GPU support in the SLEPc library for eigenvalue computations

Jose E. Roman

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

10th RES Users Conference, León – September, 2016

Outline

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 nonlinear eigenproblems and other problems

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

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

`http://slepc.upv.es`

Current version: 3.7 (released May 2016)

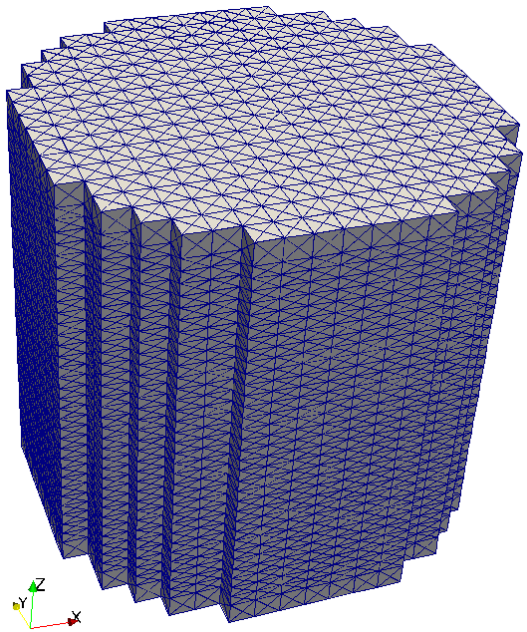
Use Case: Neutron Difusion Equation in Nuclear Eng.

Neutron power in nuclear reactor cores

- Commercial reactors such as PWR
- Both steady state and transient
- Goal: assure safety

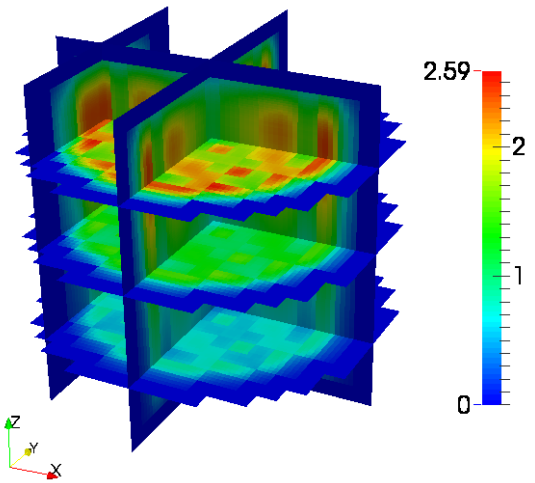
Lambda Modes Equation

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



Current trends

- Complex geometries, unstructured meshes, FVM
- Coupled neutronic-thermalhydraulic calculations



Applications

Google Scholar: 420 citations of main paper (ACM TOMS 2005)

Computational Physics, Materials Science, Electronic Structure	24 %
Computational Fluid Dynamics	13 %
PDE's, Numerical Methods	10 %
Plasma Physics	9 %
Computational Electromagnetics, Electronics, Photonics	8 %
Nuclear Engineering	6 %
Earth Sciences, Oceanology, Hydrology, Geophysics	6 %
Information Retrieval, Machine Learning, Graph Algorithms	6 %
Structural Analysis, Mechanical Engineering	5 %
Acoustics	4 %
Visualization, Computer Graphics, Image Processing	3 %
Dynamical Systems, Model Reduction, Inverse Problems	3 %
Bioengineering, Computational Neuroscience	2 %
Astrophysics	1 %

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	Cayley	Precond.

BV
...

DS
...

RG
...

FN
...

EPS: Eigenvalue Problem Solver

Compute a few eigenpairs (x, λ) of

Standard Eigenproblem

$$Ax = \lambda x$$

Generalized Eigenproblem

$$Ax = \lambda Bx$$

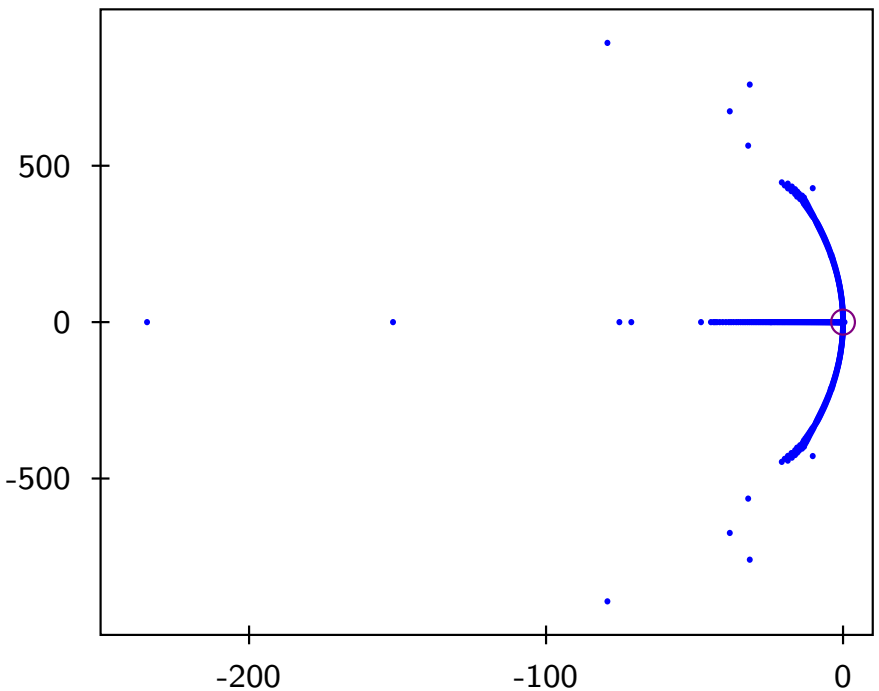
User can specify:

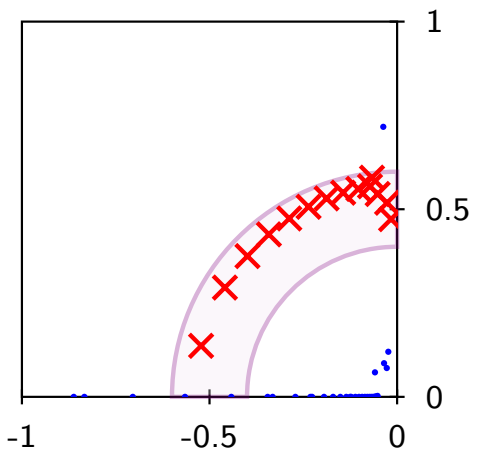
- Number of eigenpairs (`nev`), tolerance, ...
- Solver: Krylov-Schur, GD, JD, LOBPCG, CISS, SI, ...
- Selected part of spectrum
  1. Eigenvalues with largest (smallest) magnitude
  2. Those with largest (smallest) real (imaginary) part
  3. Those closest to a given target value τ of the complex plane
  4. All eigenvalues in an interval or region of the complex plane
  5. According to a user-defined criterion

Example of Computing Interior Eigenvalues

MHD1280 problem with the contour-intergral solver

- Alfvén spectra: eigenvalues in intersection of the branches





RG=ring, center=0, radius=0.5,
width=0.2, angle=0.25..0.5

PEP: Polynomial Eigenvalue Problem

$$P(\lambda)x = 0 \qquad P(\lambda) = A_0 + A_1\lambda + A_2\lambda^2 + \cdots + A_d\lambda^d$$

Explicit linearization: $L(\lambda) = \mathcal{L}_0 - \lambda\mathcal{L}_1$, with $L(\lambda)y = 0$ and

$$\mathcal{L}_0 = \begin{bmatrix} & I & & \\ & & \ddots & \\ & & & I \\ -A_0 & -A_1 & \cdots & -A_{d-1} \end{bmatrix} \quad \mathcal{L}_1 = \begin{bmatrix} I & & & \\ & \ddots & & \\ & & I & \\ & & & A_d \end{bmatrix} \quad y = \begin{bmatrix} x \\ x\lambda \\ \vdots \\ x\lambda^{d-1} \end{bmatrix}$$

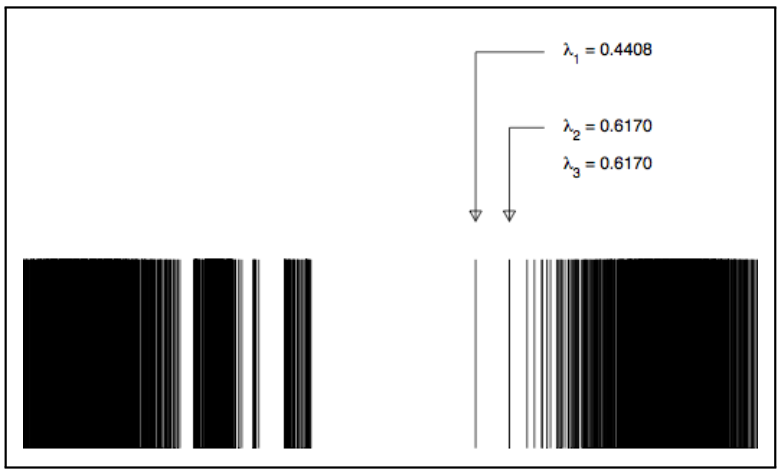
Compute an eigenpair (y, λ) of $L(\lambda)$, then extract x from y

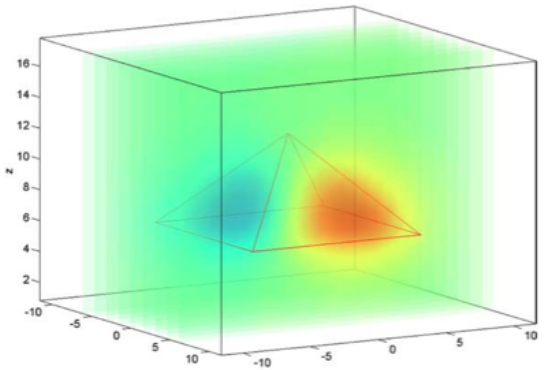
- Drawback: dimension of linearized problem is dn
- Alternatives: implicit linearization, Jacobi-Davidson

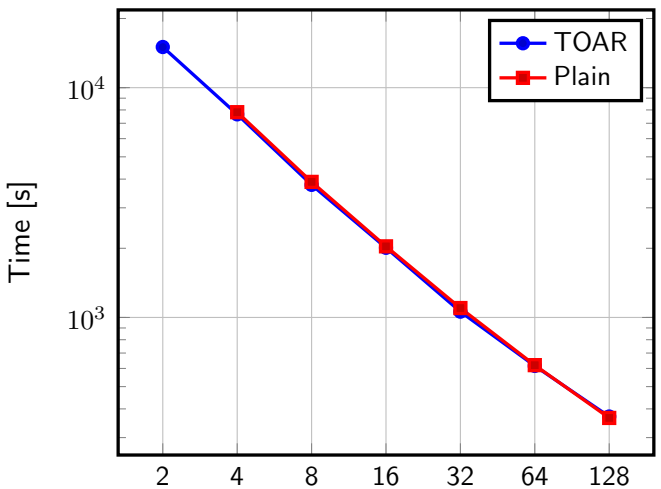
PEP Example: Quantum Dot Simulation

3D pyramidal quantum dot discretized with finite volumes

Tsung-Min Hwang et al. (2004). "Numerical Simulation of Three Dimensional Pyramid Quantum Dot," Journal of Computational Physics, 196(1): 208-232.







