Métodos Iterativos para Valores Propios con SLEPc

Vicente Hernández, José Román, Eloy Romero, Andrés Tomás

Universidad Politécnica de Valencia

UNIVERSIDAD POLITECNICA DE VALENCIA

Grid y Computación de Altas Prestaciones

GRyCAP

Contenido

1. Introducción
2. Portable, Extensible Toolkit for Scientific Computation
   - Vectores
   - Matrices
   - Sistemas lineales
   - Precondicionadores
3. Scalable Library for Eigenvalue Problem Computations
   - Valores y vectores propios
   - Transformaciones espectrales
   - SVD
   - Desarrollos actuales

Problema de valores y vectores propios

Problema estándar

$$Ax = \lambda x$$

Donde $A \in \mathbb{C}^{n \times n}$

n soluciones

$$\lambda_1, \lambda_2, \ldots, \lambda_n \in \mathbb{C}$$

$$x_1, x_2, \ldots, x_n \in \mathbb{C}^n$$

Si A es Hermitiana (simétrica), entonces $\lambda_i \in \mathbb{R}$

Resolución del problema de valores y vectores propios

Métodos basados en transformaciones de semejanza

- Solución completa
- Matrices densas

Métodos iterativos

- Solución parcial
- Matrices dispersas

Problema de valores y vectores propios generalizado

Problema generalizado

$$Ax = \lambda Bx$$

Con $A, B \in \mathbb{C}^{n \times n}$

Transformación espectral

$$Ax = \lambda Bx \longrightarrow Tx = \theta x$$

Cambia los valores propios pero no los vectores propios
Utilizadas para:

- Cálculo de valores propios en el interior del espectro
- Resolver problemas generalizados con métodos estándar

Inconveniente: suele ser necesario resolver un sistema lineal en cada iteración

Aplicaciones del problema de valores y vectores propios

Ejemplos:

- Análisis dinámico de estructuras (ingeniería civil)
- Análisis de estabilidad (problemas de control)
- Cálculo de funciones propias (electromagnetismo)
- Búsqueda de información (PageRank de Google)

Calcular unos unos pocos valores propios que cumplan:

- ser dominantes (mayor magnitud)
- tener la parte real más pequeña (o grande)
- estar en un región del plano complejo

Descomposición en valores y vectores singulares (SVD)

SVD

$$A = U\Sigma V^*$$

Donde $A \in \mathbb{R}^{n\times m}$ o $A \in \mathbb{C}^{n\times m}$
U y V matrices unitarias $m \times m$ y $n \times n$
Σ es una matriz diagonal $m \times n$ con $\Sigma_{ii} = \sigma_i$ y $\sigma_i \in \mathbb{R}$

- Valores propios de AA^*, A^*A o $H(A) = \begin{bmatrix} 0^{m\times m} & A \\ A^* & 0^{n\times n} \end{bmatrix}$
- Bidiagonalización

Descomposición en valores y vectores singulares (SVD)

SVD parcial de rango t

$$A_t = U_t \Sigma_t V_t^* = \sum_{i=1}^{t} u_i \sigma_i v_i^*$$

Aplicaciones de la SVD

- Aproximación de una matriz por otra de menor rango
  - Filtrado de ruido
  - Reconstrucción de imágenes
- Pseudo-inversa
- Cálculo del espacio nulo
- Cálculo de la 2-norma

Herramientas

- Elementos de álgebra lineal: vectores y matrices dispersas
- Métodos iterativos para sistemas lineales (directos e iterativos)
- Ejecución en paralelo (transparente al usuario)
- Independencia de plataforma
- Herramientas de depuración y medida de prestaciones

- Métodos para valores propios (y SVD)
- Transformaciones espectrales

Herramientas

Proporcionadas por PETSc

- Elementos de álgebra lineal: vectores y matrices dispersas
- Métodos iterativos para sistemas lineales (directos e iterativos)
- Ejecución en paralelo (transparente al usuario)
- Independencia de plataforma
- Herramientas de depuración y medida de prestaciones

