github.com/lkeegan/blockCG

Liam Keegan
liam@keegan.ch

July 10, 2018

1 Introduction

This document contains a description of the algorithms implement in the github repo, and references to the original constructions. They are all variants of CG inverters, where the inversion may include N_{shifts} shifts and act simultaneously on n_{RHS} vectors. Notation conventions used:

- $\mathbf{A}$ is the hermitian positive definite $L \times L$ matrix to be inverted
- x, r, p, t, etc. are L–component complex vectors
- X, R, P, T, etc. are $L \times n_{\text{RHS}}$ complex block–vectors
- α, β, δ, ζ, θ, etc. are $n_{\text{RHS}} \times n_{\text{RHS}}$ complex matrices
- σ is a positive real scalar shift

The norm used here is defined for a vector x as

$$|x| \equiv \left(x^\dagger x\right)^{1/2} \tag{1.1}$$

and similarly for each vector i within a block–vector X,

$$|X|_i \equiv |X_i| = \left(X_i^\dagger X_i\right)^{1/2} \tag{1.2}$$

2 Solvers

2.1 CG

The standard CG solver for $\mathbf{A}x = b$, with initial guess $x_0 = 0$ and criterion for convergence that the relative norm of the residual is smaller than some number ϵ, i.e. $|\mathbf{A}x_k - b|/|b| < \epsilon$ is given in Alg. 1.

Algorithm 1 CG: Solve $\mathbf{A}x = b$

$x, p, r, t \in \mathcal{C}^L$; $\alpha, \beta \in \mathcal{R}$
$x_0 = t_0 = 0$, $r_0 = p_0 = b$
for $k = 1, 2, \ldots$ until $|r_k|/|r_0| < \epsilon$ **do**
    $t_k \leftarrow \mathbf{A}p_{k-1}$
    $\alpha_k \leftarrow (p_{k-1}^\dagger t_k)^{-1}(r_{k-1}^\dagger r_{k-1})$
    $r_k \leftarrow r_{k-1} - t_k\alpha_k$
    $\beta_k \leftarrow (r_{k-1}^\dagger r_{k-1})^{-1}(r_k^\dagger r_k)$
    $x_k \leftarrow x_{k-1} + p_{k-1}\alpha_k$
    $p_k \leftarrow r_k + p_{k-1}\beta_k$
end for

2.2 SCG

Since the Krylov basis is shift–invariant, the residual r_k^σ of the solution of the shifted equation $(\mathbf{A} + \sigma)x_k^\sigma = b$, where σ is a real and positive scalar, can be related to the residual of the unshifted equation, $r_k^\sigma = r_k\zeta_k^\sigma$. This allows the construction of a shifted CG (SCG) solver [1], which can solve multiple shifts wihout additional applications of the matrix $\mathbf{A}$, as described in Alg. 2. Here for convenience we introduce the variable $\theta_k^{\sigma_j} \equiv \zeta_k^{\sigma_j}/\zeta_{k-1}^{\sigma_j}$. The relative residual for a given shift, $\zeta_k^{\sigma_j}|r_k|/|r_0|$, can be monitored, and once it is smaller than machine precision one can stop updating the shifted solution. For the case $N_{\text{shifts}} = 1$ this solver reduces to the CG solver of Alg. 1.

Algorithm 2 SCG: Solve $(\mathbf{A} + \sigma_j)x^{\sigma_j} = b$, where $j = 0, \ldots, N_{\text{shifts}} - 1$

