PROCEEDINGS OF SCIENCE

A CG method for multiple right hand sides and multiple shifts in lattice QCD calculations

Sebastian Birk[*] [†]
Fachbereich C, Mathematik und Naturwissenschaften, Bergische Universität Wuppertal, D-42097 Wuppertal, Germany
E-mail: birk@math.uni-wuppertal.de

Andreas Frommer
Fachbereich C, Mathematik und Naturwissenschaften, Bergische Universität Wuppertal, D-42097 Wuppertal, Germany
E-mail: frommer@math.uni-wuppertal.de

We consider the task of computing solutions of linear systems that only differ by a shift with the identity matrix as well as linear systems with several different right hand sides. In the past Krylov subspace methods have been developed which exploit either the need for solutions to multiple right hand sides (e.g. deflation type methods and block methods) or multiple shifts (e.g. shifted CG) with some success. In this paper we present a block Krylov subspace method which, based on a block Lanczos process, exploits both features—shifts and multiple right hand sides—at once. Such situations arise, for example, in lattice QCD simulations within the Rational Hybrid Monte Carlo algorithm. We give numerical evidence that our method is superior to applying other iterative methods to each of the systems individually as well as, in some cases, to shifted or block Krylov subspace methods.

XXIX International Symposium on Lattice Field Theory
July 10-16, 2011
Squaw Valley, Lake Tahoe, California

[*]Speaker.

[†]This work was supported by Deutsche Forschungsgemeinschaft through the Collaborative Research Centre SFB-TR 55 "Hadron physics from Lattice QCD"

PoS(Lattice 2011)027

1. Introduction

In this paper we consider solving systems of linear equations of the form

$$(\sigma_j I + A)x_{i,j} = b_i \tag{1.1}$$

where $\sigma_j I + A \in \mathbb{C}^{n\times n}$ is a hermitian positive definite (hpd) matrix for every shift $\sigma_j \in \mathbb{C}$, $x_{i,j}, b_i \in \mathbb{C}^n$, $i = 1,\ldots,m$ and $j = 1,\ldots,s$.

Systems like these arise naturally in lattice QCD, where the b_i represent multiple source terms, and A is a discrete version of the Dirac operator like, e.g. the Wilson fermion matrix. For example the Rational Hybrid Monte Carlo algorithm [8] needs to solve $A^\nu x_i = b_i$ with $\nu \in (-1,1)$ for multiple right hand sides (rhs) b_i. The matrix power A^ν therein is computed using a rational approximation which can be represented as a partial fraction expansion thus giving rise to multiple shifted systems. Another application is the computation of symplectic integrators that require the evaluation of $A^\nu B A^\nu x_i = b_i$ for multiple random right hand sides. In that case the inner application of A^ν is approximated via a partial fraction expansion that generates multiple right hand sides for the outer application even if there was only one right hand side b_1.

By arranging the m right hand sides and the corresponding solutions in the matrices

$$B = [b_1|\cdots|b_m] \text{ and } X_j = [x_{1,j}|\cdots|x_{m,j}]$$

we can rewrite the systems (1.1) as

$$(\sigma_j I + A)X_j = B. \tag{1.2}$$

In the past Krylov subspace methods have been developed which exploit the composition of multiple right hand sides to blocks and solve each of the j systems in (1.2) in one go [9, 4]. It was realised that in these so called block methods deflation, i.e. removing those vectors that become linearly dependent, is important in order to guarantee convergence[7]. Other algorithms make use of multiple shifts for systems with a single right hand side and compute solutions for every shifted system by spanning the Krylov subspace just once [6]. For non-hermitian matrices there even exists a method that combines multiple right hand sides–including deflation–and shifted systems [5]. All these methods improved the computational costs for solving the kind of problem they are focused on.

We here present a new, deflated shifted block CG (DSBlockCG) method that merges every aspect: the block idea, the idea of computing solutions for the shifted systems alongside one seed system and the focus on hpd matrices. It is based on a block Lanczos process which is the restriction to the hermitian case of the non-symmetric block Lanczos process from [1]. This process is capable of handling multiple starting vectors and includes deflation.

