SLEPc: Scalable Library for Eigenvalue Problem Computations

Tutorial – version 3.6

Jose E. Roman

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

Celebrating 20 years of PETSc, Argonne – June, 2015

Outline

Eigenproblems

Large-scale eigenvalue problems are among the most demanding calculations in scientific computing

Example application areas:

- Dynamic structural analysis (e.g., civil engineering)
- Stability analysis (e.g., control engineering)
- Eigenfunction determination (e.g., electromagnetics)
- Bifurcation analysis (e.g., fluid dynamics)
- Information retrieval (e.g., latent semantic indexing)

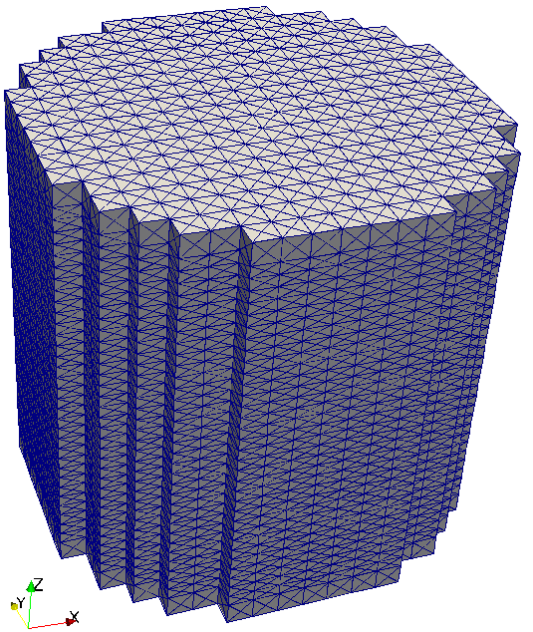
Use Case: Neutron Difusion Equation in Nuclear Eng.