Quintic polynomial, $n \approx 12$ mill.

Scaling for tol=10^{-8}, nev=5, ncv=40 with inexact shift-and-invert (`bcgs+bjacobi`)

NEP: General Nonlinear Eigenproblems

$$T(\lambda)x = 0$$

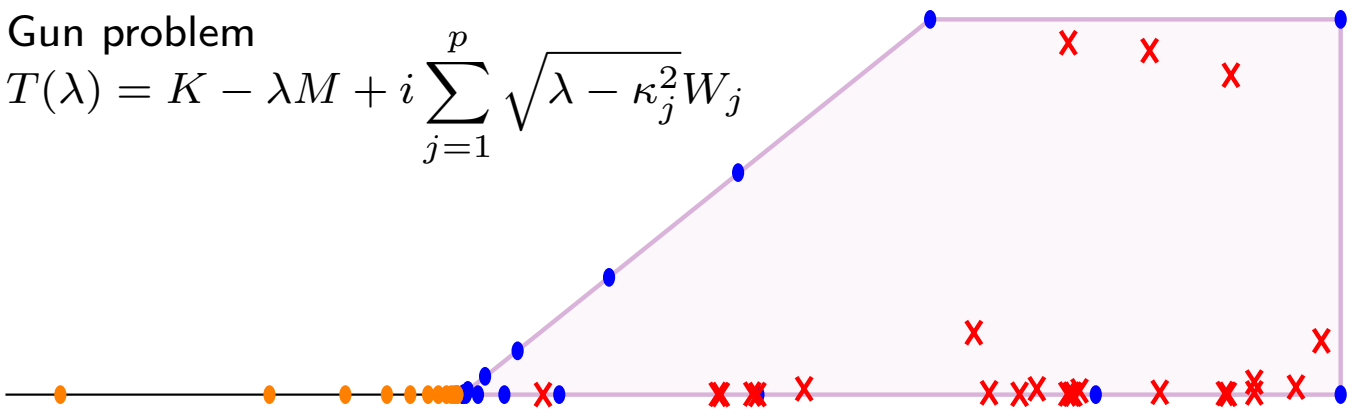
T: matrix-valued function analytic on $\Omega \subset \mathbb{C}$

Solvers:

- Single-vector iterations
- Chebyshev polynomial interpolation (uses `PEP`)
- Contour integral
- NLEIGS: rational interpolation with linearization

Gun problem

$$T(\lambda) = K - \lambda M + i \sum_{j=1}^{p} \sqrt{\lambda - \kappa_j^2} W_j$$



Other Functionality

SVD: partial singular value decomposition, $A = U\Sigma V^*$

- Low-rank approximation of matrices
- Ill-posed least-squares problems (e.g., medical imaging)
- Latent-semantic indexing (big data analysis)
- Stability analysis via pseudo-spectra

MFN: compute the action of a matrix function, $y = f(A)v$

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

PETSc's Approach to GPU Computing

Low level operations can be done on GPU, with NVIDIA CUDA

- Basic dense operations with cuBLAS
- Sparse operations with cuSPARSE

`VECCUDA`: A special `Vec` whose array is mirrored in the GPU

- Supports all operations in the `Vec` interface
- Can be activated at run time: `-vec_type cuda`

Memory coherence flag:

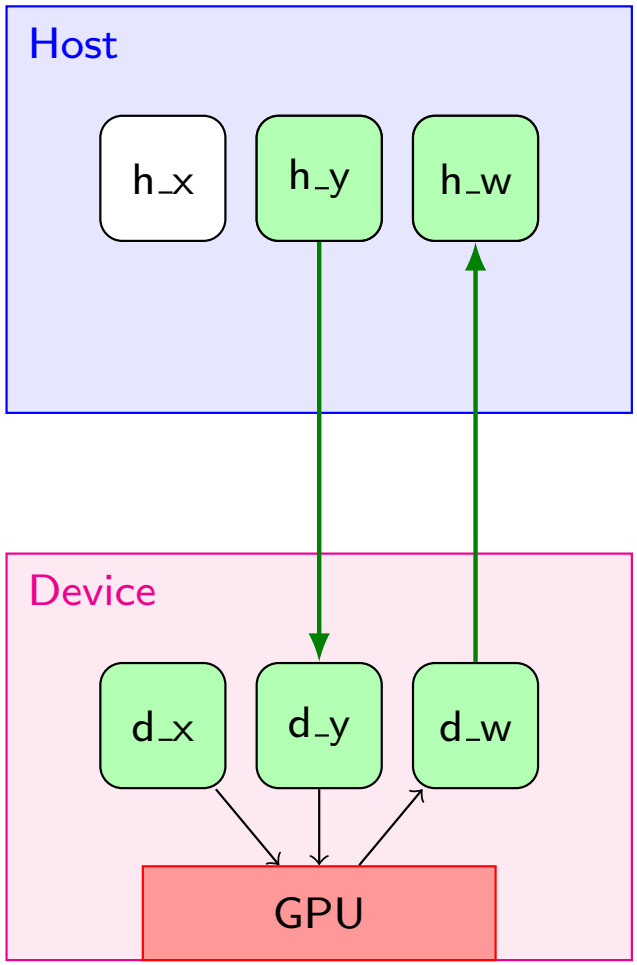
`UNALLOCATED`	No allocation on the GPU
`GPU`	Values on GPU are current
`CPU`	Values on CPU are current
`BOTH`	Values on both are current

