Accelerating deflation of eigenvalues for fermion matrix inversions on GPUs

Alexei Strelchenko

Scientific Computing Division @ Fermilab

The 33rd International Symposium on Lattice Field Theory
Kobe, July 16

Outline

- Mixed precision deflation algorithms in QUDA
  - Symmetric/non-symmetric systems with many RHS
- Numerical experiments
- Current status and future work

Deflation algorithms in QUDA

- Lanczos (Hyung-Jin Kim)
- The Incremental eigCG
- GMRES-DR
- GMRES-Proj

Notes:

1. Lanczos is callable from CPS, no internal test routines
2. Lanczos and eigCG/GMRES-DR use different infrastructure for eigenvector computing (lots of overlaps)

Aspects of mixed precision deflation on accelerators

- Better FP performance
- Better GPU memory usage
- Less storage requirements
- Better network usage
- Accelerators are faster in low precision (e.g., for Kepler, DP/SP-ratio is 1 / 3)

- Some ideas:
  - A. Frommer *et al*, arXiv:1204.5463
  - A.S. and A. Stathopoulos, PoS Lattice 2014, 031
- Current status:
  - https://github.com/lattice/quda/tree/feature/deflation

The eigCG(k, m) algorithm

- A. Stathopoulos and K. Orginos, SIAM J.Sci.Comput. 32 (2010) 439-462

Input: $A \in \mathbb{C}^{n\times n}$, b, restart length m, Ritz dim. $2k$
% $V^{(m)} \in \mathbb{C}^{n\times m}$: orthonorm. basis of the search subspace;
repeat
  % Run normal CG iterations for $j = 0, 1, 2...$;
  % Construct the Lanc. basis $V^{(m)}$ and the proj. matrix $T_m = V^\dagger AV$;
  % Restart periodically with $2k$ Ritz vecs computed from T_m and T_{m-1} ;
  % Update approx. solution x_j, and residual r_j for each iteration ;
until *convergence*;

Incremental eigCG framework: summary

- Incremental phase
  - for the first s_1 RHS: call $eigCG(k, m)$ solver
  - add k eigenvectors to a separate subspace after each RHS.
- Init-CG phase
  - for all subsequent RHS (i.e., $s > s_1$): apply Galerkin oblique projection
  - call standard CG

The mixed precision eigCG

Set the accumulation array $U[]$, apply iterative refinements:

1 Run eigCG(k,m) until convergence in low precision
2 Accumulate k eigenvectors in U
3 Accumulate the solution
4 Recompute the residual in higher precision $r = b - Ax$

If not accurate enough:

5 Project out U from the residual r
6 Go to step 1 and solve **deflated** error equation $Ae = r$ in low prec.

The GMRES-DR(k,m) algorithm

- R. Morgan, SIAM J.Sci.Comput. 24 (2002) 20-37

Input: $A \in \mathbb{C}^{n\times n}$, b, restart length m, harm. Ritz dim. k
% First cycle: run GMRES (no deflation): compute V_{m+1}, $\bar{H}_m$;
% Compute aprox. solution x_m, and residual r_m ;
repeat
  % Compute k lowest eigenpairs of $H_m + \beta^2 H^{-H} e_m e_m^H$, $\beta = h_{m+1,m}$;
  % Build augmenting subspace $\mathcal{U}$ (spanned by harm. Ritz vectors) ;
  % Update $H_m \leftarrow H_k^{new}$ s.t. $AV_k^{new} = V_{k+1}^{new} \bar{H}_k^{new}$ holds ;
  % Run Arnoldi process for $i = k+1, ..., m$;
  % Compute aprox. solution x_m, and residual r_m for this cycle ;
until *convergence*;

Systems with many RHS : GMRES-DR/GMRES-Proj algorithm

- GMRES-DR phase
  - solve the first RHS with GMRES-DR(k, m) solver
  - store k eigenvectors and the projection matrix.
- Projection phase
  - for all subsequent RHS: apply Minres projection
  - apply one cycle of GMRES(m)
  - test residual norm for convergence. If not satisfied, repeat all steps

The mixed precision GMRES-DR(k,m)

(Following Frommer et al)

Apply iterative refinements:

