PROCEEDINGS OF SCIENCE

Extending the QUDA library with the eigCG solver

Alexei Strelchenko*
Scientific Computing Division, Fermilab, Batavia, IL 60510-5011, USA
E-mail: astrel@fnal.gov

Andreas Stathopoulos
Computer Science Department, College of William & Mary, Williamsburg, VA 23187-8795, USA
E-mail: andreas@cs.wm.edu

While the incremental eigCG algorithm [1] is included in many LQCD software packages, its realization on GPU micro-architectures was still missing. In this session we report our experience of the eigCG implementation in the QUDA library. In particular, we will focus on how to employ the mixed precision technique to accelerate solutions of large sparse linear systems with multiple right-hand sides on GPUs. Although application of mixed precision techniques is a well-known optimization approach for linear solvers, its utilization for the eigenvector computing within eigCG requires special consideration. We will discuss implementation aspects of the mixed precision deflation and illustrate its numerical behavior on the example of the Wilson twisted mass fermion matrix inversions.

The 32nd International Symposium on Lattice Field Theory
23-28 June, 2014
Columbia University New York, NY

*Speaker.

PoS(LATTICE2014)031

1. Introduction

In the past few years, lattice QCD applications have been continually under intensive development to address computationally demanding scientific problems in nuclear and high energy physics. This development involves elaboration of sophisticated algorithms, on one the hand, and software optimization, on the other, to efficiently exploit contemporary HPC ecosystems. A good example of such efforts are QUDA and recently released QPhiX libraries which also provide with development platforms to design software tools for performing calculations in lattice QCD on NVIDIA Tesla and Intel Xeon Phi based clusters.

In this report we will consider implementation and optimization deflated solvers in the QUDA library. In particular, we will discuss the incremental eigCG algorithm for computing and deflating low-lying eigenmodes while solving multiple right hand side symmetric linear systems [1]. As a by-product, we developed a general framework based on MAGMA GPU library [2], upon which one can build future eigenvector solvers for the lattice QCD computations on GPUs.

2. The mixed precision incremental eigCG in QUDA

In this section we will briefly present our implementation of the mixed precision eigCG algorithm in QUDA in more details. One of our main motivation here is grounded on the fact that in all contemporary micro-architectures single precision arithmetic is at least twice as fast as double precision arithmetic. (For recent HPC-oriented accelerators like NVIDIA Kepler GPUs these differ by a factor of three.) When solving a linear system of equations, there are two widely used approaches of exploiting the possible acceleration provided by single precision arithmetic units, while aiming at a full precision result. The simplest one is to use an iterative refinement strategy, also known as defect-correction method [3]. Such an approach allows the residual to be reduced in an inner solve using low precision, while the residual calculation and solution accumulation are done in high precision. The other approach is based on the reliable updates techniques [4]. Within this scheme, the iterated residual and the solution vector are updated using a low precision matrix-vector product, while errors introduced by the low precision arithmetic in a Krylov solver is rectified periodically by using group-wise updates of these vectors in high precision. The main advantage of the reliable updates is that they do not lead to restarting of the previously generated Krylov sub-space, that necessarily happens in the iterative refinement procedure (see, e.g., [5] for the performance comparison of the techniques).

The advantage of the mixed precision approach in the context of the incremental eigCG is twofold. First, computed eigenvectors can be accumulated in low precision format that will result in both better execution time (for CG deflation) and essentially lower memory requirements. The latter point is especially important for memory-restricted accelerators. Second, in a realistic scenario, the mixed precision *incremental* eigCG might need a fewer number of right hand sides while generating a required number of the eigenvectors. This point will be discussed in more details in the next section. Regarding the algorithmic part we adopt a Matlab-like notations used in the original paper on eigCG [1].

2.1 The algorithm

In this report we consider solving systems of linear equations of the form :

$$Ax_i = b_i \tag{2.1}$$

where $i = 0, \ldots, s$ is the index of a system to solve, and A is a hermitian matrix. For instance, in the QUDA library one is typically dealing with two equivalent types of "even-odd" preconditioning: the so-called symmetric and asymmetric preconditioning. In our analysis we will choose the asymmetric preconditioning for which the matrix A can be represented, e.g., as $A = M_{ee}^{\dagger} M_{ee}$, where

$$M_{ee} = (R_{ee} - \kappa^2 \not{D}_{eo} R_{oo}^{-1} \not{D}_{oe}). \tag{2.2}$$

Here $\not{D}$ describes hopping terms of the fermion matrix and R stands for a local operator. For the numerical experiments, we used a twisted mass fermion operator implementation in QUDA [6].