`MATAIJCUSPARSE`

- Similar to `Vec`, but some operations are done on CPU only

Example of Vec Allocation and Coherence

```
VecCreate(PETSC_COMM_WORLD,&x);
VecSetSizes(x,PETSC_DECIDE,n);
VecSetType(x,VECCUDA);
VecSetFromOptions(x);
VecDuplicate(x,&y);
VecDuplicate(x,&w);
VecSet(x,1.0);
VecGetArray(y,&py);
for (i=0;i<n;i++) py[i] = ...
VecRestoreArray(y,&py);
VecWAXPY(w,2.0,x,y);
VecGetArrayRead(w,&pw);
for (i=0;i<n;i++) dest[i] = pw[i];
VecRestoreArrayRead(w,&pw);
VecDestroy(&x);
VecDestroy(&y);
VecDestroy(&w);
```



SLEPc Extension: Basis Vectors (BV)

BV provides the concept of a block of vectors that represent the basis of a subspace; sample operations:

`BVMult`	$Y \leftarrow \beta Y + \alpha XQ$
`BVDot`	$M \leftarrow Y^*X$
`BVMatMult`	$M \leftarrow AX$
`BVMatProject`	$M \leftarrow Y^*AX$
`BVScale`	$Y \leftarrow \alpha Y$

Goal: to increase arithmetic intensity (BLAS-2 vs BLAS-1)

```
$ ./ex9 -n 8000 -eps_nev 32 -log_summary -bv_type vecs
BVMult    32563 1.0 3.2903e+01 1.0 6.61e+10 1.0 0.0e+00 0.0e+00 ...   2009
BVDot     32064 1.0 1.6213e+01 1.0 5.07e+10 1.0 0.0e+00 0.0e+00 ...   3128

$ ./ex9 -n 8000 -eps_nev 32 -log_summary -bv_type mat
BVMult    32563 1.0 2.4755e+01 1.0 8.24e+10 1.0 0.0e+00 0.0e+00 ...   3329
BVDot     32064 1.0 1.4507e+01 1.0 5.07e+10 1.0 0.0e+00 0.0e+00 ...   3497
```

Even better in block solvers (LOBPCG): BLAS-3, `MatMatMult`

Sample Performance CPU vs GPU

Lin matrix (from UF Sparse Matrix Collection)

- Application: structural analysis
- Matrix size 256,000, banded with ∼1.8 million nonzeros
- Compute 500 eigenvalues

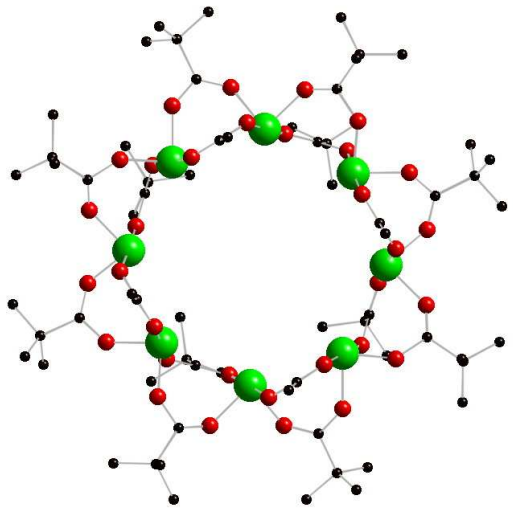
	double		single	
	CPU	GPU	CPU	GPU
Total	2849.6	313.3	1312.2	201.5
MatMult	29.6	2.4	22.9	2.0
BVDotVec	1402.8	161.3	627.5	116.6
BVMultVec	1371.4	124.0	630.6	64.9
BVMultInPlace	22.8	2.4	13.3	1.1
Other	23.1	23.3	17.9	17.0

Performance gain factor GPU vs CPU: **9.1** (double) / **6.5** (single)

Molecular Clusters

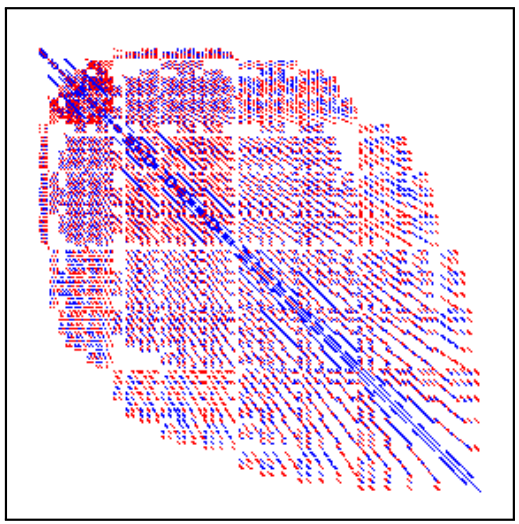
Analysis of high-nuclearity spin clusters

- Original code by ICMol (Valencia)
- Goal: study magnetic susceptibility and magnetization as well as inelastic neutron scattering spectra