Neutron power in nuclear reactor cores

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

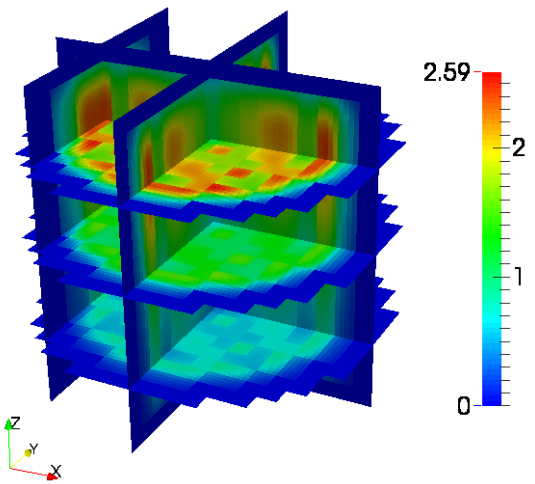
Lambda Modes Equation

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

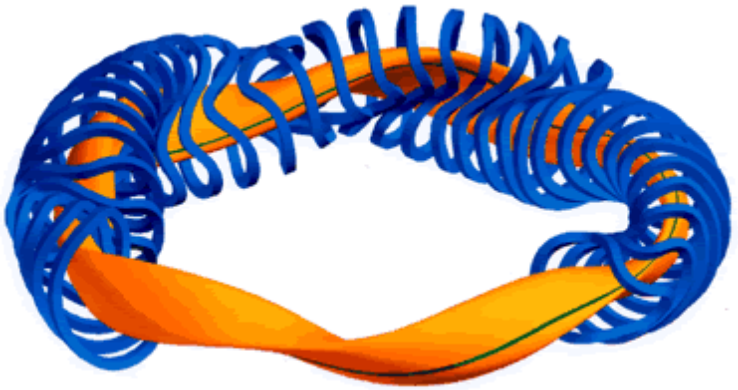


Current trends

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



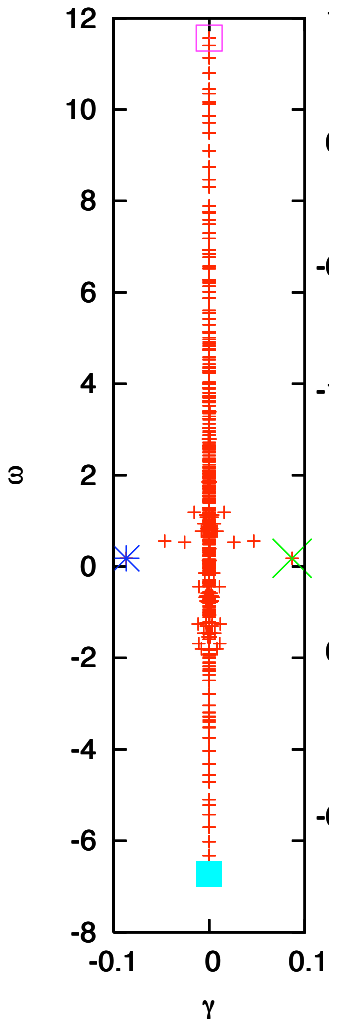
Use Case: Gyrokinetic Equations in Plasma Physics



IPP Max-Planck-Institut für Plasmaphysik

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
- Sizes ranging from a few millions to a billion
- Estimate optimal timestep (largest eigenvalue); track sub-dominant instabilities (rightmost evals)

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, E. Romero, A. Tomas

`http://slepc.upv.es`

Current version: 3.6 (released June 2015)

PETSc

Nonlinear Systems		
Line Search	Trust Region	Other

Time Steppers			
Euler	Backward Euler	Pseudo Time Step	Other

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

Preconditioners						
Additive Schwarz	Block Jacobi	Jacobi	ILU	ICC	LU	Other

Matrices					
Compressed Sparse Row	Block CSR	Symmetric Block CSR	Dense	CUSP	Other

Vectors	
Standard	CUSP

Index Sets			
Indices	Block	Stride	Other

SLEPc

Polynomial Eigensolver		
TOAR	Q-Arnoldi	Linear-ization

Nonlinear Eigensolver			
SLP	RII	N-Arnoldi	Interp.

SVD Solver			
Cross Product	Cyclic Matrix	Lanczos	Thick R. Lanczos

M. Function
Krylov

Linear Eigensolver					
Krylov-Schur	GD	JD	LOBPCG	CISS	Other

Spectral Transformation			
Shift	Shift-and-invert	Cayley	Preconditioner

BV	DS	RG	FN

Problem Classes

The user must choose the most appropriate solver for each problem class

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

This tutorial focuses on the linear eigenvalue problem (EPS)

EPS: Eigenvalue Problem Solver

Compute a few eigenpairs (x, λ) of

Standard Eigenproblem

$$Ax = \lambda x$$

Generalized Eigenproblem

$$Ax = \lambda Bx$$

where A, B can be real or complex, symmetric (Hermitian) or not

User can specify:

- Number of eigenpairs (`nev`), subspace dimension (`ncv`)
- Selected part of spectrum
- Tolerance, maximum number of iterations
- Advanced: extraction type, initial guess, constraints, balancing

Basic EPS Usage

```
EPS         eps;       /*  eigensolver context  */
Mat         A, B;      /*  matrices of Ax=kBx   */
Vec         xr, xi;    /*  eigenvector, x       */
PetscScalar kr, ki;    /*  eigenvalue, k        */

EPSCreate(PETSC_COMM_WORLD, &eps);
EPSSetOperators(eps, A, B);
EPSSetProblemType(eps, EPS_GNHEP);
EPSSetFromOptions(eps);
EPSSolve(eps);
EPSGetConverged(eps, &nconv);
for (i=0; i<nconv; i++) {
  EPSGetEigenpair(eps, i, &kr, &ki, xr, xi);
}
EPSDestroy(eps);
```

Problem Definition

`EPSSetOperators(EPS eps,Mat A,Mat B)`

Pass one or two matrices that define the problem $Ax = \lambda Bx$

- For a standard problem, set `B=NULL`
- Any PETSc matrix type, including *shell* matrices

`EPSSetProblemType(EPS eps,EPSProblemType type)`

To indicate the problem type (hint for the solver)

`EPS_HEP`	standard Hermitian problem, $A = A^*$, all λ_i real
`EPS_NHEP`	standard non-Hermitian problem
`EPS_GHEP`	generalized Hermitian problem, A, B symmetric (Hermitian), B positive (semi-)definite, all λ_i real
`EPS_GNHEP`	generalized non-Hermitian problem

Solution of the Eigenvalue Problem

There are n eigenvalues (counted with their multiplicities)

Partial eigensolution: nev solutions

$$\lambda_0, \lambda_1, \ldots, \lambda_{nev-1} \in \mathbb{C}$$
$$x_0, x_1, \ldots, x_{nev-1} \in \mathbb{C}^n$$

nev = number of eigenvalues / eigenvectors (eigenpairs)

Which eigenvalues must be computed?