2. Block Krylov subspace methods

For ease of presentation we first disregard the shifts, i.e. we deal with systems

$$Ax_i = b_i, i = 1,\ldots,m.$$

We define the *k-th block Krylov subspace* with respect to A and $B = [b_1|\cdots|b_m]$ as

$$\begin{aligned} K_k(A,B) &= \operatorname{span}\left\{b_1,\ldots,b_m,Ab_1,\ldots,Ab_m,A^2b_1,\ldots,A^{k-1}b_m\right\} \\ &=: \operatorname{colspan}\left[\, B \mid AB \mid A^2B \mid \cdots \mid A^{k-1}B \,\right]. \end{aligned} \tag{2.1}$$

Clearly, all the Krylov subspaces $K_k(A,b_i) := \operatorname{span}\left\{b_i,Ab_i,\ldots,A^{k-1}b_i\right\}$ are contained in $K_k(A,B)$. While the dimension of each space $K_k(A,b_i)$ is k (unless we have reached an invariant subspace), the dimension of the block subspace can be smaller than mk. This is due to the fact that some of the subspaces $K_k(A,b_i)$ can have non-trivial intersections even when all the b_i are linearly independent.

The block conjugate gradient methods in [9] and [4] create block iterates $X^{(k)} = [x_1^{(k)}|\ldots|x_m^{(k)}]$ with $x_i^{(k)} \in K_k(A,B)$ and advance in each step from $K_k(A,B)$ to $K_{k+1}(A,B)$. The iterates $X^{(k)}$ are obtained such that they satisfy the Galerkin condition

$$X^{(k)} \in K_k(A,B) \text{ and } R^{(k)} = B - AX^{(k)} \perp K_k(A,B),$$

which in the non-block case reduces to the classical variational characterisation of the CG iterates. Let $V^{(k)}$ denote a matrix whose columns form a basis of $K_k(A,B)$. Then the Galerkin condition is equivalent to

$$X^{(k)} = V^{(k)} \cdot \left((V^{(k)})^H A V^{(k)}\right)^{-1} \cdot (V^{(k)})^H B. \tag{2.2}$$

3. The deflated shifted block CG method

Based on the block Lanczos process[1] we can now explain the DSBlockCG method. We start by considering the unshifted block system $AX = B$. Handling additional shifts will be addressed at the end of this section. For details on the derivation and implementation of the algorithm we refer to our upcoming paper [2].

The block Lanczos process generates matrices $T^{(k)}$ and $V^{(k)}$. The orthonormal columns of the latter form a basis of a k-dimensional subspace which we will call the *k-th deflated block Krylov subspace* $K_k^{\text{defl}}(A,B)$. Additionally the relation

$$T^{(k)} = (V^{(k)})^H A V^{(k)}. \tag{3.1}$$

holds and $T^{(k)}$ is a hermitian, banded matrix with semi-bandwidth m. Note, that the dimension k of $K_k^{\text{defl}}(A,B)$ differs from the dimension of the non-deflated subspace $K_k(A,B)$ in (2.1) which is at most mk, but might be less if some vectors spanning $K_k(A,B)$ are linearly dependent. As soon as such deflation occurs, the bandwidth of the trailing right lower submatrix of $T^{(k)}$ decreases accordingly.

A deflated block Krylov subspace method generates iterates $X^{(k)} = [x_1^{(k)}|\ldots|x_m^{(k)}]$, s.t. $x_j^{(k)} \in K_k^{\text{defl}}(A,B)$. Building upon (2.2), but using the orthogonal basis of the deflated block Krylov subspace and (3.1), we obtain the iterates $X^{(k)}$ as

$$X^{(k)} := V^{(k)}(T^{(k)})^{-1}(V^{(k)})^H B. \tag{3.2}$$

In order to obtain a feasible iterative method, cheap updates for the iterates have to be obtained. By computing a root-free Cholesky decomposition $L^{(k)}D^{(k)}(L^{(k)})^H$ of the matrix $T^{(k)}$ we can update the iterates via

$$\begin{aligned} X^{(k)} &= V^{(k)}(L^{(k)}D^{(k)}(L^{(k)})^H)^{-1}(V^{(k)})^H B \\ &= X^{(k-1)} + \frac{1}{d^{(k)}} p^{(k)} u^{(k)} \end{aligned}$$

where $d^{(k)}$ is the (k,k)-entry in $D^{(k)}$, $p^{(k)} \in \mathbb{C}^n$ and $u^{(k)} \in \mathbb{C}^{1\times m}$. The important property for the feasibility of our method is that the Cholesky decomposition as well as $p^{(k)}$ and $u^{(k)}$ can be updated with short recurrences of length m or even less if deflation occurred.