Example: ring of 10 Ni(II) ions with antiferromagnetic interaction

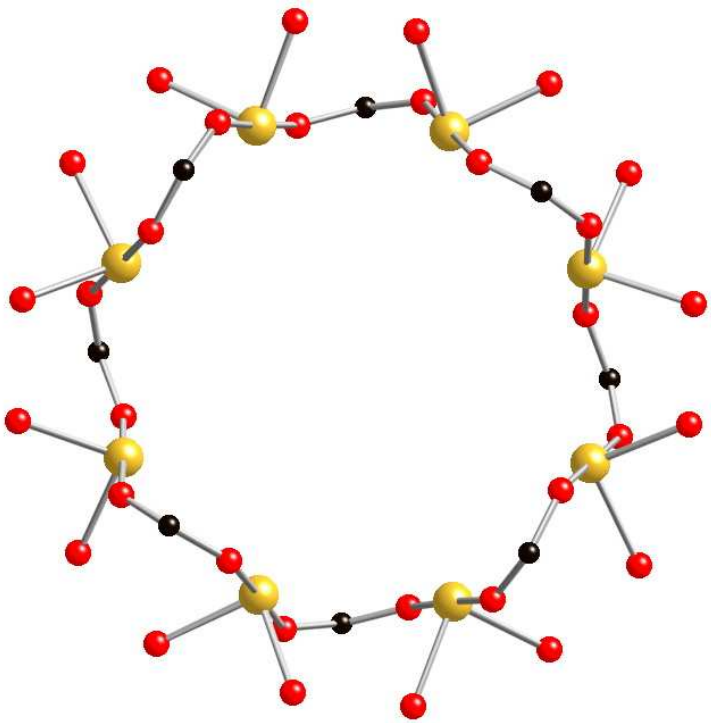
- Dimension: 59,049
- Nonzero elements: $\sim$35 million
- Real, symmetric, indefinite
- 600 leftmost eigenvalues

