A staggered eigensolver based on sparse matrix bidiagonalization

James C. Osborn
Xiao-Yong Jin

Argonne National Lab

Lattice 2017
Grenada, Spain

SVD-based Staggered Eigensolver

- Staggered Dirac matrix

$$\begin{pmatrix} 0 & A \\ -A^\dagger & 0 \end{pmatrix} \qquad A = D_{oe}$$

- Eigenvalues from SVD

$$\mathrm{A} = \mathrm{Q}\ \Lambda\ W^\dagger \qquad \text{Eigenvalues} \rightarrow \pm i\Lambda$$

- Direct eigenvalue options

normal equations

$$A^\dagger A$$

Jordan-Wielandt form

$$\begin{pmatrix} 0 & A \\ A^\dagger & 0 \end{pmatrix}$$

- Normal equations can be difficult for poorly conditioned systems

SVD-based Staggered Eigensolver

- Directly calculate SVD of Staggered Dirac matrix
- Based on Golub-Kahan-Lanczos bidiagonalization

$$B_k = \begin{pmatrix} \alpha_1 & \beta_1 & & \\ & \alpha_2 & \ddots & \\ & & \ddots & \beta_{k-1} \\ & & & \alpha_k \end{pmatrix}$$

$$\mathrm{A}\ \mathrm{V}_k = U_k B_k$$

$$A^\dagger U_k = V_k B_k^\dagger + \beta_k v_{k+1} e_k^\dagger$$

- Iterative process, start with v_1, generate u_1 and B_1, …
- Reduce problem to SVD of bidiagonal matrix B

$$BX = Y\Lambda$$

$$A(VX) = (UY)\Lambda \rightarrow AW = Q\Lambda$$

Error correction

- Iterative SVD not exact

$$AW = Q\Lambda \to Q^\dagger AW = Q^\dagger Q\Lambda$$
$$Q^\dagger Q \approx 1$$

- Lanczos vectors lose orthogonality
  - Reorthogonalize (full or selective)
  - Can be expensive: typically run for O(10+ k) iterations
  - Staggered D relatively cheap
  - Only need O(1k) low eigenvectors
- Can fix orthogonality after SVD with Rayleigh-Ritz update

$$(W^\dagger A^\dagger AW)Z = (W^\dagger W)ZE$$
$$W' = WZ$$

SVD-based Staggered Eigensolver history

- Originally developed code around 2011 using Qlua & QOPQDP
- Used to study eigenvalue/vector statistics and deflation
- Wanted high precision especially for near-zero modes
- Developed iterative approach

```
V1 = GKLanczosSVD( randomVector )
repeat until converged:
  x = smallestUnconverged( V1 )     $r = A^\dagger A v - \lambda^2 v$     $\delta(\lambda^2) \leq |r|$
  V2 = GKLanczosSVD( x )
  V1 = combine(V1, V2)
```

SVD-based Staggered Eigensolver history

- Combine
  - Ideally full diagonalization in span of V1, V2
  - Full diagonalization can be approximated with many smaller ones
  - Diagonalize over spaces of eigenvectors with similar eigenvalues
  - Lots of ways to do this, makes big difference in convergence
- Qlua code did many smaller diagonalizations
  - Several tuning parameters could have large impact on performance

- Recently ported code to new framework
- Started experimenting on improving the method

Updated SVD-based Staggered Eigensolver

- Ported Qlua code to QEX (Quantum Expressions)
  - New LFT framework in Nim language (Fri. 18:30-18:50 Soft. Dev.)
  - Nim has python-like syntax, but strongly typed, compiles to C/C++
  - Lua → Nim fairly easy by hand (Lua dynamic types → Nim generics)
  - C → Nim done via c2nim plus cleanup by hand

- Tuning code and algorithm
  - More efficient GKL-SVD scales well to 100+ k iterations
  - Experimenting with generating eigenvectors with single run

