Hands-on practice

《How to connect a PC cluster which is used during this innovation camp》

login_name: guest**
host_name: gauss.issp.u-tokyo.ac.jp

login_name is announced when you sent a PUBLIC ssh-key.
If you didn't receive your *login_name*, please contact us.

Linux or mac:
`ssh -l` *`login_name`* `-i` *`identity_file host_name`*
or
`ssh -i` *`identity_file login_name@host_name`*
(example: `ssh -i private_key.ppk guest99@gauss.issp.u-tokyo.ac.jp`)

Windows:
You can use Cygwin, Windows Subsystem for Linux or some terminals (such as Putty, Tera Term, Poderosa, and so on).

System

* Login node : gauss
* Computational nodes : gauss01 - gauss13 (13 nodes)
  * CPU : Xeon E5-2680v4, 28 cores/node
  * RAM : 64 GB/node
  * Do not login directly the computational nodes.
    Use the queuing system.
* Sample of batch job script : /home/public/sample.sh

Programs

* Binary : /home/public/bin/
* Source : /home/public/program/

samples for the practice

/home/lctr/iccms2/

```
|-- MateriApps_review_template.zip
`-- practice
    |-- doc_how_to_use_each_program_code.zip
    |-- eigenkernel
    |   |-- Makefile
    |   |-- Makefile.inc
    |   `-- run.sh
    |-- k-ep
    |   |-- Makefile.inc
    |   `-- run.sh
    `-- komega
```

These samples also have been uploaded in https://github.com/issp-center-dev/ICCMS/tree/master/2018/2018-10-02/fukuda

EigenKernel

(https://github.com/eigenkernel/eigenkernel/tree/eigenkernel_dev)

- generalized eigenvalue problems

$$A\vec{y}_k = \lambda_k B\vec{y}_k$$

$A, B : M \times M$ real-symmetric matrices
(*B* is positive definite.)

$\{\lambda_k\}$: Eigenvalues

$\{\vec{y}_k\}$: Eigenvectors

input

$$AY = BY\Lambda$$

output

$$\Lambda \equiv \mathrm{diag}(\lambda_1, \lambda_2, \dots)$$

$$Y \equiv (\vec{y}_1, \vec{y}_2, \dots)$$

- standard eigenvalue problem

$$A'Z = Z\Lambda$$

$$B = U^T U$$

$$A' \equiv U^{-T} A U^{-1}$$: real symmetric

$$Y = U^{-1} Z$$

```
./eigenkernel_app -s general_scalapack A.mtx B.mtx
```

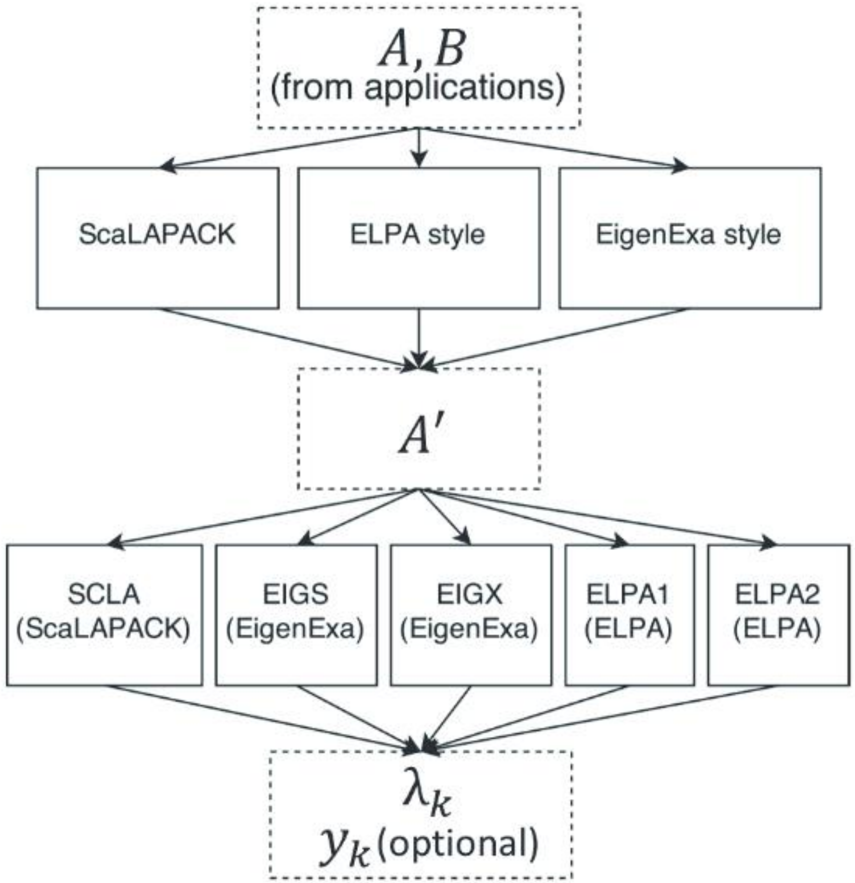

Fig. 3 Workflow of the hybrid GEP solver.

sample code : `src/main.f90`

k-ep

(Solution of the *k*-th eigenvalue problem)

(https://github.com/lee-djl/k-ep)

We assume that the eigenvalue problem-specific target index *k* satisfies $1 \ll k \ll n$ such that λ_k is not at either end of $[\lambda_1, \lambda_n]$.

input

$$A\vec{y}_k = \lambda_k B\vec{y}_k$$

output

$A, B : n \times n$ large sparse Hermitian matrices (*B* is positive definite.)

λ_k : *k*-th Eigenvalue $\lambda_1 \leq \cdots \lambda_k \leq \cdots \lambda_n$

$\vec{y}_k$: *k*-th Eigenvector

If you want to calculate 2343rd eigenvalue and eigenvector,

```
./example.out A.mtx B.mtx 2343 > output.txt
```

sample code : `example/example.f90`

$K\omega$

(https://github.com/issp-center-dev/Komega/releases)

$K\omega$ is a library to solve the shifted linear equation within the Krylov subspace.

$$G_{ij}(z) = \langle i|(z\hat{I} - \hat{H})^{-1}|j\rangle \equiv \varphi_i^* \cdot (z\hat{I} - \hat{H})^{-1}\varphi_j$$

output

input

For example,

$$\hat{H} = \sum_i \left(\hat{S}_{ix} \quad \hat{S}_{iy} \quad \hat{S}_{iz} \right) \begin{pmatrix} J_x & D_z & 0 \\ -D_z & J_y & 0 \\ 0 & 0 & J_z \end{pmatrix} \left(\hat{S}_{i+1x}\, \hat{S}_{i+1y}\, \hat{S}_{i+1z} \right)$$

- ($\hat{H}$, z) = (complex, complex): Shifted Bi-Conjugate Gradient(BiCG) method *[1]*
- ($\hat{H}$, z) = (real, complex): Shifted Conjugate Orthogonal Conjugate Gradient(COCG) method *[2]*
- ($\hat{H}$, z) = (complex, real): Shifted Conjugate Gradient(CG) method (using complex vector)
- ($\hat{H}$, z) = (real, real): Shifted Conjugate Gradient(CG) method (using real vector)

Hands-on practice

Presentation:
Please review for an app in MateriApps (https://ma.issp.u-tokyo.ac.jp/en/).