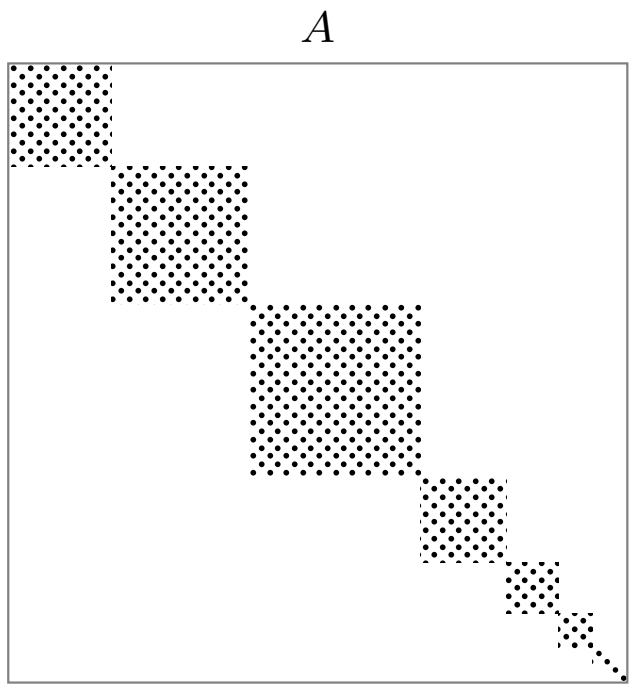


Molecular Magnetism: Isotropic Systems

Isotropic analysis discards anisotropic exchange interactions

- Spins are decoupled, can reach much larger problem sizes





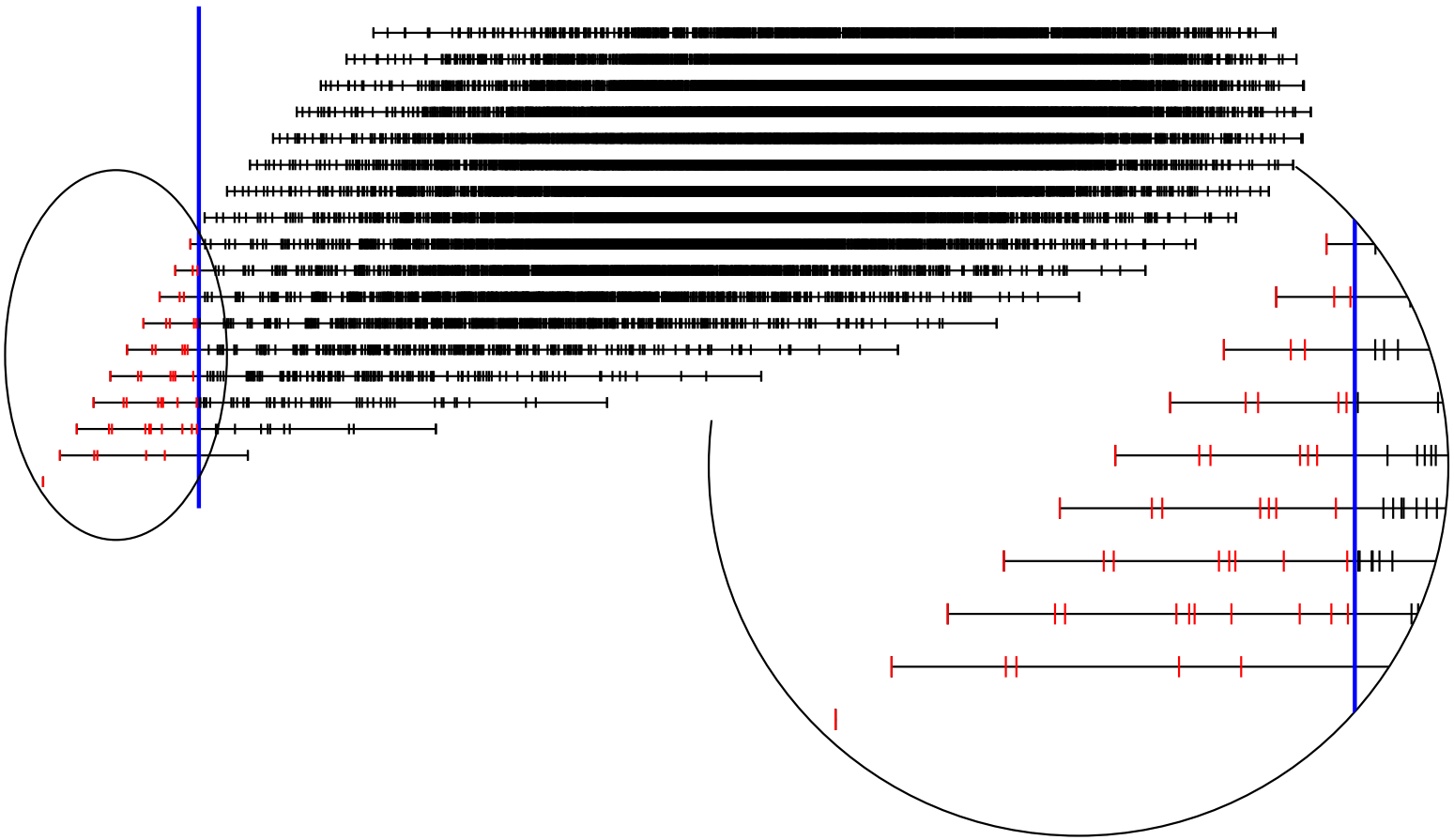
Hamiltonian matrix, blocks are sparse

Molecular Magnetism: Energy Levels

Energy levels of 7 manganese ring with antiferromagnetic coupling

Blue: requested energetic cut

Red: eigenvalues to be computed

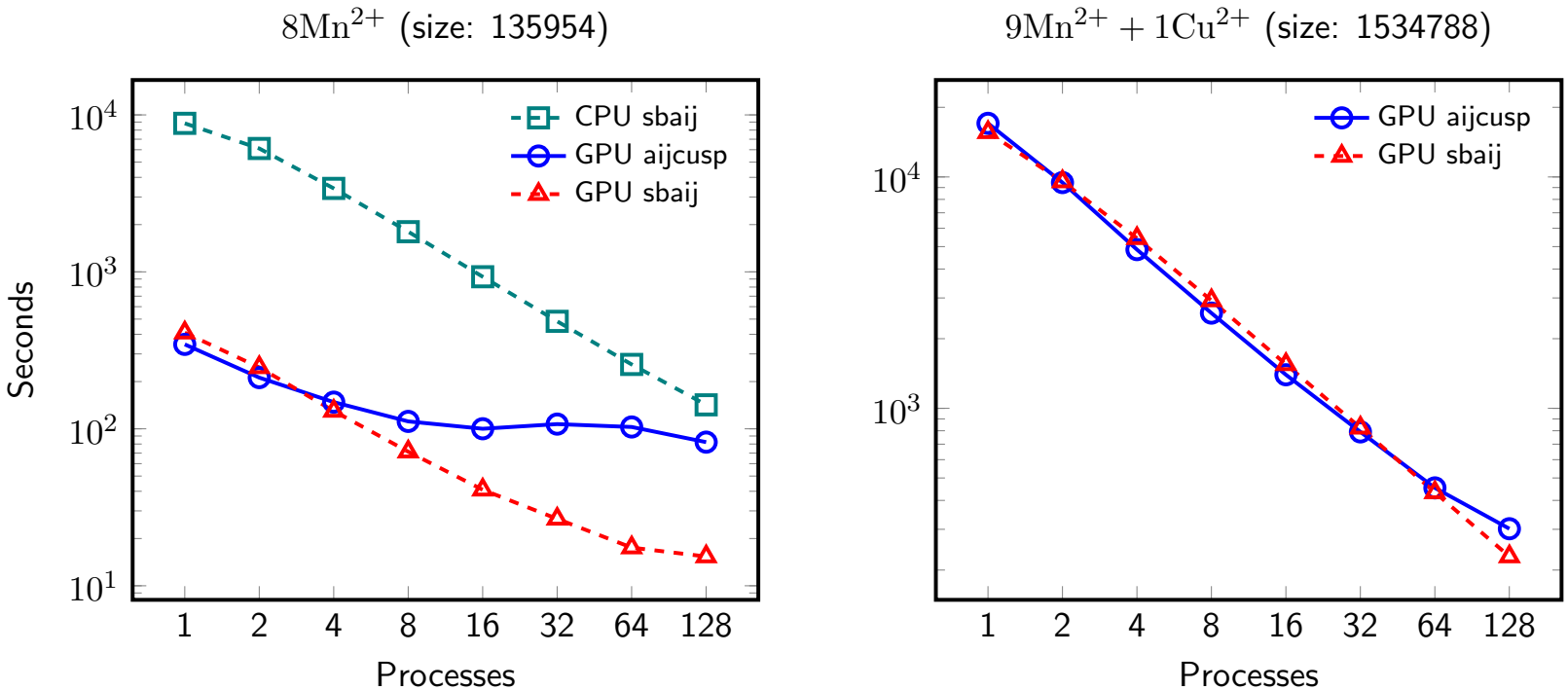


Molecular Magnetism: Computational Results

Computation of matrix coefficients becomes the bottleneck

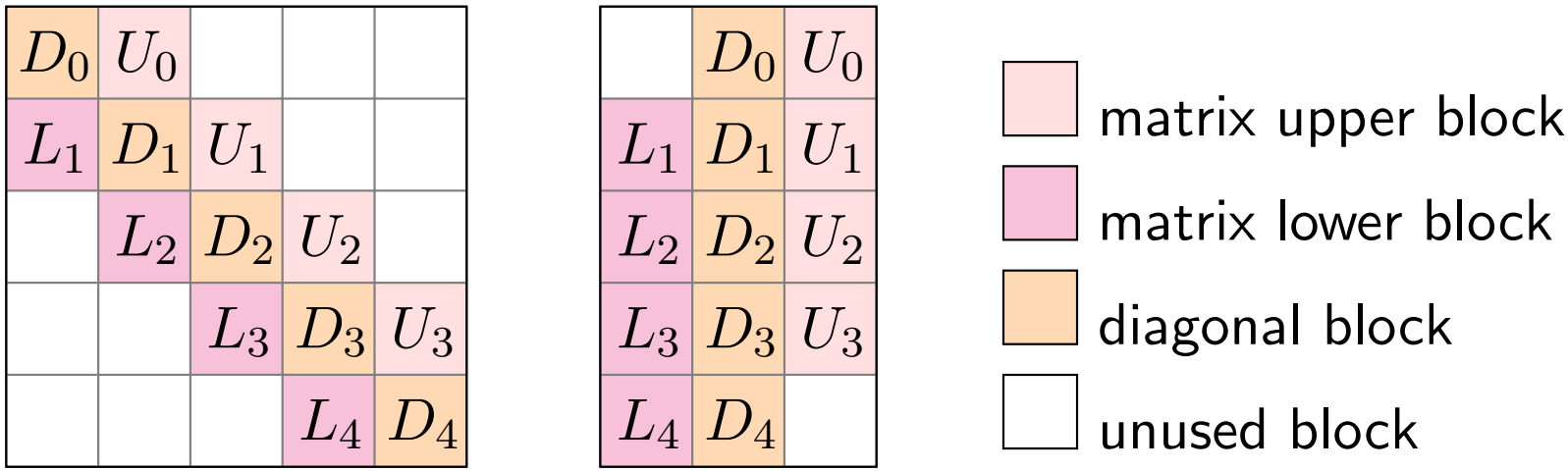
- GPU kernels required in user code to be efficient

Total problem solve time in double precision arithmetic:



A. Lamas Daviña, E. Ramos and J. E. Roman, "Optimized analysis of isotropic high-nuclearity clusters with GPU acceleration", *Comput. Phys. Commun.* (2016). DOI: 10.1016/j.cpc.2016.08.014