Refinement cycles correspond one-to-one to the restart cycles of GMRES-DR

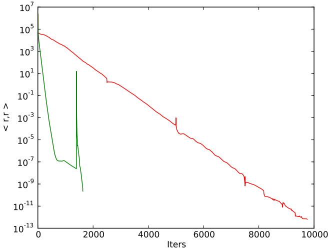
After each cycle :
recompute the residual in HP $r = b - Ax$ and convert it back to LP
If the iterative and the exact residual differ by more than a small amount:
Do a "clean" restart (don't use any deflation subspace)

Lattice setup

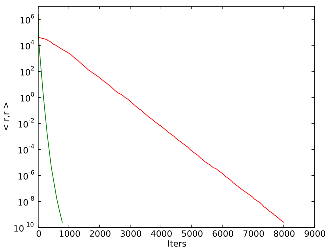
- HISQ (2+1+1)-flavor ensemble: $32^3 \times 48$, $M_\pi = 131.0(MeV), m_l/m_s = 1/27$
- Twisted mass ensemble: $48^3 \times 96$, $M_\pi = 230.0(MeV)$

- eigCG parameters: $nev = 8,\ m = 144$, $tol = 10^{-8}(HISQ), 10^{-10}(TMF),\ \ tol_{rest} = 10^{-5}(HISQ), 10^{-6}(TMF)$
- GMRES-DR parameters: $nev = 144,\ m = 192,\ \ tol = 10^{-10}$

- Used pi0g nodes @ FNAL, Tesla K40 accelerators



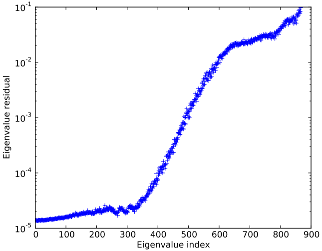
(a) 1a: mixed precision



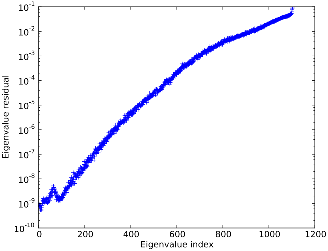
(b) 1b: full precision

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

Eigenvalue accuracy ($32^3 \times 48$ HISQ)



(a) 2a: mixed precision



(b) 2b: full precision

Figure : Eigenvalue residual norms of the even-odd HISQ operator, computed within the incr. stage

$32^3 \times 48$ **HISQ**

Mixed precision incremental eigCG performance, 1280 eigenvectors, $32^3 \times 48$ HISQ configuration (8 nodes, 32 K40m)

Solver type	Inc. NRHS / Time	CG Iters	CG Time
inc. eigCG (full)	160 (2315)	516(x15.5)	1.10 (x10.5)
inc. eigCG (mixed)	104 (1544)	946 (x8.4)	1.63 (x7.1)
undef. CG (mixed)	N.A.	8000 (x1.0)	11.6 (x1.0)

$32^3 \times 48$ HISQ

Mixed precision incremental eigCG performance, 800 eigenvectors, $32^3 \times 48$ HISQ configuration (6 nodes, 24 K40m)

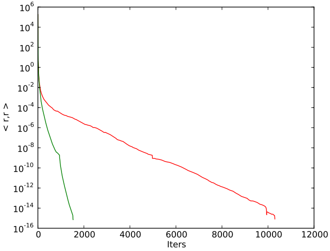
Solver type	Inc. NRHS / Time	CG Iters	CG Time
inc. eigCG (full)	100 (1534)	790(x10.1)	1.48 (x7.8)
inc. eigCG (mixed)	46 (642)	1255 (x6.4)	2.13 (x5.4)
undef. CG (mixed)	N.A.	8000 (x1.0)	11.5 (x1.0)

$32^3 \times 48$ HISQ (single vs double prec. eigenvecs)

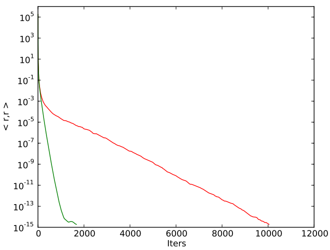
The incremental eigCG performance,
$32^3 \times 48$ HISQ configuration .