In the standard incremental eigCG workflow for the Hermitian systems with multiple right hand sides, one distinguishes the incremental phase within which one incrementally computes and improves the approximate Ritz vectors for a relatively small number of eigCG calls and the deflation phase within which one utilizes the computed eigenvectors to deflate the CG for the remaining systems. Our purpose is to combine the former phase with a mixed precision technique and to investigate how this will effect the latter one in real-live scenarios.

Algorithm 1 depicts the iterative refinement version of the incremental eigCG. As in the conventional method, we use a set of m search vectors V to compute nev Ritz vectors within a given eigCG call. These resulting eigenvectors are then accumulated in a larger set U after each iteration of the outer loop over the right hand sides. Unlike the full precision version, in addition to the outer loop, the mixed precision scheme allows also to exploit the inner iterative refinement loop to provide an extra source for the eigenvector accuracy improvement within a single right hand side. Namely, given two different tolerances, an external tolerance ε_h and a "sloppy" tolerance ε_l, we start with the deflated initial guess $\hat{y}_0$, and call low precision eigCG inside the inner iterative refinement loop, provided the residual gap is sufficient to construct nev eigenvalues. (In our experiments, we demanded a few restarts of the search space V within a single eigCG call: using this parameter we decided whether to accumulate the resulting nev Ritz vectors.) Otherwise, one can employ the (deflated) low precision CG instead. The resulting nev Ritz vectors are then used to expand U, in the same manner as it's performed in the outer incremental loop. An important modification of the naive iterative refinement scheme is that we utilize the constructed set U to perform the Galerkin projection of the initial guess $\hat{y}_0$ using the residual $\hat{r}_k$, which is a right hand side for the next system in the iterative refinement procedure. Also, the computed eigenvectors can be stored in low precision (e.g., signle or even half precision format) in the set U.

Let's make a few remarks concerning the algorithm. The parameter ε_l is typically chosen to be as small as possible to avoid unnecessary restarting, subject to the constraint $\varepsilon_l > ulp^l$. In practice, we work with IEEE single precision epsilon ($\varepsilon_l \sim 10^{-7}$) and we aim at a final norm of the residual of the order of 10^{-10}. Obviously, the initCG can be a mixed precision solver itself. In particular, we made use of the single-half and double-half mixed precision CG based on reliable updates in the incremental and initCG phases, respectively. Note, however, that the iterative eigCG is in single precision.

Set $U = [], H[]$ % accumulated Ritz vectors;
for $i = 1 : s_1$ *or until* U *is complete* **do**
  Set $k = 0,\ V = []$ % restart index and the eigCG search space;
  $\hat{y}_0 = x_0^{(i)} + UH^{-1}U^{\dagger}(b^{(i)} - Ax_0^{(i)})$;
  $r_0 = b^{(i)} - A\hat{y}_0$;
  $\hat{r}_0 = r_0$;
  while $\|r_k\| > \varepsilon_h$ **do**
    if $\delta\|r_k\| > \varepsilon_h$ **then**
      Solve $[\hat{y}, V] = eigCG(nev, m, A, \hat{y}_0, \hat{r}_k)$ to precision ε_l;
      $\bar{V}$ = orthonormalize V against U;
      Set $U = [U, \bar{V}]$ % augment U;
      Update the projection matrix H ;
    else
      Solve $A\hat{y} = \hat{r}_k$ with initial guess $\hat{y}_0$ using initCG to precision ε_l;
    end
    $x_{k+1}^{(i)} = x_k^{(i)} + \hat{y}$;
    $r_{k+1} = b - A\hat{y}$;
    $\hat{r}_{k+1} = r_{k+1}$;
    Deflate initial guess $\hat{y}_0 = UH^{-1}U^{\dagger}\hat{r}_{k+1}$;
    $k = k + 1$;
  end
end

Algorithm 1: The mixed precision incremental eigCG(nev, m) algorithm with the outer solver tolerance ε_h and inner solver tolerance ε_l, respectively. The upper index in parentheses corresponds to outer right hand side index and the lower index is a iterative refinement loop index. ($\hat{}$) denotes low precision. Parameter δ represents the leftover residual that indicates how close the iterated residual r_k to the tolerance ε_h.

2.2 Implementation notes

To integrate this algorithm in QUDA, we enhanced the spinor classes `ColorSpinorParam` and `ColorSpinorField` to allow one to collect and work with eigenvectors sets. In particular, one typically needs to perform operations like dslash application or BLAS operations including spinor-matrix-complex-matrix multiplications. Currently, dslash can be applied on a single element from the eigenvector set; operations with a subset or the whole set of the eigenvectors will be added in the future optimizations.