1. Those 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

Available Eigensolvers

User code is independent of the selected solver

1. Single vector iteration: power iteration, inverse iteration, RQI
2. Subspace iteration with Rayleigh-Ritz projection and locking
3. Explicitly restarted Arnoldi and Lanczos
4. **Krylov-Schur**, including thick-restart Lanczos
5. Generalized Davidson, Jacobi-Davidson
6. Conjugate gradient methods: LOBPCG, RQCG
7. CISS, a contour-integral solver
8. External packages, and LAPACK for testing

. . . but some solvers are specific for a particular case:

- LOBPCG computes smallest λ_i of symmetric problems
- CISS allows computation of all λ_i within a region

Processing Command-Line Options

EPSSetFromOptions(EPS eps)

Looks in the command line for options related to EPS

For example, the following command line

```
$ ./ex1 -eps_hermitian
```

is equivalent to a call EPSSetProblemType(eps,EPS_HEP)

Other options have an associated function call

```
$ ./ex1 -eps_nev 6 -eps_tol 1e-8
```

EPSView(EPS eps,PetscViewer viewer)

Prints information about the object (equivalent to -eps_view)

Sample Output of -eps_view (edited)

```
EPS Object: 1 MPI processes
  type: krylovschur
    Krylov-Schur: 50% of basis vectors kept after restart
    Krylov-Schur: using the locking variant
  problem type: symmetric eigenvalue problem
  extraction type: Rayleigh-Ritz
  selected portion of the spectrum: largest eigenvalues in magnitude
  number of eigenvalues (nev): 1
  number of column vectors (ncv): 16
  maximum dimension of projected problem (mpd): 16
  maximum number of iterations: 100
  tolerance: 1e-08
BV Object: 1 MPI processes
  type: svec
  orthogonalization method: classical Gram-Schmidt
  orthogonalization refinement: if needed (eta: 0.7071)
DS Object: 1 MPI processes
  type: hep
  solving the problem with: Implicit QR method (_steqr)
ST Object: 1 MPI processes
  type: shift
  shift: 0
```

EPS: Run-Time Examples

```
$ ./ex5 -eps_type krylovschur -eps_nev 6 -eps_ncv 24

$ ./ex5 -eps_type arnoldi -eps_tol 1e-11 -eps_max_it 2000

$ ./ex1 -eps_type subspace -eps_hermitian -log_summary

$ ./ex1 -eps_type lobpcg -eps_smallest_real

$ ./ex5 -eps_type gd -eps_gd_blocksize 2

$ ./ex9 -eps_type arpack -eps_largest_real
```

Viewing the Solution

Eigenvalues and eigenvectors can be viewed with `PetscViewers`

- Text output, e.g. M-file

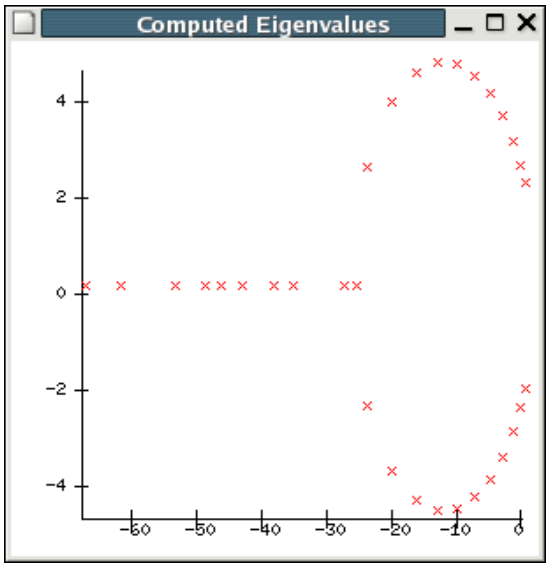
  `-eps_view_values :myeig.m:ascii_matlab`

