pentadiag	teng	4147	625.55	$2.24 \cdot 10^{-9}$	$7.16 \cdot 10^{-14}$
Pentadiag	zhong	4347	718.88	$5.22 \cdot 10^{-9}$	$4.71 \cdot 10^{-10}$
Pentadiag	shao	4091	822.16	$2.25 \cdot 10^{-9}$	$6.55 \cdot 10^{-14}$
Pentadiag	gruning	4349	988.31	$5.28 \cdot 10^{-9}$	$3.15 \cdot 10^{-10}$
Pentadiag	non-Hermitian	2011	3121.20	$9.88 \cdot 10^{-9}$	$5.54 \cdot 10^{-02}$

Full Diagonalization

For problems that need to compute the full spectrum, SLEPc provides interfaces to third-party libraries

- Same interface, no need to change the program

Symmetric eigenvalue problem

- ScaLAPACK: standard library for distributed-memory dense linear algebra, included in Intel MKL
- ELPA: much faster and more scalable
- Elemental: uses a different data distribution

SVD

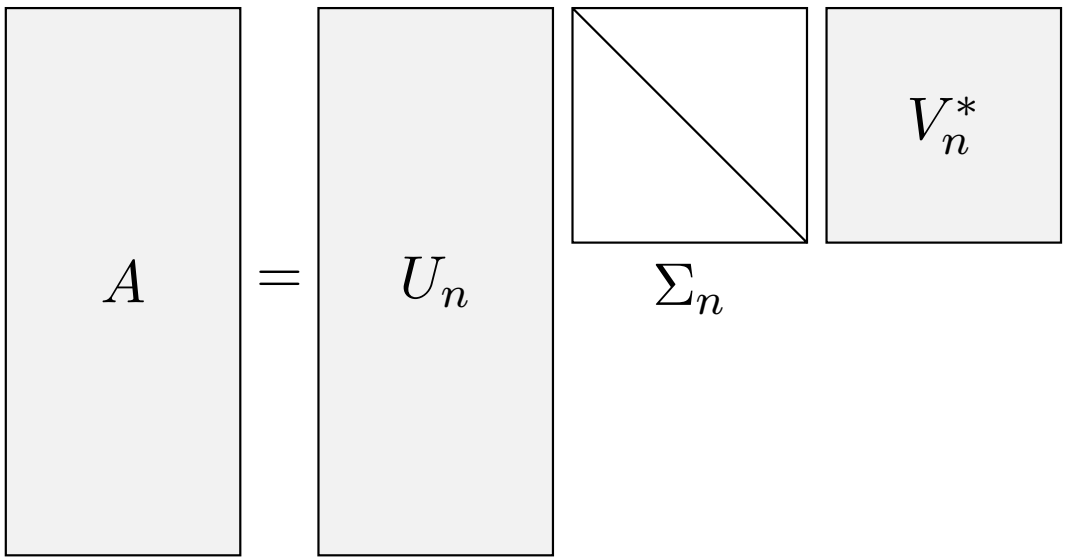
- ScaLAPACK, KSVD and Elemental

Command-line Processing

Command-line options allows great flexibility without code changes

```
$ ./ex1 -eps_type krylovschur -eps_smallest_real
$ ./ex1 -eps_nev 6 -eps_ncv 24
$ ./ex1 -eps_tol 1e-12 -eps_max_it 2000
$ ./ex1 -eps_type lobpcg -eps_lobpcg_blocksize 4
$ ./ex1 -eps_type gd -eps_gd_plusk 1
$ ./ex1 -eps_type arpack -eps_view_values draw
$ ./ex1 -eps_target 2.1 -st_type sinvert
$ ./ex1 -eps_interval 1.8,2.5 -st_type sinvert
        -st_pc_type cholesky
$ ./ex1 -eps_view -eps_monitor -log_view
```

SVD: Singular Value Decomposition



In SLEPc: *partial* SVD (only a subset of the singular triplets)