HISQ eigensolver tests

- MILC HISQ lattices

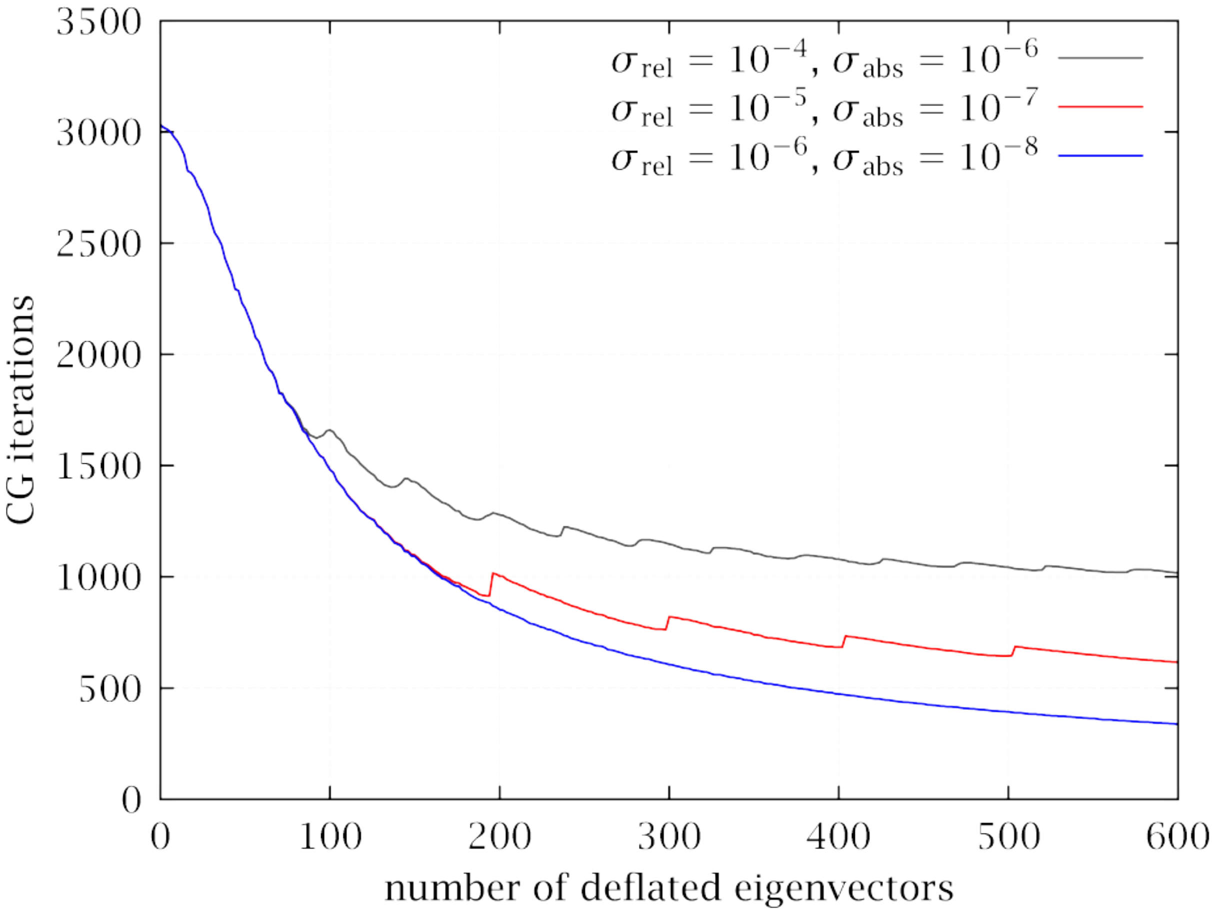
size	a (fm)	m_l / m_s	λ_1 SVD iterations
24^3 x 48	0.15	1/10	35 k
32^3 x 48	0.15	1/27	80 k
48^3 x 64	0.12	1/27	200 k
64^3 x 96	0.09	1/27	~300k

- SVD iterations for lowest eigenvalue to converge (many more will too)
- Can’t store all 200k SVD vectors – make 2 passes
- Single large SVD not necessarily best strategy, but works well so far