Proporcionadas por SLEPc

- Métodos para valores propios (y SVD)
- Transformaciones espectrales

PETSc: Portable, Extensible Toolkit for Scientific Computation

- Orientada a la resolución de ecuaciones en derivadas parciales
- Portable a cualquier máquina paralela que disponga de MPI
- Programada en C
- Utilizable desde C, C++, FORTRAN y Python
- Técnicas de programación orientada a objetos (encapsulación y polimorfismo)
- Ampliamente documentada y con muchos ejemplos
- Código abierto desarrollado por el Argonne National Lab desde 1991

`http://www.mcs.anl.gov/petsc`

Versión actual: 3.1.0 (Marzo 2010)

SLEPc: Scalable Library for Eigenvalue Problem Computations

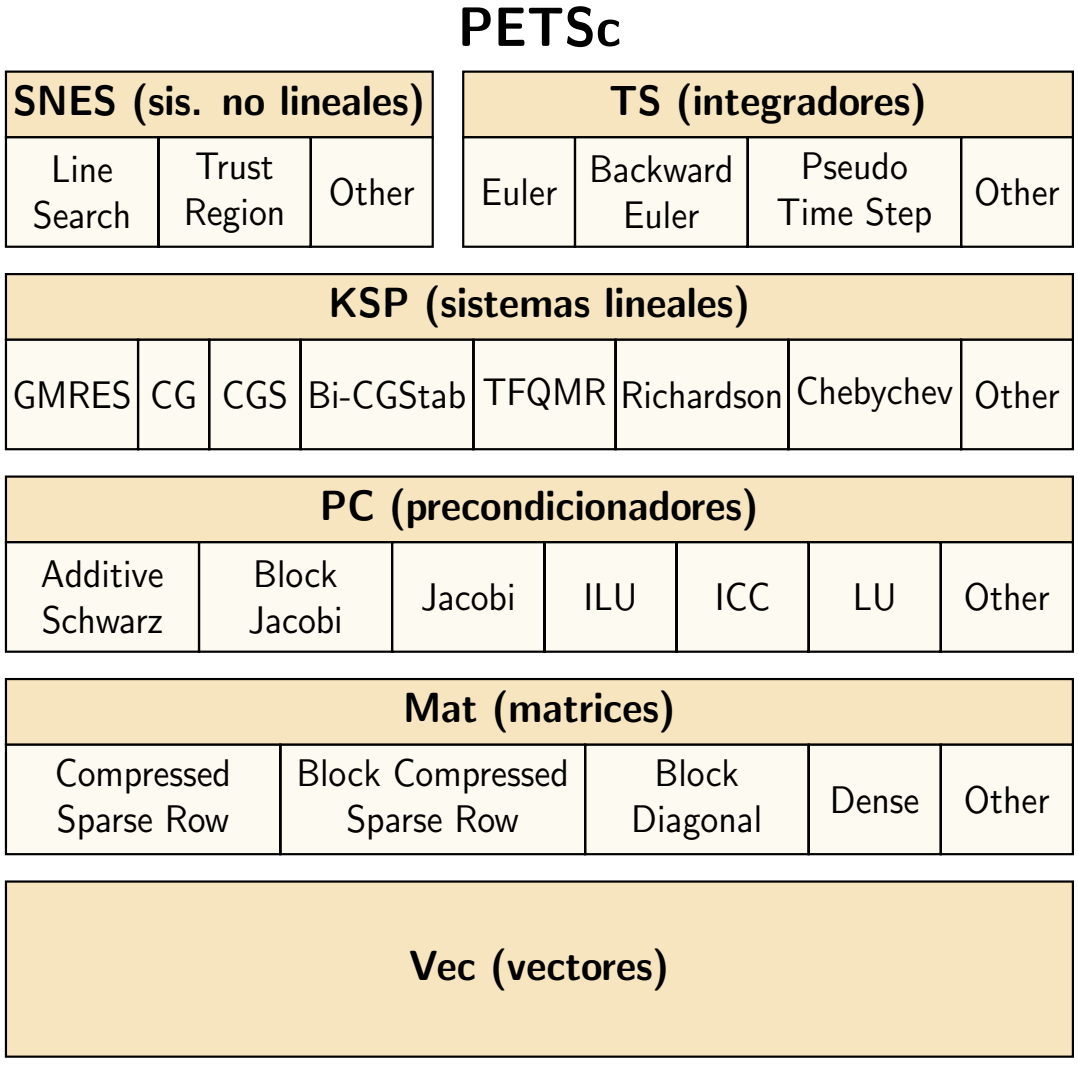
Extensión de PETSc para la resolución paralela de problemas de valores y vectores propios para matrices dispersas de gran tamaño

