Towards using the GRID as a low level library/ Some tips on BiCGstab

Issaku Kanamori (Hiroshima Univ.)

Aug. 1, 2016 @ QCDNA (Edinburgh)

Priority Issue 9, to be Tackled by Using Post K Computer
“Elucidation of the Fundamental Laws and Evolution of the Universe”

Outline

Two independent topics:

1. implementation for the mic architecture
2. solver algorithm

Towards using the GRID as a low level library

— implementation for the mic architecture —

Introduction

We need a fast, flexible and easy-to-maintain code for

- (in 4 years) the next generation of the K-Computer (Post K computer) in Japan
- (before that) something new, KNL based mic architecture

Options

- Designing from the scratch ? — not enough (human) resources
- Just using existing ones s.t. CHROMA or GRID? — an easy solution, but hacking/tuning to specific machines would require a lot. It would be better to have our own code and code developers.
- Partial use of the existing ones.

some of the codes developed in Japanese lattice community:

- Bridge++ (active; flexible but not very optimal)
- Iroiro++ (main developers left, would be unmaintaind)
- Old Fortran codes (running on K-computer)

Our strategy for the KNL mic architecture

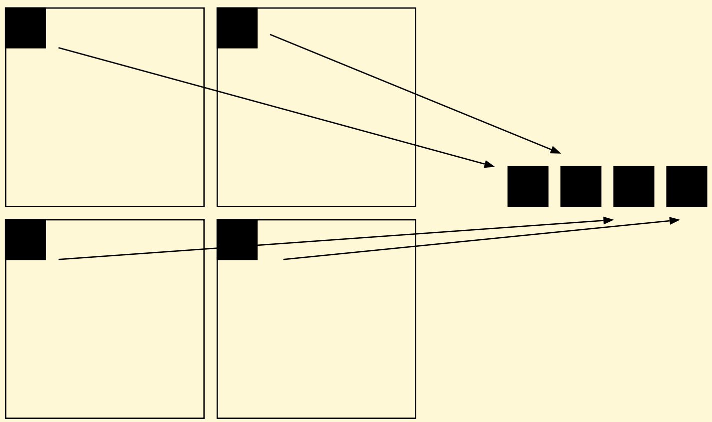
- 1st stage: use an existing general code (e.g., Bridge++) and accelerate it by using tuned codes in solver
- 2nd stage: if the above is not enough (or too nice), accelerate it more

```
Solver and other algorithms: own impelementation/
                             Bridge++?
```

```
Field, Dirac op. (Actions): own impelementation
Lattice: GRID
```

```
Simd related data structure:
  GRID
```

```
Communication,I/O:
  GRID (?)
```



simd vector elements are distributed to different sub-domains (GRID)

Details of Using GRID

field as an array of `LatticeComplex` (=`Lattice<vComplex>`, `vComplex` is a simd vector)

- a large flexibility — easy to use for non-QCD simulation s.t. different gauge rep. of fermion and super Yang-Mills
- less complicated in template

```
  // gauge field
  class Field_Gauge : public Field_ptr {
  public:
    const static int dof=Nc*Nc*Nd;
    GridBase *grid;
    LatticeComplex data[dof];
 ....
    LatticeComplex& operator()(const int mu, const int a, const int b){
      return data[mu*Nc*Nc+a*Nc+b];
    }
....
   };
```

(planned)

- separate the degrees of freedom into `dof_inner` (s.t. color) and `dof_outer` (s.t. spinor, vector etc.)
- field as `Lattice<vComplex[dof_inner]> data[dof_outer]`
- ...may need to hack GRID more.

Performance?

... will appear in Lattice 2017
on KNL machine “Oakforest-PACS”*
at JCAHPC, U. of Tsukuba and U. of Tokyo

* will start full operation in Dec., 2016

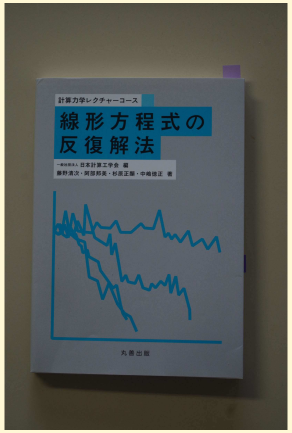
Some tips on BiCGstab

— solver algorithm —

Some tips on BiCGstab: Is it commonly known?

- Almost no cost but very efficient
- I found it in a recent textbook of solvers.
- It might be well-known for the experts in QCD as well.

Comments are very very welcome!