Next, the incremental eigCG solver is provided by the `IncEigCG` class that encapsulates all necessary parameters and methods, including the solver driver to execute the solver on GPUs. For the end-user this driver is exposed via a convenient C-like interface function `incrEigQuda`.

Finally, for the search vector restarts within the eigCG we made use of MAGMA library that is a collection of LAPACK routines ported on the CUDA platform. Based on this library we dveloped an interface that can be exploited for the future eigenvector solvers in QUDA such eigBiCG or GMRES-DR.

Table 1: Mixed precision incremental eigCG performance for 164 right hand sides, $24^3 \times 48$ dynamical twisted mass configuration with $\kappa = 0.163270,\ \ \mu = 0.0040$

Solver type	Inc. NRHS	Iters.	Time (sec.)	Speed up
incremental eigCG (mixed)	29	453	345	x4.8
incremental eigCG (full)	36	411	476	x3.5
undeflated CG (mixed)	N.A.	4200	1656	x1.0

Table 2: Mixed precision incremental eigCG performance for 176 right hand sides, $48^3 \times 96$ dynamical twisted mass configuration with $\kappa = 0.156361,\ \ \mu = 0.0015$

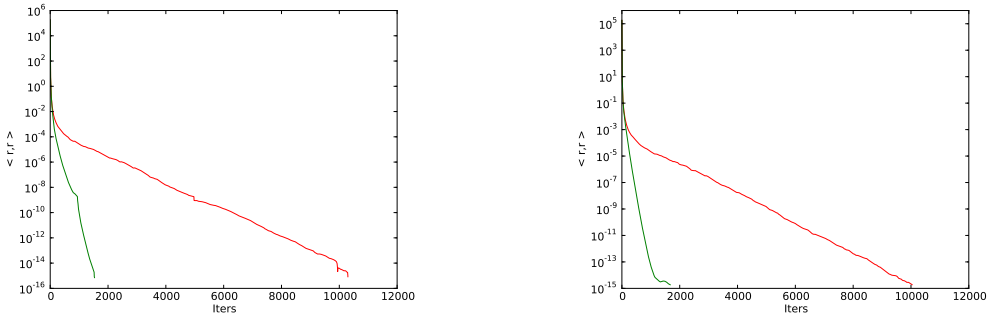
Solver type	Inc. NRHS	Iters	Time (sec.)	Speed up
incremental eigCG (mixed)	24	1285	1532	x5.9
incremental eigCG (full)	48	1008	3152	x2.9
undeflated CG (mixed)	N.A.	10000	9000	x1.0

3. Numerical experiments

In this section we will demonstrate numerical behavior of the mixed precision incremental eigCG combined with the asymmetric preconditioning for the twisted mass fermion matrix inversions. All tests were performed on the recently deployed USQCD pi0g GPU cluster at Fermilab equipped with NVIDIA Tesla K40 GPUs. Each compute node comprises four accelerators based on Kepler GK110 micro-architecture and 12GB on-board memory. We investigated two cases: $24^3 \times 48$ lattice with $\kappa = 0.163270,\ \ \mu = 0.0040$, and $48^3 \times 96$ lattice with $\kappa = 0.156361,\ \ \mu = 0.0015$. The smaller lattice was executed on a single pi0g node, while for the bigger one we had to utilized 24 compute nodes. We summarized our results (for the external tolerance 10^{-10}) in Tables 1-2.

Now a few remarks about the obtained results. For the smaller lattice we considered 164 total right hand sides and we computed 288 aproximate eigenvectors to perform deflation in the initCG stage. For the bigger one we selected 176 systems and 384 aproximate eigenvectors, respectively. In both cases, the eigCG parameters were chosen to be $m = 144$ and $nev = 8$. The first column indicates which solver was used in the analysis: the mixed precision incremental eigCG, standard (full precision) incremental eigCG and undeflated (mixed precision) CG. The second column in the tables shows actual number of RHS in the *incremental* stage to compute eigenvectors, while the third column collects average number of iterations taken by the (un)deflated CG to converge in the *initCG* stage. Next, the forth column provides total execution times and the last one contains information about the observed speed-up w.r.t. undeflated inversions in terms of the total execution time required to solve all systems.

For the $24^3 \times 48$ lattice (see Table 1) one can see that the incremental stage requires less number of right hand sides to get the given number of Ritz vectors, for the price of slightly worse



(a) 1a: mixed precision

(b) 1b: full precision

Figure 1: Convergence of the incremental eigCG for the first and the last RHS.

eigenvector quality. Nevetheless, our main observertion is that the mixed precision deflation results in better total execution time and overall performance gain w.r.t. undeflated solves.