- Para problemas de valores propios simétricos o generales
- Con aritmética real o compleja
- Soporte para transformaciones espectrales y SVD parcial
- Dispone de distintos métodos propios e interfaz a otras librerías (ej. ARPACK, PRIMME)
- Código abierto con licencia GPL

`http://www.grycap.upv.es/slepc`

Versión actual: 3.0.0 (Febrero 2009)

Interfaz orientado a objetos de PETSc y SLEPc



Interfaz orientado a objetos de PETSc y SLEPc

PETSc

SNES (sis. no lineales)		
Line Search	Trust Region	Other

TS (integradores)			
Euler	Backward Euler	Pseudo Time Step	Other

KSP (sistemas lineales)							
GMRES	CG	CGS	Bi-CGStab	TFQMR	Richardson	Chebychev	Other

PC (precondicionadores)						
Additive Schwarz	Block Jacobi	Jacobi	ILU	ICC	LU	Other

Mat (matrices)				
Compressed Sparse Row	Block Compressed Sparse Row	Block Diagonal	Dense	Other

Vec (vectores)

SLEPc

SVD (valores singulares)			
Cross Product	Cyclic Matrix	Lanczos	Thick Res. Lanczos

EPS (valores propios)			
Krylov-Schur	Arnoldi	Lanczos	Other

ST (transformación espectral)			
Shift	Shift-and-invert	Cayley	Fold

Contenido

1. Introducción
2. Portable, Extensible Toolkit for Scientific Computation
   - Vectores
   - Matrices
   - Sistemas lineales
   - Precondicionadores
3. Scalable Library for Eigenvalue Problem Computations
   - Valores y vectores propios
   - Transformaciones espectrales
   - SVD
   - Desarrollos actuales

Vectores

Distribución paralela: cada procesador tiene un bloque contiguo de elementos

- Operaciones básicas
  - `VecCreate(MPI_Comm, *Vec)`
  - `VecSetSizes(Vec, n, N)`
  - `VecGetOwnershipRange(Vec, *low, *high)`
  - `VecView(Vec, viewer)`
  - `VecLoad(viewer, *Vec)`
  - `VecDestroy(Vec)`
- Operaciones con elementos
  - `VecSetValues(Vec, ni, ix[], y[])`
  - `VecGetValues(Vec, ni, ix[], y[])`

Operaciones tipo BLAS

Función	Operación
`VecAXPY(Vec y, alpha, Vec x)`	$y = y + \alpha x$
`VecWAYPX(Vec w, alpha, Vec x, Vec y)`	$w = y + \alpha x$
`VecDot(Vec y, Vec x, *r)`	$r = y^* x$
`VecNorm(Vec x, NORM_2, *r)`	$r = \|x\|_2$
`VecScale(Vec x, alpha)`	$x = \alpha x$
`VecCopy(Vec y, Vec x)`	$x = y$
`VecMax(Vec x, *idx, *r)`	$r = \max(x_i)$

Matrices

Distribución paralela: cada procesador tiene un bloque contiguo de filas completas

- Operaciones básicas
  - `MatCreate(MPI_Comm, *Mat)`
  - `MatSetSizes(Mat, m, n, M, N)`
  - `MatGetOwnershipRange(Mat, *low, *high)`
  - `MatView(Mat, viewer)`
  - `MatLoad(viewer, *Mat)`
  - `MatDestroy(Mat)`
- Operaciones con elementos
  - `MatSetValues(Mat, m, idxm[], n, idxn[], v[])`
  - `MatGetValues(Mat, m, idxm[], n, idxn[], v[])`

Matrices

Producto matrix-vector (BLAS 2)

`MatMult(Mat A, Vec x, Vec y)` $y = Ax$

Tipos de matrices

`MatSetType(Mat, type)`

- Dispersas: AIJ, AIJ a bloques, AIJ a bloques simétrica
- Densas
- Producto matriz vector definido por el usuario (Shell):
  `MatShellSetOperation(Mat, MATOP_MULT, func)`