Block-Tridiagonal Matrices with Compact Storage



Methods we use to solve linear systems:

- Direct linear solvers
  - BCYCLIC [Hirshman et al. 2010]
  - SPIKE [Polizzi/Sameh 2006]
- Inexact shift-and-invert
  - Truncated SPIKE [Polizzi/Sameh 2006]

Test Cases

Plasma confined by magnetic toroidal field [Hirshman et al. 2010]

- 3D PDE where two coordinates are angular and one is radial
- Fourier transform results in dense block blocks

Transfer problem in stellar atmospheres [Ahues et al. 2002]

- Integral operator $T : X \to X$, $X = L^1 \in ([0, \tau^\star])$

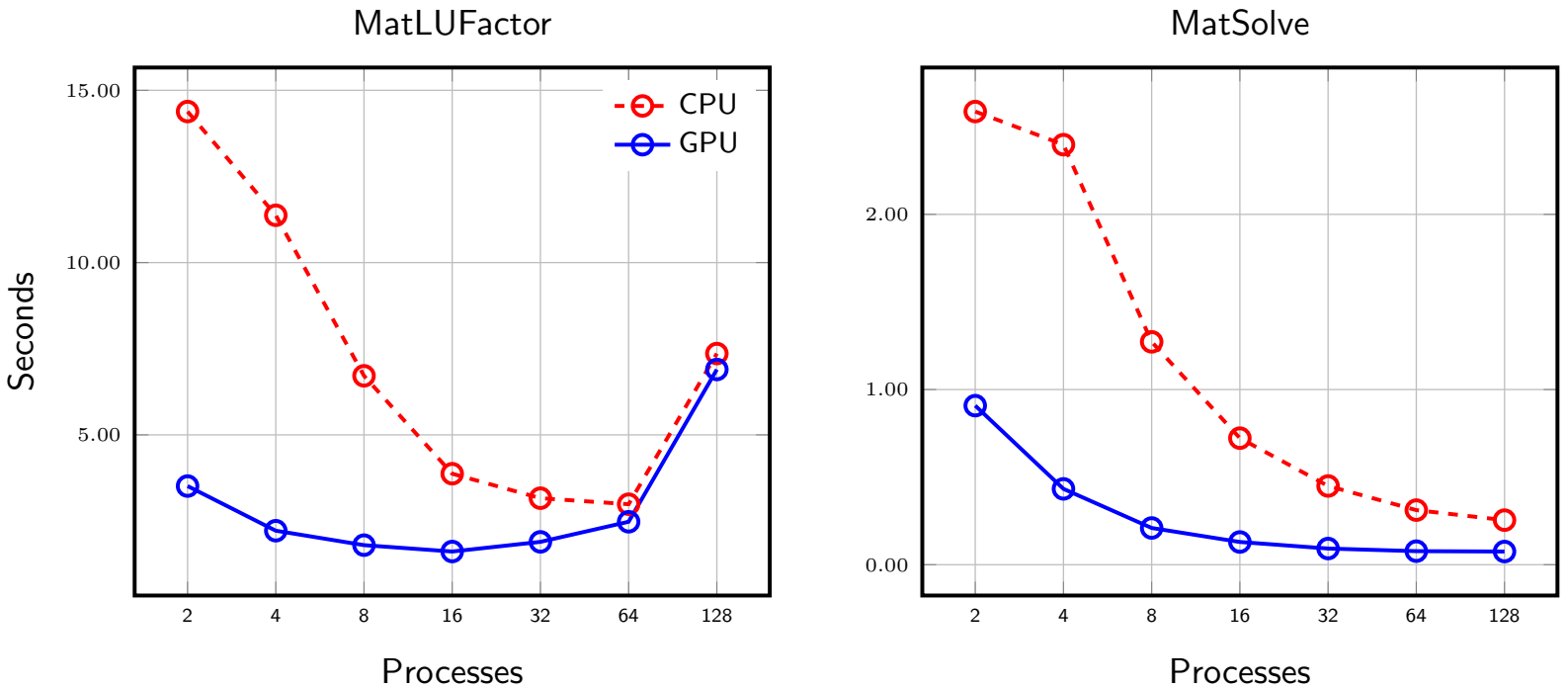
$$(T\varphi)(\tau) = \frac{\varpi}{2} \int_0^{\tau^\star} \int_1^\infty \frac{e^{-|\tau-\tau'|\mu}}{\mu} d\mu \varphi(\tau') d\tau, \quad \tau \in [0, \tau^\star]$$