Eigv. prec	GPU nodes (NEV)	Deflated CG (Time)	GB per GPU
DP	6 (800)	1.48	6.5
SP	6 (800)	2.13	3.3
DP	4 (800)	1.54	9.4
SP	4 (800)	2.23	4.9
DP	2 (480)	3.22	9.8
SP	2 (800)	3.20	8.4

Incremental eigCG convergence ($48^3 \times 96$ TMF)



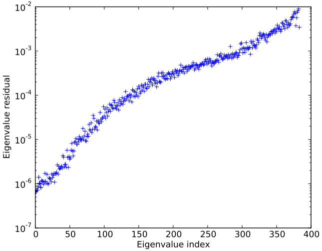
(a) 1a: mixed precision



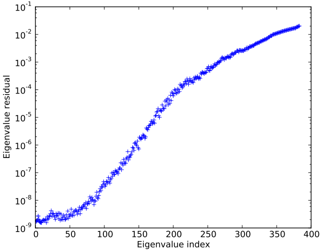
(b) 1b: full precision

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

Eigenvalue accuracy ($48^3 \times 96$ TMF)



(a) 2a: mixed precision



(b) 2b: full precision

Figure : Eigenvalue residual norms for the operator $A = M^\dagger M$

$48^3 \times 96$ TMF, 384 evecs (24 nodes, 96 GPUs)

Mixed precision incremental eigCG performance, 384 eigenvectors,
$48^3 \times 96$ dynamical twisted mass configuration with $\kappa = 0.156361, \quad \mu = 0.0015$

Solver type	Inc. NRHS	CG Iters	CG Time
inc. eigCG (full)	48	1001 (x10)	6.1 (x8.8)
inc. eigCG (mixed)	24	1370 (x7.3)	7.8 (x7.0)
undef. CG (mixed)	N.A.	10000 (x1.0)	54.0 (x1.0)

$48^3 \times 96$ TMF (single vs double prec. eigenvecs)

Mixed precision incremental eigCG performance,
$48^3 \times 96$ dynamical twisted mass configuration with $\kappa = 0.156361, \quad \mu = 0.0015.$
Computed on 96 GPUs (24 nodes).

Eigv. prec	NEV	Deflated CG (Iters / Time)	GB per GPU
DP	384 (48 RHS)	1001 /6.1	10.0
SP	384 (24 RHS)	1370 /7.8	5.5
SP	640 (40 RHS)	856 /5.3	9.0
SP	768 (48 RHS)	753 /4.9	10.5

$48^3 \times 96$ lattice (eigenvec. accuracy, 96 GPUs)

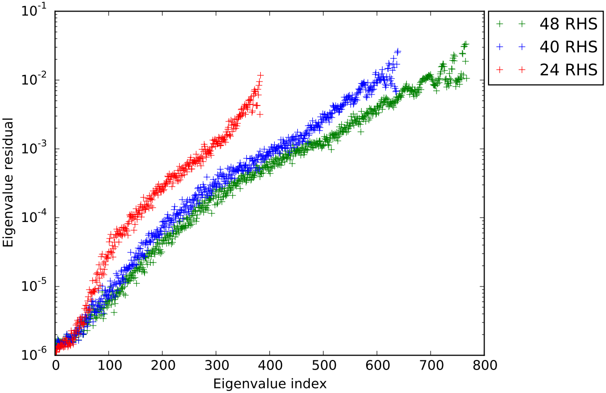


Figure : Eigenvector residual norms, computed on 24 pi0g nodes.

$48^3 \times 96$ TMF (Strong scaling)

Mixed precision incremental eigCG performance,
$48^3 \times 96$ dynamical twisted mass configuration with $\kappa = 0.156361, \quad \mu = 0.0015$

GPUs	NEV	Deflated CG (Iters / Time)	GB per GPU
96 (SP)	768 (48 RHS)	753 /4.9	10.5
64 (SP)	512 (32 RHS)	1045 /6.9	10.0
48 (SP)	384 (24 RHS)	1311 /8.0	10.5
96 (DP)	384 (48 RHS)	1001 /6.1	10.0

$48^3 \times 96$ TMF, GMRES-DR convergence (*preliminary*)

Mixed precision GMRES-DR($k = 144$, $m = 192$) performance,
$48^3 \times 96$ dynamical twisted mass configuration with $\kappa = 0.161322$, $\mu = 0.004$

- GMRES(m) - stagnates at tolerance 10^{-3}
- full precision GMRES-DR(k, m) - 12 nodes, exec. time 1690 secs (14063 iters)
- mixed precision GMRES-DR(k, m) - 8 nodes, exec. time 1517 secs (17862 iters)
- mixed precision GMRES(m)-Proj(k) - 8 nodes, exec. time 432 secs (22560 iters)

Future work

- Deflation:
  - multi-mass version of GMRES-DR
  - flexible version of GMRES-DR
  - SAP + FGMRESDR
  - more optimizations for systems with many RHS