Otras operaciones

`MatMultTranspose(Mat A, Vec x, Vec y)`	$y = A^*x$
`MatAXPY(Mat B, alpha, Mat A)`	$B = B + \alpha A$
`MatNorm(Mat A, NORM_FROBENIUS, *r)`	$r = \|A\|_F$
`MatScale(Mat A, alpha)`	$A = \alpha A$
`MatShift(Mat A, alpha)`	$A = A + \alpha I$

Métodos iterativos para sistemas lineales

Operaciones

- `KSPCreate(MPI_Comm, *KSP)`
- `KSPSetOperators(KSP, Mat A, Mat M)`
- `KSPSetType(KSP, type)`
- `KSPSetTolerances(KSP, rtol, abstol, dtol, maxits)`
- `KSPSetFromOptions(KSP)`
- `KSPView(KSP, viewer)`
- `KSPSolve(KSP, Vec b, Vec x)`
- `KSPDestroy(KSP)`

Métodos iterativos para sistemas lineales

- Richardson
- Chebychev
- conjugate gradients
- GMRES
- Bi-CG-stab
- transpose free QMR
- conjugate residuals
- conjugate gradient squared
- bi-conjugate gradient
- MINRES
- flexible GMRES
- LSQR
- SYMMLQ
- LGMRES
- Conjugate gradient on the normal equations

Precondicionadores

Todo objeto KSP contiene automáticamente un objeto PC

- `KSPGetPC(KSP, *PC)`

Precondicionadores paralelos

- Internos: Jacobi, block Jacobi, point block Jacobi, SOR, additive Schwarz
- Externos: BlockSolve95, Parasails, SPAI 3.0, Prometheus, BoomerAMG, ML
- Métodos directos: MUMPS, SPOOLES, SuperLU_Dist, DSCPACK, XYTlib

Contenido

1. Introducción
2. Portable, Extensible Toolkit for Scientific Computation
   - Vectores
   - Matrices
   - Sistemas lineales
   - Precondicionadores
3. Scalable Library for Eigenvalue Problem Computations
   - Valores y vectores propios
   - Transformaciones espectrales
   - SVD
   - Desarrollos actuales

Métodos iterativos para valores y vectores propios

Operaciones

- `EPSCreate(MPI_Comm, *EPS)`
- `EPSSetOperators(EPS, Mat A, Mat B)`
- `EPSSetProblemType(EPS, type)`
- `EPSSetWhichEigenpairs(EPS, which)`
- `EPSSetDimensions(EPS, nev, ncv, mpd)`
- `EPSSetTolerances(EPS, tol, maxits)`
- `EPSSetType(EPS, type)`
- `EPSSetFromOptions(EPS)`
- `EPSSolve(EPS)`
- `EPSGetEigenpair(EPS, i, *er, *ei, Vec Vr, Vec Vi)`
- `EPSDestroy(EPS)`

Métodos disponibles en SLEPc

- Potencia / RQI
- Subespacio
- Arnoldi / Lanczos con reinicio explícito
- Krylov-Schur (Lanczos con reinicio grueso)
- Librerías externas: ARPACK, PRIMME, BLOPEX, BLZPACK, TRLAN

Krylov-Schur

- Equivalente al reinicio implícito (ARPACK) con los valores de Ritz como dessplazamientos
- Pero no utiliza la iteración QR (más sencillo y más estable)
- Eficiente también para problemas Hermitianos