- Plotting eigenvalues

  `-eps_view_values draw`

- Eigenvectors, e.g. to binary file

  `-eps_view_vectors binary:evec.bin`



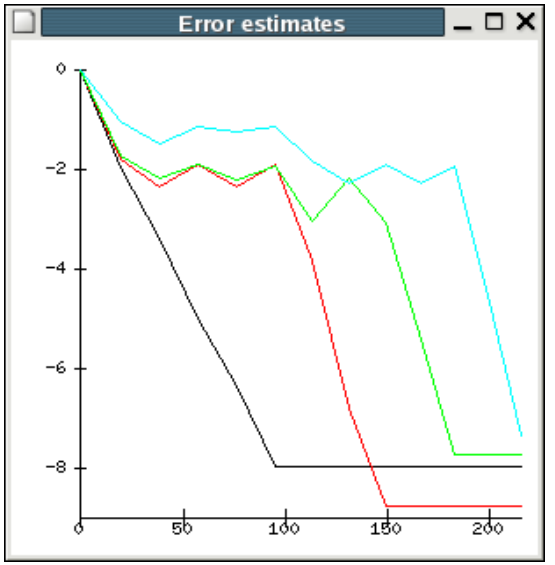

```
$ ./ex1 -eps_error_relative ::ascii_info_detail
   ---------------------- --------------------
              k              ||Ax-kx||/||kx||
   ---------------------- --------------------
         3.999326             1.26221e-09
         3.997304             3.82982e-10
         3.993936             2.76971e-09
         3.989224             4.94104e-10
         3.983171             6.19307e-10
         3.975781              5.9628e-10
   ---------------------- --------------------
```

Can also compute and display residual errors

Monitoring Convergence

Graphical monitors `-eps_monitor_lg`
`-eps_monitor_lg_all`

Textual monitors `-eps_monitor`
`-eps_monitor_all`
`-eps_monitor_conv`




```
 1 EPS nconv=0 first unconverged value (error) -0.0695109+2.10989i (2.38956768e-01)
 2 EPS nconv=0 first unconverged value (error) -0.0231046+2.14902i (1.09212525e-01)
 3 EPS nconv=0 first unconverged value (error) -0.000633399+2.14178i (2.67086904e-02)
 4 EPS nconv=0 first unconverged value (error) 9.89074e-05+2.13924i (6.62097793e-03)
 5 EPS nconv=0 first unconverged value (error) -0.000149404+2.13976i (1.53444214e-02)
 6 EPS nconv=0 first unconverged value (error) 0.000183676+2.13939i (2.85521004e-03)
 7 EPS nconv=0 first unconverged value (error) 0.000192479+2.13938i (9.97563492e-04)
 8 EPS nconv=0 first unconverged value (error) 0.000192534+2.13938i (1.77259863e-04)
 9 EPS nconv=0 first unconverged value (error) 0.000192557+2.13938i (2.82539990e-05)
10 EPS nconv=0 first unconverged value (error) 0.000192559+2.13938i (2.51440008e-06)
11 EPS nconv=2 first unconverged value (error) -0.671923+2.52712i (8.92724972e-05)
```