In order to be able to stop the iteration we should be able to compute the norms of the residuals $r_j^{(k)} = b_j - Ax_j^{(k)}, 1 \le j \le m$. Fortunately, although the residuals are not directly available, their norms can be computed at very low additional cost. The iterates $X^{(k)}$ are generated in such a way, that the residuals $R^{(k)}$ are orthogonal to the Lanczos vectors $v^{(j)}$ for $j \le i-m$. They may thus be written as

$$R^{(k)} = W^{(k)}C^{(k)}$$

where $C^{(k)} \in \mathbb{C}^{m\times m}$ and $W^{(k)} = [v^{(k)}|\dots|v^{(i+m-1)}] \in \mathbb{C}^{n\times m}$. Thus, if we are only interested in the norm of each of the residuals we can just compute the norm of the columns of $C^{(k)} \in \mathbb{C}^{m\times m}$, because $W^{(k)}$ has orthonormal columns. Indeed, there is a computationally cheap way to compute $C^{(k)}$ using only the matrix $T^{(k)}$ and its Cholesky decomposition. If deflation occurred the number of columns of $W^{(k)}$ and rows of $C^{(k)}$ even decreases.

The block method so far can be extended to handle shifted systems and multiple right hand sides at the same time. For ease of notation we focus on a situation where we have just one additional shift σ and use the notation $A_\sigma := \sigma I + A$. Matrices and vectors belonging to the shifted system will also be noted by the index σ. Krylov subspaces are shift invariant, i.e. $K_k(A,b) = K_k(A_\sigma,b)$ for all k, see [10]. Moreover, starting with the initial vector b, the Lanczos process produces exactly the same vectors, whether we take A or A_σ. This property immediately carries over to the block Lanczos process and to the deflated block Lanczos process. We can make use of

$$(V^{(k)})^H(\sigma I_n + A)V^{(k)} = \sigma I_k + T^{(k)} =: T_\sigma^{(k)},$$

so that we do not have to compute $V^{(k)}$ and $T^{(k)}$ for each shifted system individually. Instead we can reuse these from the unshifted system $AX = B$ and only have to compute the Cholesky decomposition $L_\sigma^{(k)}$ and $D_\sigma^{(k)}$ as well as the vectors $p_\sigma^{(k)}$ and $u_\sigma^{(k)}$ for each shift.

4. Numerical results

In this section we present and discuss results of applying the algorithm from section 3 to some QCD test problems. We were working on non-parallelised algorithms, therefore our tests were restricted to rather small lattice sizes. A Chroma implementation of the algorithm is currently under development.

In order to obtain fair time measurements we implemented all of the algorithms in C++ using the uBLAS library [3] for sparse and dense BLAS operations. We stored the Wilson Dirac operator

D as a sparse matrix and did not exploit any further properties like its four dimensional lattice structure or symmetries. We chose a quenched configuration with temperature $\beta = 6.0$. As compiler we used g++ in version 4.5.1. The results were produced on a Core 2 Quad Q9650 running at 3.00 GHz. Since our programs were not parallelized they were running a single core of that machine.

We compare our algorithm to three other algorithms. The first of them is non-preconditioned conjugate gradients (CG). For CG we have to solve each of the systems

$$(\sigma_j I + A)x_{i,j} = b_i$$

separately. The second algorithm is the shifted conjugate gradients algorithm from [6] referred to as shiftedCG. It has to be applied m times, once for each right hand side b_i. For each right hand side all s shifted systems are solved at the same time. The third and last competing algorithm is BCGrQ from [4]. For each shift, this block algorithm solves all the systems belonging to the various right hand sides in one go.