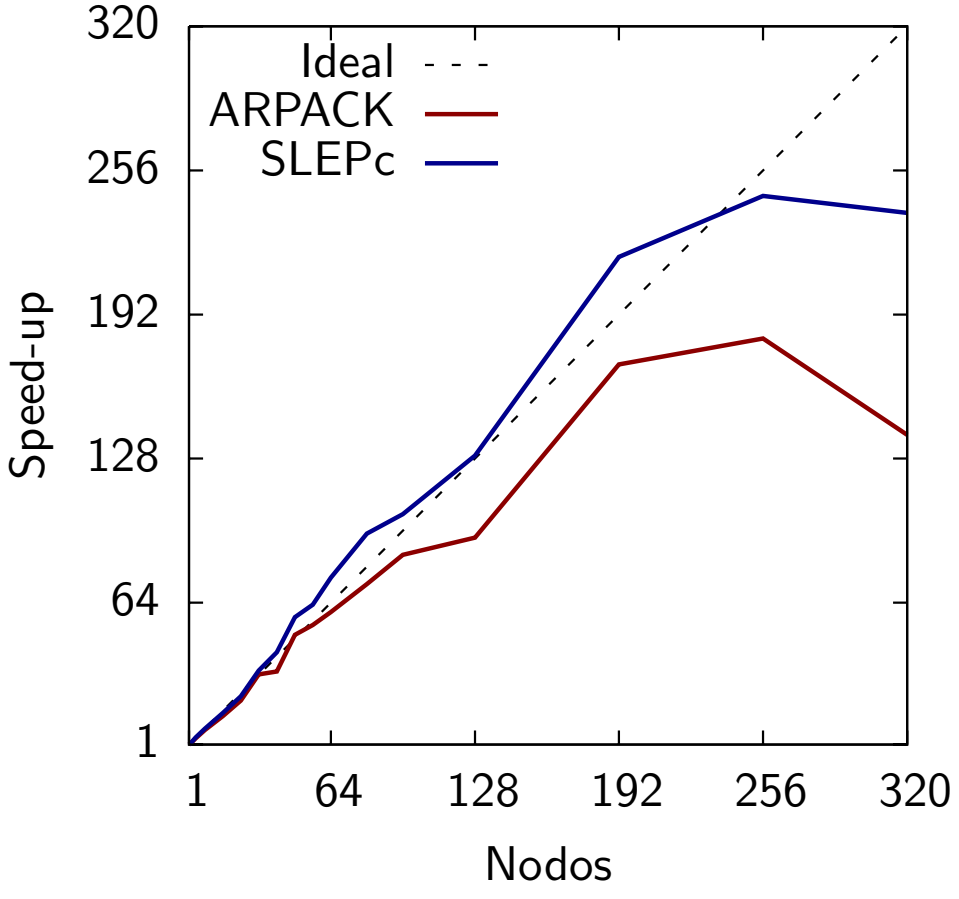
Prestaciones paralelas en MareNostrum

Speed-up

$$S_p = \frac{T_s}{T_p}$$

Matriz PRE2

- Universidad de Florida
- Dimensión 659,033
- 5,834,044 elem. no nulos
- 10 valores propios con una base de 30 vectores
- 1 procesador por nodo