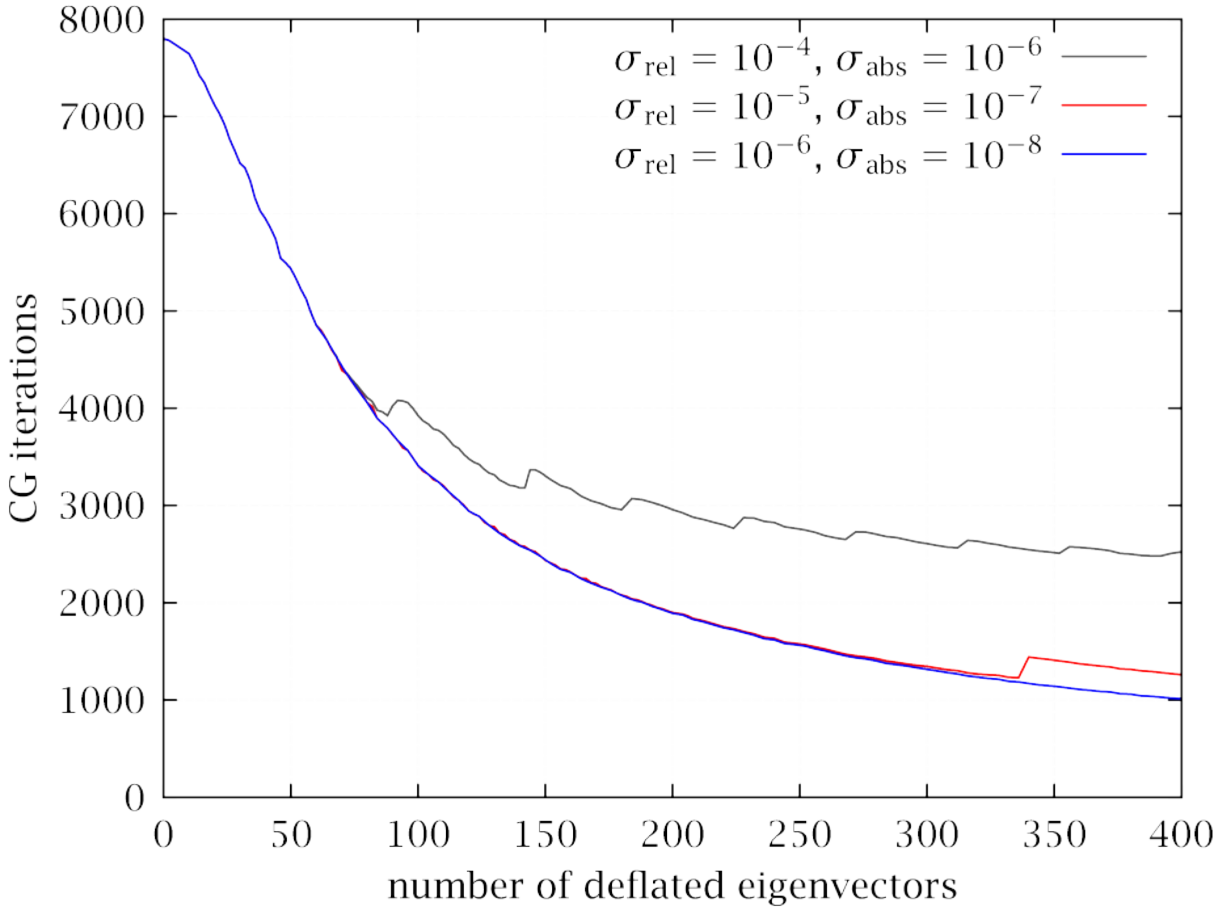
Testing

- Using deflation as test for necessary quality of eigenvectors
- Deflate eigenvectors, then run solver and look at reduction in iterations
- Comparing eigensolver time to PRIMME (http://www.cs.wm.edu/~andreas/software)
  - Many tuning parameters
  - Tuning can give ~3x or more improvement over defaults
- More tuning possible for both codes, just used as a reference
- Results from BG/Q