Seiji Fujino, Kuniyoshi Abe, Masaaki Sugihara and Norimasa Nakashima (2013, Maruzen) “線形方程式の反復解法”(=“iterative method for linear equations”)

the original paper:
Gerard L. G. Sleijpen and Herk A. van der Vorst
Numerical Algorithms 10 (1995) 203-223

A standard BiCGstab

A naive residual vector in BiCGstab: $|s\rangle = |b\rangle - A\,|x\rangle$

The stabilized residual vector : $|r\rangle = (1 - \omega A)\,|s\rangle$

ω is determined to minimize the norm or $|r\rangle$:

$$\frac{\partial}{\partial \omega^*}\langle r|r\rangle = 0 \Rightarrow \omega = \frac{\langle As|s\rangle}{\langle As|As\rangle} \equiv \omega^{(0)}$$

Another strategy: avoid poor accuracy of α, β

strategy: modify ω to keep the accuracy of coefficients α and β for updating
(naive BiCG part: $|x\rangle \leftarrow |x\rangle + \alpha\,|p\rangle$, $|p\rangle \leftarrow |s\rangle + \beta\,|p\rangle$)

($|r^*\rangle$: shadow vector)

$$\text{bad}: \left|\left\langle r^* \middle| r\right\rangle\right| \ll \left|\,|r^*\rangle\right|\left|\,|r\rangle\right| \quad \underset{\langle r^*|r\rangle \propto \omega}{\Rightarrow} \quad \text{bad}: |\omega| \ll \left|\,|r\rangle\right|$$

Desirable: maximize $\frac{|\omega|}{\left||r\rangle\right|}$, which requires

$$\frac{\partial}{\partial \omega^*}\frac{\langle r|r\rangle}{\omega\omega^*} = 0 \Rightarrow \omega = \frac{\langle s|s\rangle}{\langle s|As\rangle} \equiv \tilde{\omega}$$

Making a compromise: "vanilla strategy"

$\omega^{(0)}$ and $\tilde{\omega}$ are rewritten:

$$\omega^{(0)} = c\frac{\big||s\rangle\big|}{\big||As\rangle\big|}, \qquad \tilde{\omega} = \frac{1}{c^*}\frac{\big||s\rangle\big|}{\big||As\rangle\big|}, \qquad c \equiv \frac{\langle As|s\rangle}{\big||s\rangle\big|\big||As\rangle\big|}.$$

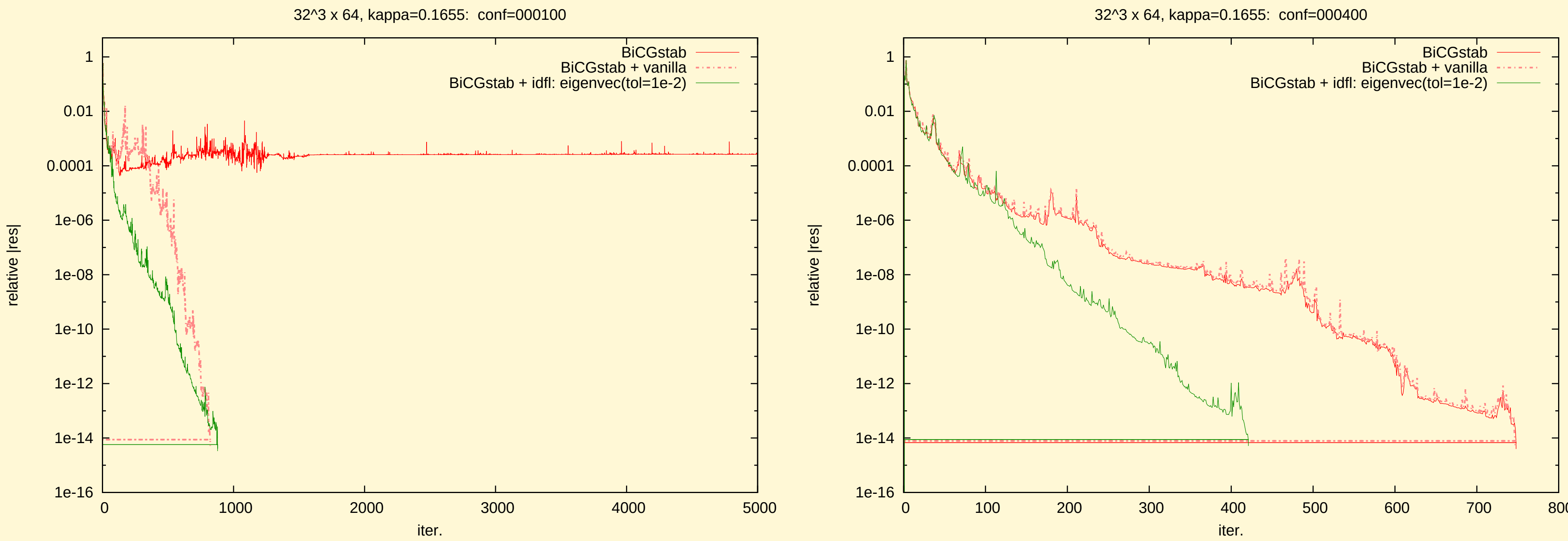
$|c| = 1$ gives $\omega^{(0)} = \tilde{\omega}$, and a smaller $|c|$ gives a larger discrepancy (i.e., looses precision). In order to keep the precision, we introduce a cutoff Ω:

$$|c| \geq \Omega: \qquad \omega = c\frac{\big||s\rangle\big|}{\big||As\rangle\big|} = \frac{\langle As|s\rangle}{\langle As|As\rangle} = \omega^{(0)}$$

$$|c| < \Omega: \qquad \omega = \Omega\frac{\big||s\rangle\big|}{\big||As\rangle\big|} = \frac{\Omega}{|c|}\omega^{(0)}$$

$\Omega = 0.7 \simeq 1/\sqrt{2}$ is a good choice.

Test: seems promising



with (almost) no extra cost, the stability is drastically improved.
.... and faster than inexact deflation (due to the additional cost for the projection)

solve $A|x\rangle = |b\rangle$ for $|x\rangle$ with BiCGstab + vanilla

// initialization

$|x_0\rangle \leftarrow |b\rangle,\ |r_0\rangle \leftarrow |b\rangle - A|x\rangle$

$\left|r_0^*\right\rangle \leftarrow |r\rangle$ (in fact it is arbitrary)

$|p_0\rangle \leftarrow |r_0\rangle,\ \rho_0 \leftarrow \left\langle r^* \middle| r_0\right\rangle,\ j \leftarrow 0$

while $\left|\left|r_j\right\rangle\right|$ is NOT small enough **do**

// BiCG part

$\alpha_j \leftarrow \rho_j / \left\langle r_0^* \middle| A \middle| p_j \right\rangle$ needs A for $\left|Ap_j\right\rangle = A\left|p_j\right\rangle$

$\left|s_j\right\rangle \leftarrow \left|r_j\right\rangle - \alpha_j A\left|p_j\right\rangle,\ \left|x_j\right\rangle \leftarrow \left|x_j\right\rangle + \alpha_j|p\rangle$

if $\left|\left|s_j\right\rangle\right|$ is small enough **then**

break

end if

// Stab part

$\omega_j \leftarrow \left\langle As_j \middle| s_j \right\rangle / \left\langle As_j \middle| As_j \right\rangle$ needs A for $\left|As_j\right\rangle = A\left|s_j\right\rangle$

// vanilla correction of ω_j

$c \leftarrow \left\langle As_j \middle| s_j \right\rangle / \sqrt{\left\langle s_j \middle| s_j \right\rangle \left\langle As_j \middle| As_j \right\rangle}$

if $|c| < \Omega$ **then**

$\omega_j \leftarrow \frac{\Omega}{|c|}\omega_j$ $\Omega = 0.7$ is a good choice

end if

$\left|x_{j+1}\right\rangle \leftarrow \left|x_j\right\rangle + \omega_j\left|s_j\right\rangle,\ \left|r_{j+1}\right\rangle \leftarrow \left|s_j\right\rangle - \omega_j A\left|s_j\right\rangle$

$\rho_{j+1} \leftarrow \left\langle r_0^* \middle| r_{j+1}\right\rangle,\ \beta_j \leftarrow \rho_{j+1}/\rho_j$

$\left|p_{j+1}\right\rangle \leftarrow \left|r_j\right\rangle + \beta_j\left(\left|p_j\right\rangle - \omega_j A\left|p_j\right\rangle\right)$

$j \leftarrow j+1$

end while

Towards using the GRID as a low level

Issaku Kanamori (Hiroshima Univ.)

Aug. 1, 2016 QCDNA (Edinburgh)

library / Some tips on BiCGstab

Priority Issue 9, to be Tackled by Using Post K Computer
“Elucidation of the Fundamental Laws and Evolution of the Universe”