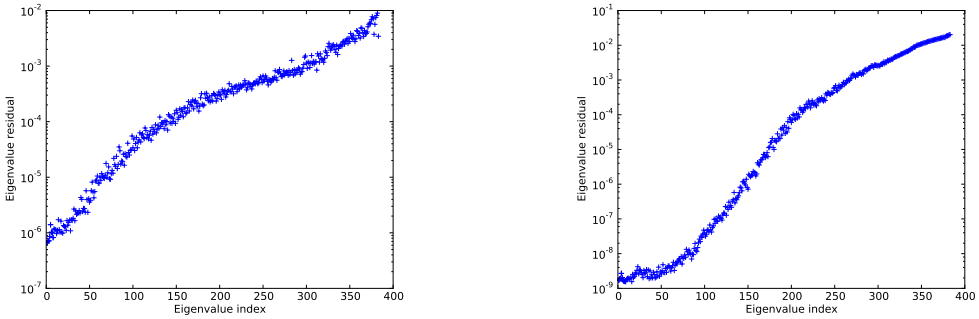
Next, Table 2 gives similar information for the $48^3 \times 96$ lattice, where efficiency of the mixed precision incremental stage is the most prominent. Here we note an almost $\times 2$ speed-up of the mixed precision eigCG solves w.r.t. the full precision one. It should be emphazised, however, that if the number of RHS would be sufficiently high, the performance difference will be eliminated due to better deflation quality of the full precision incremental eigCG. That is, we conclude that the mixed precision approach might be preferable for larger lattices (and smaller quark masses), and when the number of systems to solve is not too high (say, of order $O(100)$ or less).

Our next goal is to illustrate the performance profile of the mixed precision eigCG on the example of the $48^3 \times 96$ lattice. As we could see from Table 2, in the incremental stage required half the number of RHS to get 384 eigenvectors. This means that the iterative refinement algorithm allows to execute two eigCG calls per right hand side, to accumulate the eigenvectors. This fact also explains significantly lower runtime for the mixed precision version.

Fig. 1 presents convergence plots for mixed (left side) and full (right side, resp.) precision incremental stage inversions. Here we choose the first and the last right hand sides in each case. Finally, Fig. 2 compares the residual norm of the computed eigenvalues computed by mixed (left side) and full (right side, resp.) precision incremental eigCG.

4. Conclusion

We extended the QUDA library with the incremental eigCG solver. The main objective of this study was to combine mixed precision technique with eigenvectors computing. In fact, our aim is twofold. First, one can reduce the overall execution time using low precision arithmetics (and more faster device-memory communications), and second, one can optimize global memory usage for the computed eigenvectors. The last point is especially important since deflation methods are usually resource demanding, which is critical for the accelerators. Currently, QUDA does not support low precision storage for the eigenvectors; this feature will be included in future releases.



(a) 2a: mixed precision

(b) 2b: full precision

Figure 2: Eigenvalue residual norms computed within the incremental stage for the operator $A = M^{\dagger}M$, where M is given in (2). Results were obtained using 24 (left plot) and 48 (right plot) right-hand sides, respectively.

For the particular case of the $48^3 \times 96$ lattice we found that mixed precision eigCG can provide around $\times 2$ performance gain over the full precision eigCG. (And $\sim \times 6$ gain over the undeflated CG solves). That is, in realistic scenarios, i.e., when the number of right hand sides is not too big, mixed precision approach allows to achieve an appropriate deflation quality with significantly better overall execution time. The code will be included in the next production release of QUDA (ver. 0.7) and is available in the master branch [7].

5. Acknowledgements

This work was supported by the SciDAC 3 project. A. Stathopoulos was supported by grants NSF CCF 1218349 and DOE DE-FC02-12ER41890. The computations for this report were done on pi0g cluster at Fermilab.

References

[1] A. Stathopoulos and K. Orginos, SIAM J. Sci. Comput. **32**, 439 (2010)

[2] http://icl.cs.utk.edu/magma/index.html

[3] R. S. Martin, G. Peters and J. H. Wilkinson, "Handbook series linear algebra: Iterative refinement of the solution of a positive definite system of equations", Numerische Mathematik **8**, 203216 (1966).

[4] G. L. G. Sleijpen, and H. A. van der Vorst, "Reliable updated residuals in hybrid Bi-CG methods", Computing **56**, 141 (1996)

[5] M. A. Clark *et al*, Comput. Phys. Commun. **181**, 1517 (2010) [arXiv:0911.3191 [hep-lat]].

[6] A. Strelchenko *et al*, PoS LATTICE **2013**, 415 (2014) [arXiv:1311.4462 [hep-lat]].

[7] QUDA library. http://lattice.github.io/quda/.