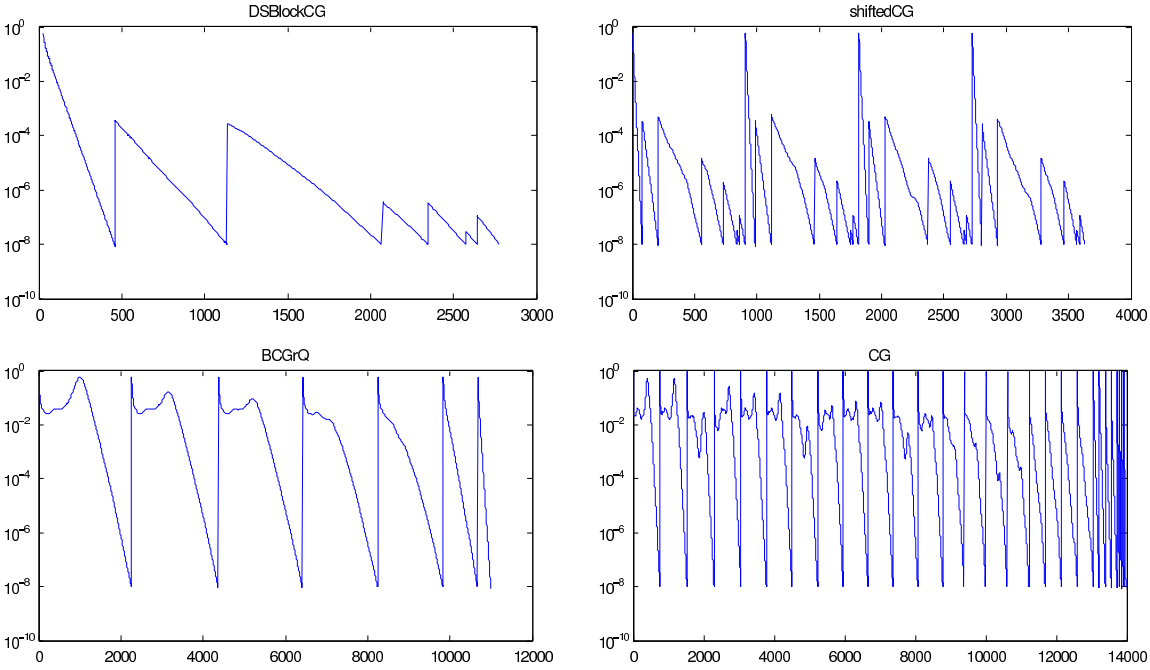

Figure 1: Convergence plots for all the compared algorithms on a 16^4 lattice with 4 right hand sides and 7 shifts. Horizontal axis: time in seconds. Vertical axis: relative norm of residual. The saw tooth shape of the plots has two causes: a) for DSBlockCG and shiftedCG only the residual for the worst conditioned non-converged system is computed and plotted. b) methods that can not solve all of the systems at the same time need subsequent runs which are plotted one after the other.

Figure 1 shows a generic run of all four algorithms for a 16^4 lattice to a target relative residual of 10^{-8}. We chose the same 4 random right hand sides for all methods. The 7 shifts were chosen, s.t. the smallest eigenvalue of the worst conditioned shifted system was of magnitude 10^{-6} and for the best conditioned system it was of magnitude 10^{-1}. DSBlockCG in the top left plot shows a saw tooth like convergence. This is caused by just computing the residual for the best conditioned not yet converged system which saves some work. Every time the plot of the relative residual

jumps up, all the shifted systems for one right hand side have converged and the residuals for next non-converged system are computed. Thus, this plot shows that it is important to stop updating the iterates of well conditioned systems as soon as they reach the target residual. This speeds up the computation of the remaining systems. The top right shiftedCG plot shows that the different random right hand side vectors show almost the same convergence behaviour. The bottom two plots show clearly that CG and BCGrQ are not competitive. Table 1 gives the plain numbers for the same test run and shows that shiftedCG takes about 30% more time than DSBlockCG. The gap in the number of matrix-vector multiplications (mvms) is even bigger. As a result of Figure 1 we can constrain the remaining tests to comparisons of DSBlockCG and shiftedCG.

	CG	BlockCG	shifted CG	DSBlockCG
mvms	22337	13488	4883	2766
time in seconds	13970.7	10993.3	3630.53	2778.82
relative time	5.03	3.96	1.31	1

Table 1: Number of mvms, time and relative time as compared to DSBlockCG.

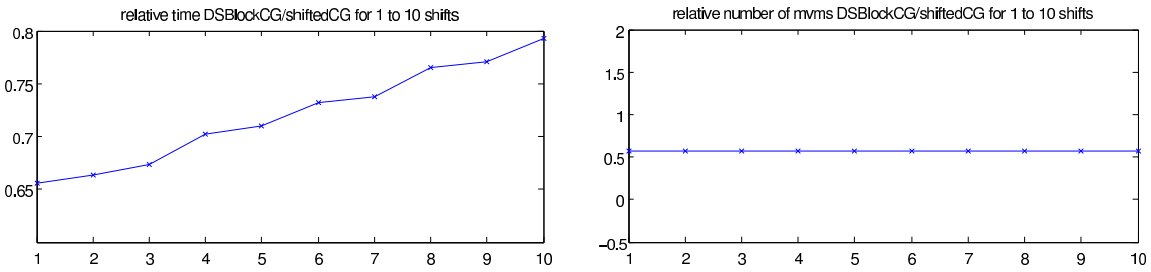

Figure 2: The left plot shows the relative time (vertical axis) of algorithm DSBlockCG over shiftedCG on a 16^4 lattice for a different number of shifts (horizontal axis) and a fixed number of 4 rhs. The right plot displays the relative number of mvms for the same run.