  $\varpi \in [0, 1]$: albedo; $\tau^\star$: optical thickness of stellar atmosphere

- Discretize the eigenproblem $T\varphi = \lambda\varphi$, $\lambda \in \mathbb{C}$, $\varphi \neq 0 \in X$, on a finite-dimensional subspace X_n
- Results in a banded matrix (exponential decay)

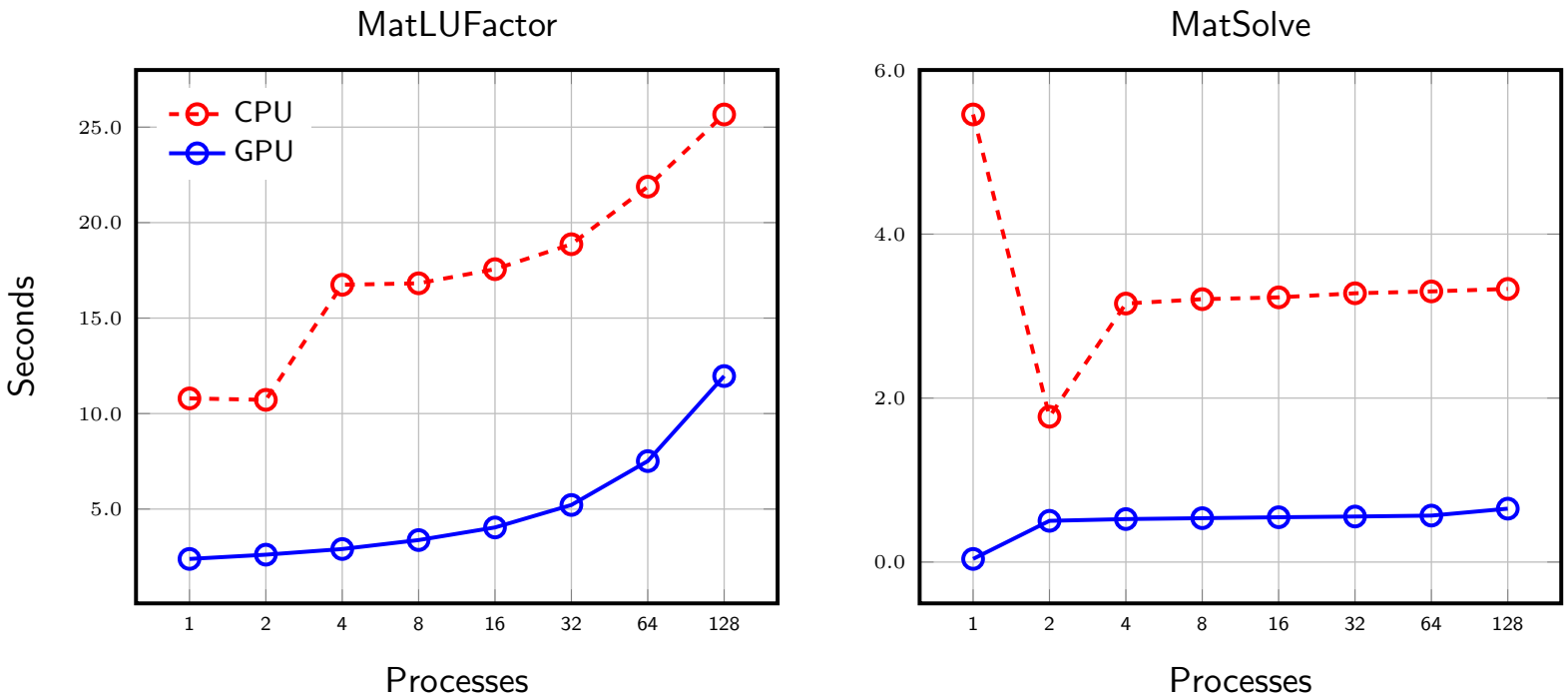
Block-Tridiagonal Matrices: Strong Scaling

Block size: $k = 1024$, Block rows: $l = m/k$, $m = 307200$



Block-Tridiagonal Matrices: Weak Scaling

Block size: $k = 1024$, Block rows: $l = m/k$, $m = procs \cdot 102400$



A. Lamas Daviña and J. E. Roman, "MPI-GPU parallelism in preconditioned Krylov solvers for block-tridiagonal matrices", in preparation (2016).

Concluding Remarks

GPU support in SLEPc:

- Can exploit GPU's trivially, with no code modification
- Usually a modest speed gain, 10x at most (wrt MKL)
- Often necessary to port application code to GPU also

Work in progress:

- Further optimizations for Krylov methods (BLAS-2)
- Extension for block eigensolvers (BLAS-3)
- Preconditioners and linear solvers still a research topic

Acknowledgement:

GOBIERNO DE ESPAÑA
MINISTERIO DE ECONOMÍA Y COMPETITIVIDAD