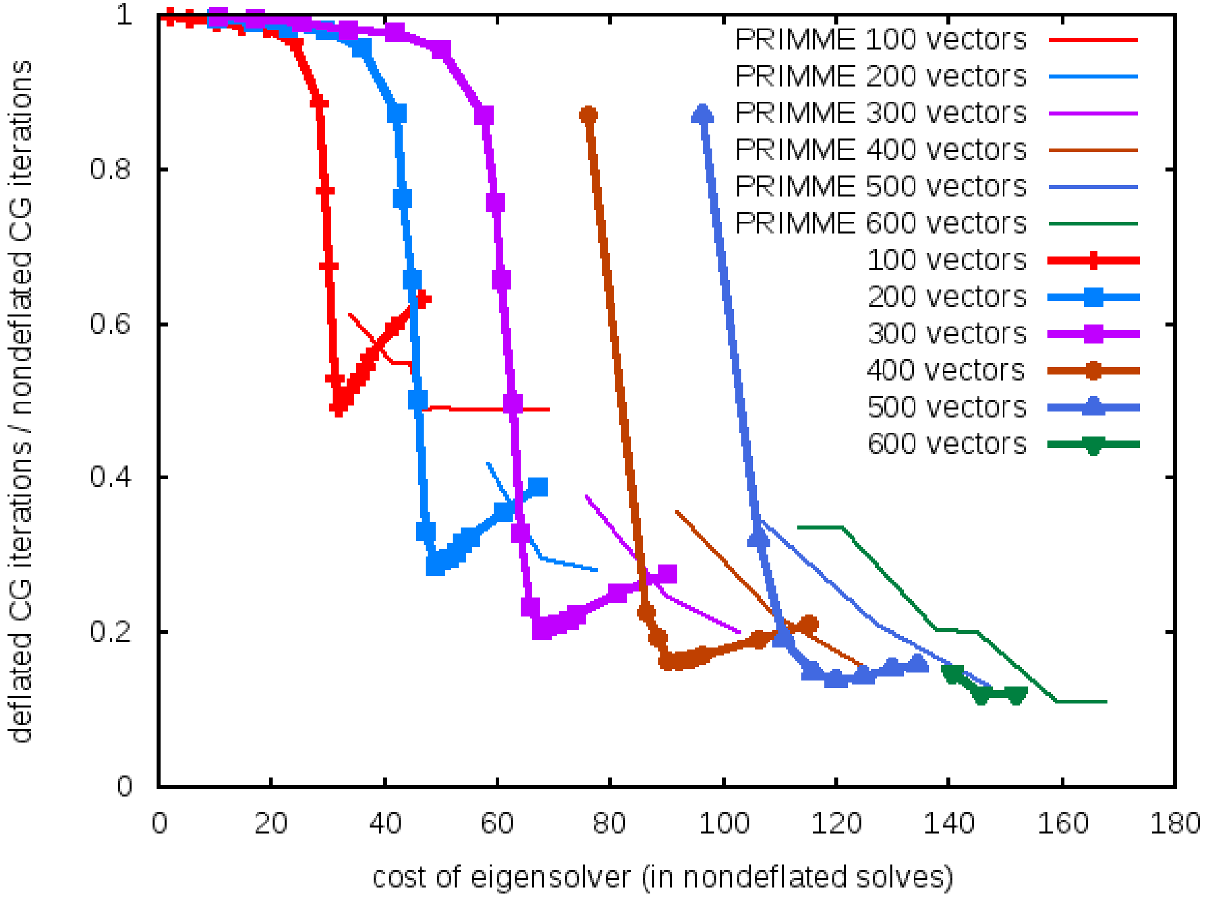
Effect of residual on deflation (24^3 x 48, generated with PRIMME)