- Largest singular values, used for low-rank approximation of A
- Can also compute smallest singular values

Recently added support for the GSVD of two matrices

MFN: Matrix Function

Many applications require the computation of $y = f(A)v$ for

- Brownian dynamics simulation, $f(A) = A^{-\frac{1}{2}}$
- Ensemble Kalman filter, $f(A) = (A + A^{\frac{1}{2}})^{-1}$
- Time-dependent Schrödinger equation, $f(A) = e^A$
- Compute rightmost eigenvalues of A via e^A

Krylov methods can be a good approach

$$AV_m = V_{m+1}\underline{H}_m, \qquad y \approx \|v\|_2 V_m f(H_m)e_1$$

What is needed:

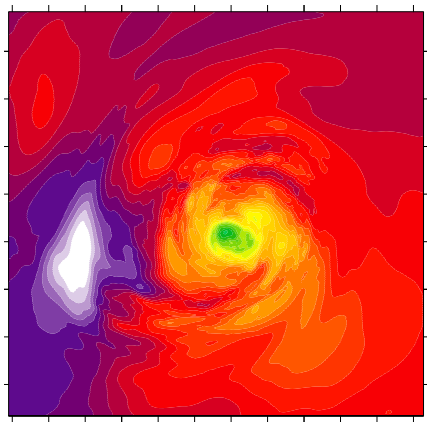
- Efficient construction of the Krylov subspace
- Computation of $f(X)$ for a small dense matrix $\rightarrow$ `FN`

Example 9: Ensemble Kalman Filter

Assimilation of satellite observations, $f(D) = (D + \sqrt{D})^{-1}$

- Compute $f(D)b$ for 30 RHS for a matrix of order 17,733
- Krylov restart parameter: $m = 150$, does 3 restarts per RHS
- `FN` using Denman–Beavers for $\sqrt{H_{km}}$
- GPU cluster, 1 process/node and 16 threads/process

		FN		MFN		Total
	Procs	Time	GF/s	Time	GF/s	Time
CPU	1	8.7	21.3	2606.8	1.4	2616.7
	16	31.4	-	261.7	24.9	278.6
GPU	1	16.7	11.1	168.0	22.2	177.8
	16	19.9	-	49.0	132.8	66.0



Exponential Integrators

Time-evolution of the time-dependent Schrödinger equation

$$i\frac{\partial}{\partial t}\Psi(t) = H(t)\Psi(t)$$

can be obtained with a time-stepping scheme

$$\Psi(t+\Delta t) = \exp(-i\Delta t H(t+\tfrac{\Delta t}{2}))\Psi(t)$$

Sample users:

H. P. Lüschen et al., "Observation of slow dynamics near the many-body localization transition in one-dimensional quasiperiodic systems", Phys. Rev. Lett. 119 (2017)

D. J. Luitz et al., "Absence of dynamical localization in interacting driven systems", SciPost Phys. 3 (2017)

N. Macé et al., "Many-body localization in a quasiperiodic Fibonacci chain", SciPost Phys. 6 (2019)

X. Antoine and E. Lorin, "A simple pseudospectral method for the computation of the time-dependent Dirac equation with Perfectly Matched Layers", J. Comput. Phys. 395 (2019)

B. Ye et al., "Emergent hydrodynamics in nonequilibrium quantum systems", Phys. Rev. Lett. 125 (2020)

Concluding Remarks

SLEPc is a mature library

- Many solvers for the linear eigenvalue problem (`EPS`)
- Specific support for SVD and GSVD (`SVD`)
- Solvers for the matrix exponential and other functions (`MFN`)
- Parallelism with MPI and also support for GPU

Ongoing and future additions

- Better structure-preserving eigensolvers (Bethe-Salpeter, LREP)
- Support for eigenvector-dependent problems

Acknowledgements: supported by grant PID2022-139568NB-I00

AGENCIA ESTATAL DE INVESTIGACIÓN