Figure 2 displays the dependence on the number of shifts while the number of right hand sides stayed fixed at 4. The results were produced using the same 16^4 lattice configuration as in the previous example. We chose the first shift, s.t. the smallest eigenvalue of the shifted system was of magnitude 10^{-6}. We then successively increased the number of shifts, s.t. the smallest eigenvalues of the shifted systems were distributed in the interval $[10^{-6}, 10^{-1}]$. The plots show that the number of mvms depends only on the worst conditioned system which means that for any number of shifts DSBlockCG only needs about half the number of mvms compared to shiftedCG. The computational costs on the other hand are rising for an increased number of shifts.

In Figure 3 we display a test run for another test configuration on a 12^4 lattice. In this case the number of 10 shifts stayed fixed. Like in the previous test cases the shifts were chosen, s.t. the smallest eigenvalue of the worst conditioned shifted system was of magnitude 10^{-6} and for the best conditioned system it was of magnitude 10^{-1}. There is a sweet spot at 4 shifts after which the additional costs start to become dominant and DSBlockCG becomes slower than shiftedCG, eventually.

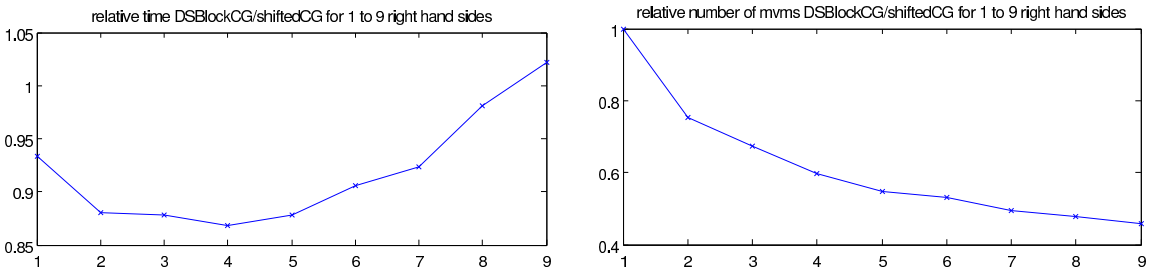

Figure 3: The left plot shows the relative time (vertical axis) of algorithm DSBlockCG over shiftedCG on a 12^4 lattice for a different number of right hand sides (horizontal axis) and fixed number of 10 shifts. The right plot displays the relative number of mvms for the same run.

5. Conclusions

We proposed a new iterative method for the solution of systems of linear equations of the form $(\sigma_j I + A)x_{i,j} = b_k$ based on a block Lanczos-type process. This method was shown to converge faster than just applying CG to every system or applying block CG methods to systems belonging to a single shift. For reasonable numbers of right hand sides and shifts our new method even proved to be faster that shiftedCG.

References

[1] J. I. Aliaga, D. L. Boley, R. W. Freund, and V. Hernández. A Lanczos-type method for multiple starting vectors. *Math. Comp.*, 69(232):1577–1601, 2000.

[2] S. Birk and A. Frommer. A deflated conjugate gradient method for multiple right hand sides and multiple shifts. In preparation.

[3] Boost. Boost C++ libraries. http://www.boost.org, 2010. Version 1.44.0.

[4] A. A. Dubrulle. Retooling the method of block conjugate gradients. *Electron. Trans. Numer. Anal.*, 12:216–233 (electronic), 2001.

[5] P. Fiebach, A. Frommer, and R. Freund. Variants of the Block-QMR method and applications in quantum chromodynamics. In *15th IMACS World Congress on Scientific Computation, Modelling and Applied Mathematics*, volume 3, pages 491–496, 1997.

[6] A. Frommer and P. Maass. Fast CG-based methods for Tikhonov-Phillips regularization. *SIAM J. Sci. Comput.*, 20(5):1831–1850 (electronic), 1999.

[7] M. H. Gutknecht. *Modern Mathematical Models, Methods and Algorithms for Real World Systems*, chapter Block Krylov Space Methods for Linear Systems With Multiple Right-hand Sides: an Introduction, pages 420–447. Anamaya Publishers, New Delhi, India, 2007.

[8] A. D. Kennedy. Algorithms for dynamical fermions. 2006. arXiv:hep-lat/0607038v1.

[9] D. P. O'Leary. The block conjugate gradient algorithm and related methods. *Linear Algebra Appl.*, 29:293 – 322, 1980.

[10] C. C. Paige, B. N. Parlett, and H. A. van der Vorst. Approximate solutions and eigenvalue bounds from Krylov subspaces. *Numer. Linear Algebra Appl.*, 2(2):115–133, 1995.