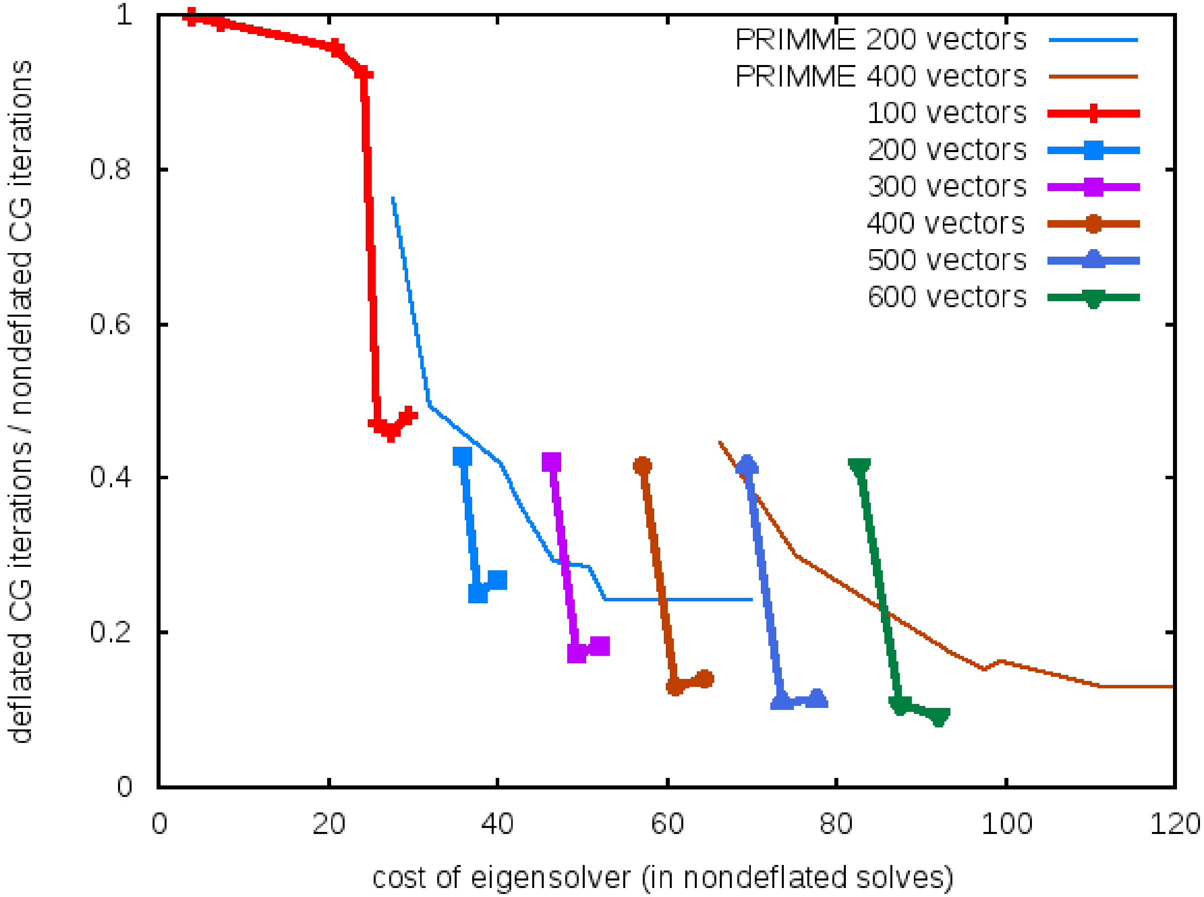
Effect of residual on deflation (32^3 x 48, generated with PRIMME)