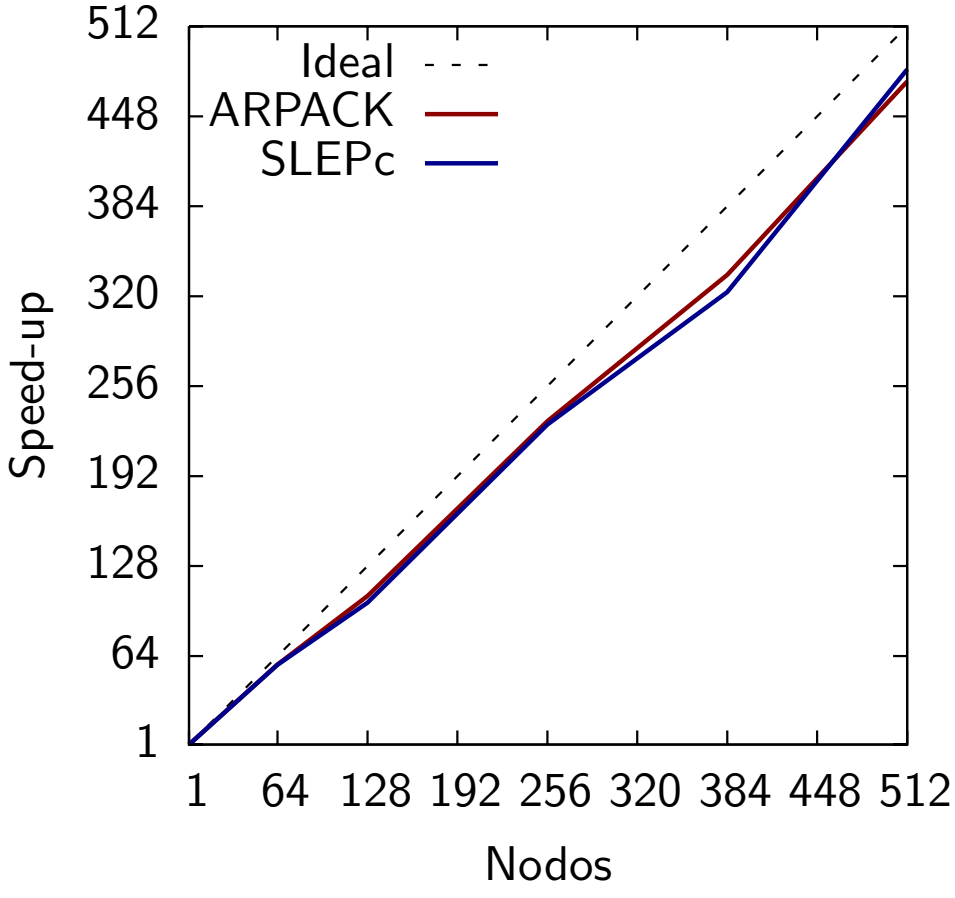
Prestaciones paralelas en MareNostrum

Speed-up escalado

$$S = \frac{\frac{T_s \times p}{I_s}}{\frac{T_p}{I_p}}$$

Matriz tridiagonal

- Dimensión 100,000$\times p$
- Elementos pseudoaleatorios entre 0 y 1
- 10 valores propios con una base de 30 vectores
- 1 procesador por nodo



Transformaciones espectrales

Todo objeto EPS contiene automáticamente un objeto ST

- `EPSGetST(EPS, *ST)`
- `STSetShift(ST, sigma)`

Todo objeto ST contiene automáticamente un objeto KSP

- `STGetKSP(ST, *KSP)`

- Desplazamiento $B^{-1}A + \sigma I$
- Desplazamiento e inversión $(A - \sigma B)^{-1}B$
- Cayley $(A - \sigma B)^{-1}(A + \tau B)$
- Plegado $(B^{-1}A + \sigma I)^2$

Programa de ejemplo

```
EPS          eps;       /*  metodo              */
Mat          A, B;      /*  matrices Ax=kBx     */
Vec          xr, xi;    /*  vector propio, x    */
PetscScalar  kr, ki;    /*  valor propio, 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);
```

Ejemplos de ejecución

```
% program -eps_view -eps_monitor

% program -eps_type krylovschur -eps_nev 6 -eps_ncv 24

% program -eps_type arnoldi -eps_tol 1e-8 -eps_max_it 2000

% program -eps_type subspace -eps_hermitian -log_summary

% program -eps_type primme -eps_smallest_real

% program -eps_type arpack -eps_tol 1e-6
          -st_type sinvert -st_shift 1
          -st_ksp_type cgs -st_ksp_rtol 1e-8
          -st_pc_type sor  -st_pc_sor_omega 1.3
```

SVD

Operaciones

- `SVDCreate(MPI_Comm, *SVD)`
- `SVDSetOperators(SVD, Mat A)`
- `SVDSetDimensions(SVD, nev, ncv, mpd)`
- `SVDSetTolerances(SVD, tol, maxits)`
- `SVDSetType(SVD, type)`
- `SVDSetFromOptions(SVD)`
- `SVDSolve(SVD)`
- `SVDGetSingularTriplet(SVD, i, *s, Vec V, Vec U)`
- `SVDDestroy(SVD)`

Métodos

- Producto cruzado
- Matriz cíclica
- Golub-Kahan-Lanczos con reinicio explícito / grueso

Desarrollos actuales

Próxima versión

- Método de Jacobi-Davidson
- Problemas cuadráticos (QEP)

$$(\lambda^2 M + \lambda C + K)x = 0$$

Trabajos futuros

- GPU
- Deflación

Resumen

- Métodos iterativos para problemas de valores y vectores propios (SVD) dispersos
- Soporte integrado para transformaciones espectrales
- Programación sencilla orientada a objetos con C, C++, Fortran y Python
- Gran flexibilidad en tiempo de ejecución
- Portable a casi cualquier máquina paralela
- Amplia documentación
- Licencia GPL

`http://www.grycap.upv.es/slepc`