Spectral Transformation

Shift-and-invert is used to compute interior eigenvalues

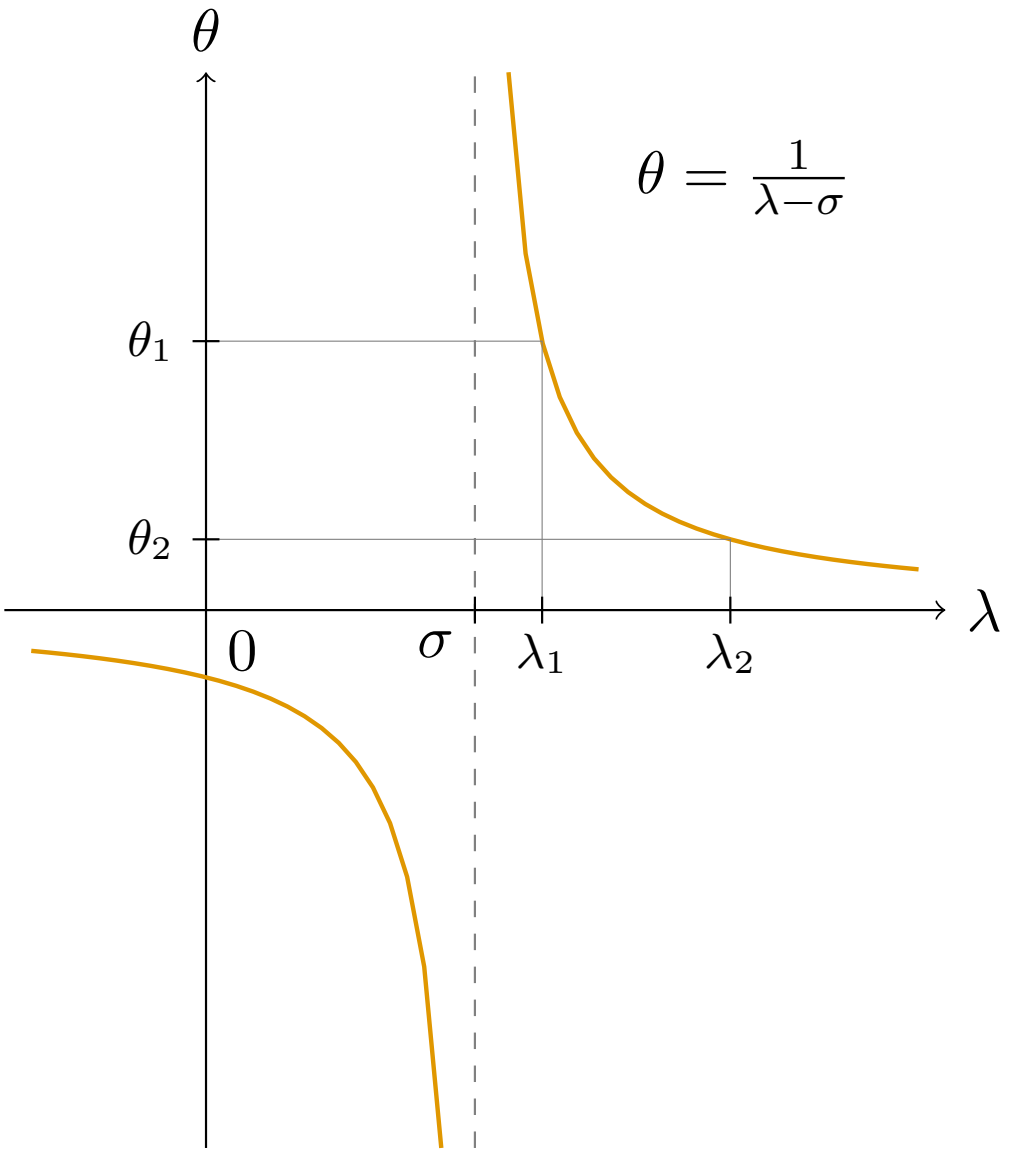
$$Ax = \lambda Bx \quad \Longrightarrow \quad (A - \sigma B)^{-1} Bx = \theta x$$

- Trivial mapping of eigenvalues: $\theta = (\lambda - \sigma)^{-1}$
- Eigenvectors are not modified
- Very fast convergence close to σ

Things to consider:

- Implicit inverse $(A - \sigma B)^{-1}$ via linear solves
- Direct linear solver for robustness
- Less effective for eigenvalues far away from σ
- Cheaper alternative: preconditioned eigensolvers (J-D)

Illustration of Shift-and-Invert



Spectral Transformation in SLEPc

An ST object is always associated to any EPS object

- The user need not create the ST object, `EPSGetST` to get it
- Internally, the eigensolver works with the operator T
- At the end, eigenvalues are transformed back automatically

ST	Standard problem	Generalized problem
`shift`	$A - \sigma I$	$B^{-1}A - \sigma I$
`sinvert`	$(A - \sigma I)^{-1}$	$(A - \sigma B)^{-1}B$
`cayley`	$(A - \sigma I)^{-1}(A + \tau I)$	$(A - \sigma B)^{-1}(A + \tau B)$
`precond`	$K^{-1} \approx (A - \sigma I)^{-1}$	$K^{-1} \approx (A - \sigma B)^{-1}$