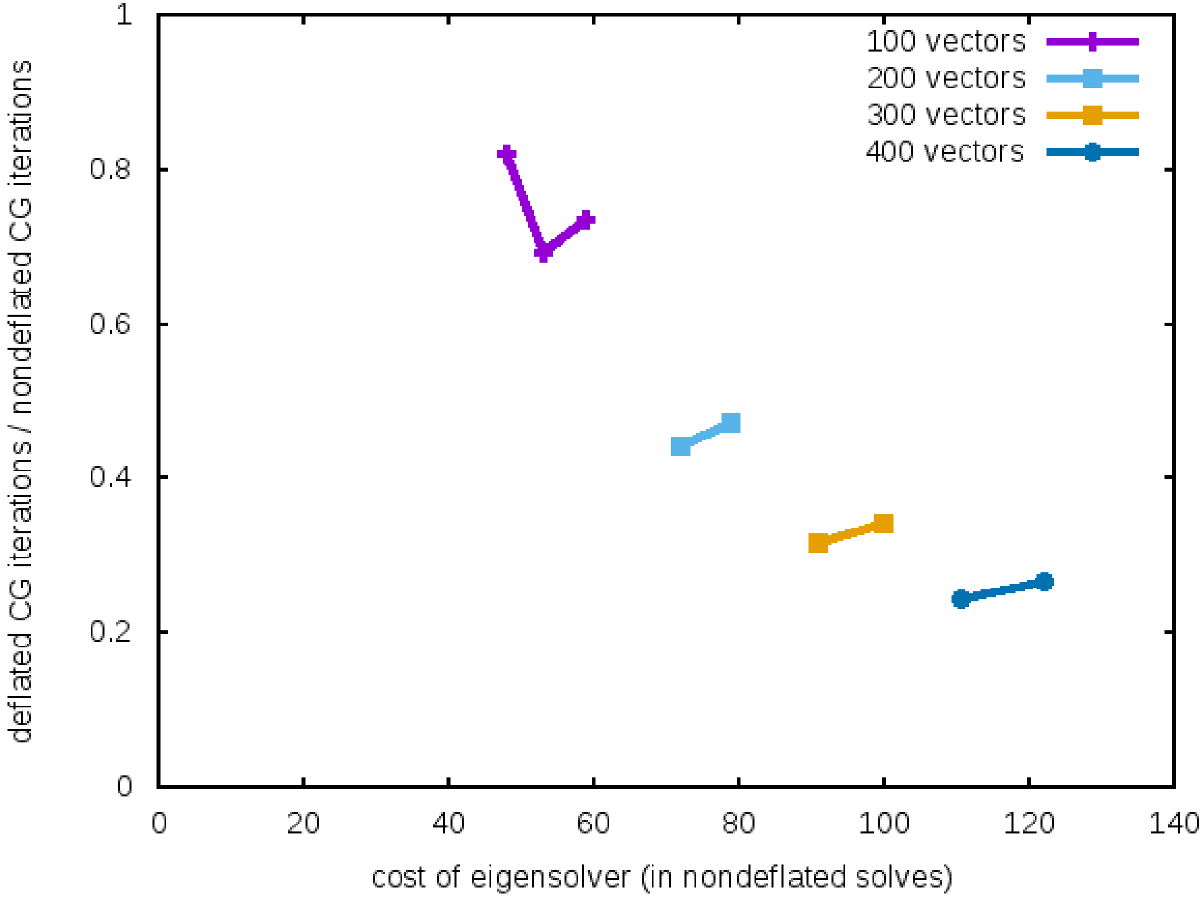
Deflated iterations vs. eigensolver cost (24^3 x 48)



Deflated iterations vs. eigensolver cost (32^3 x 48)



Deflated iterations vs. eigensolver cost (48^3 x 64)



Summary

- Golub-Kahan-Lanczos bidiagonalization provides viable staggered eigensolver
- Good results on up to 48^3 x 64 (and some initial tests on 64^3 x 96)
- Provides very accurate eigenvectors
- Deflation works well on 48^3 x 64 with O(500) vectors
- Starting tests of deflation directly in SVD
(save a reduced set of starting vectors)

- Extras

QEX (Quantum Expressions)

- QEX: LQCD framework written in Nim
  - https://github.com/jcosborn/qex
- Nim: high-level language
  - https://nim-lang.org
  - Python-like syntax
  - Strongly typed
  - Very flexible generics
  - Powerful metaprogramming
  - Compiles to C/C++

QEX example: main Lanczos Bidiagonalization loop

```
while true:
  v := p / beta
  linop.apply(r, v)
  r -= beta*u
  alpha = sqrt(r.norm2)
  a[k] = alpha
  inc k
  if k >= kmax: break

  u := r / alpha
  linop.applyAdj(p, u)
  p -= alpha*v
  beta = sqrt(p.norm2)
  b[k-1] = beta
```