A `KSP` object is handled internally for the linear solves

ST: Command-Line Examples

```
$ ./ex1 -st_type sinvert -eps_target 2.1
        -st_ksp_type preonly -st_pc_type lu
        -st_pc_factor_mat_solver_package mumps

$ ./ex1 -st_type sinvert -eps_target 2.1
        -st_ksp_type bcgs -st_ksp_rtol 1e-9
        -st_pc_type sor -st_pc_sor_omega 1.3

$ ./ex5 -eps_type gd -eps_target 0.8 -eps_harmonic
        -st_pc_type asm -st_sub_pc_factor_levels 2

$ ./ex5 -eps_type jd -st_ksp_type gmres
        -st_pc_type jacobi -st_ksp_max_it 10

$ ./ex1 -eps_interval 0.4,0.8 -st_type sinvert
        -st_ksp_type preonly -st_pc_type cholesky
```

Options for Subspace Generation

Initial Subspace

- Provide an initial trial subspace with `EPSSetInitialSpace`, e.g. from a previous computation
- Krylov solvers only support a single vector

Deflation Subspace

- Provide a deflation space with `EPSAttachDeflationSpace`
- The eigensolver operates in the restriction to the orthogonal complement
- Useful for constrained eigenproblems or problems with a known nullspace

Extraction / Balancing

Harmonic extraction

In some cases, convergence of the eigensolver may be very slow
$\rightarrow$ try to extract better approximations from the available subspace

- Compute harmonic Ritz values instead of Ritz values
- To compute interior eigenvalues (alternative to the spectral transformation)
- Particularly useful in preconditioned eigensolvers (JD, GD)

```
$ ./ex5 -m 45 -eps_harmonic -eps_target 0.8 -eps_ncv 60
```

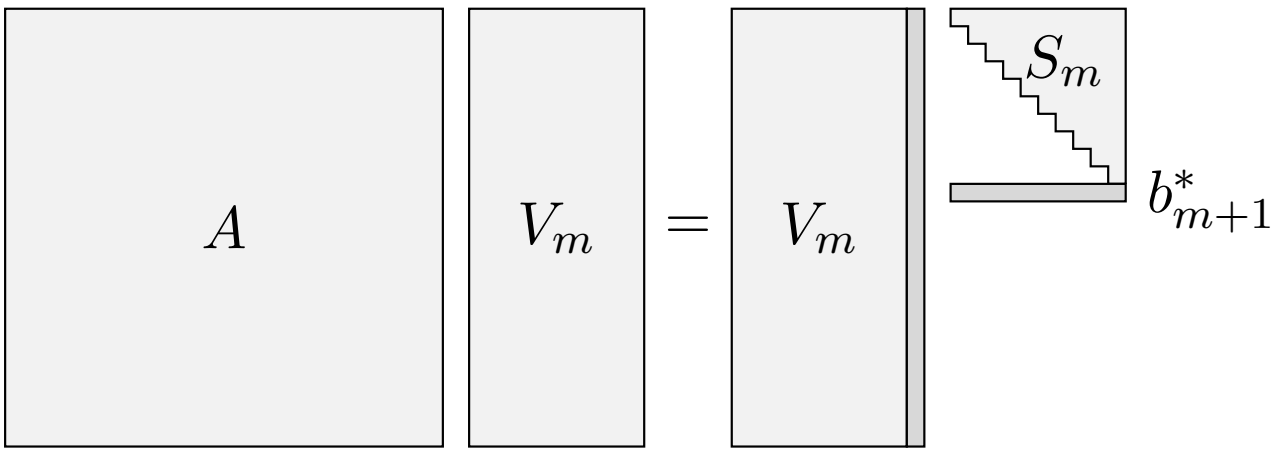
Balancing

- Possible bad accuracy if $\|A\|_2$ large (non-Hermitian problems)
- Balancing implicitly performs a diagonal similarity DAD^{-1}

Computation of Many Eigenpairs

By default, a subspace of dimension $2 \cdot nev$ is used...
For large nev, this is not appropriate

- Excessive storage and inefficient computation



Strategy: compute eigenvalues in chunks - restrict the dimension of the projected problem

```
$ ex1 -eps_nev 5000 -eps_mpd 600
```

SLEPc Highlights

- Growing number of eigensolvers
- Seamlessly integrated spectral transformation
- Easy programming with PETSc's object-oriented style
- Data-structure neutral implementation
- Run-time flexibility, giving full control over the solution process
- Portability to a wide range of parallel platforms
- Usable from code written in C, C++ and Fortran
- Extensive documentation

More Information

Homepage:

`http://slepc.upv.es`

Hands-on Exercises:

`http://slepc.upv.es/handson`

Contact email:

`slepc-maint@upv.es`