$x^{\sigma_j}, p^{\sigma_j}, r, t \in \mathcal{C}^L$; $\alpha, \beta, \zeta^{\sigma_j}, \theta^{\sigma_j} \in \mathcal{R}$
$x_0^{\sigma_j} = t_0 = 0$, $r_0 = p_0^{\sigma_j} = b$; $\beta_0 = 0$, $\alpha_0 = \zeta_0^{\sigma_j} = \theta_0^{\sigma_j} = 1$
for $k = 1, 2, \ldots$ until $|r_k|/|r_0| < \epsilon$ **do**
    $t_k \leftarrow (\mathbf{A} + \sigma_0)p_{k-1}^{\sigma_0}$
    $\alpha_k \leftarrow ((p_{k-1}^{\sigma_0})^\dagger t_k)^{-1}(r_{k-1}^\dagger r_{k-1})$
    $r_k \leftarrow r_{k-1} - t_k\alpha_k$
    $\beta_k \leftarrow (r_{k-1}^\dagger r_{k-1})^{-1}(r_k^\dagger r_k)$
    $x_k^{\sigma_0} \leftarrow x_{k-1}^{\sigma_0} + p_{k-1}^{\sigma_0}\alpha_k$
    $p_k^{\sigma_0} \leftarrow r_k + p_{k-1}^{\sigma_0}\beta_k$
    for $j = 1, \ldots, N_{\text{shifts}} - 1$ **do**
        $\theta_k^{\sigma_j} \leftarrow \left[1 + (\sigma_j - \sigma_0)\alpha_k + \beta_{k-1}(\alpha_k/\alpha_{k-1})(1 - \theta_{k-1}^{\sigma_j})\right]^{-1}$
        $\zeta_k^{\sigma_j} \leftarrow \theta_k^{\sigma_j}\zeta_{k-1}^{\sigma_j}$
        $x_k^{\sigma_j} \leftarrow x_{k-1}^{\sigma_j} + p_{k-1}^{\sigma_j}\alpha_k\theta_k^{\sigma_j}$
        $p_k^{\sigma_j} \leftarrow r_k\zeta_k^{\sigma_j} + p_{k-1}^{\sigma_j}\beta_k(\theta_k^{\sigma_j})^2$
    end for
end for

3 Block Solvers

3.1 BCG

The BlockCG (BCG) [2] algorithm is a straightforward extension of the CG algorithm to multiple RHS vectors, described in Alg. 3. For the case $n_{\text{RHS}} = 1$ this solver reduces to the CG solver of Alg. 1. Note that this solver tends to fail to converge when the matrix of residuals R becomes badly conditioned.

Algorithm 3 BCG: Solve $\mathbf{A}X = B$

$X, P, R, T \in \mathcal{C}^{L \times n_{\mathrm{RHS}}}$; $\alpha, \beta \in \mathcal{C}^{n_{\mathrm{RHS}} \times n_{\mathrm{RHS}}}$
$X_0 = T_0 = 0$, $P_0 = R_0 = B$
for $k = 1, 2, \ldots$ until $|R_k|_i / |R_0|_i < \epsilon \, \forall i$ **do**
    $T_k \leftarrow \mathbf{A} P_k$
    $\alpha_k \leftarrow (P_k^\dagger T_k)^{-1} (R_{k-1}^\dagger R_{k-1})$
    $R_k \leftarrow R_{k-1} - T_k \alpha_k$
    $\beta_k \leftarrow (R_{k-1}^\dagger R_{k-1})^{-1} (R_k^\dagger R_k)$
    $X_k \leftarrow X_{k-1} + P_{k-1} \alpha_k$
    $P_k \leftarrow R_k + P_{k-1} \beta_k$
end for

3.2 BCGrQ

This is a variant of the BCG algorithm where the matrix of residuals R_k is orthogonalised at each step, $Q_k \delta_k = R_k$, where Q_k is orthornormal and R_k is upper triangular. This algorithm is known as BCGrQ [3], described in Alg. 4, and it has significantly improved convergence properties compared to BCG.

Algorithm 4 BCGrQ: Solve $\mathbf{A}X = B$

$X, P, Q, T \in \mathcal{C}^{L \times n_{\mathrm{RHS}}}$; $\alpha, \rho, \delta \in \mathcal{C}^{n_{\mathrm{RHS}} \times n_{\mathrm{RHS}}}$
$X_0 = T_0 = 0$, $\{Q_0, \delta_0\} = \mathrm{qr}(B)$, $P_0 = R_0$
for $k = 1, 2, \ldots$ until $\sqrt{\sum_j \delta_k(i,j) / \sum_j \delta_0(i,j)} < \epsilon \, \forall i$ **do**
    $T_k \leftarrow \mathbf{A} P_{k-1}$
    $\alpha_k \leftarrow (P_{k-1}^\dagger T_k)^{-1}$
    $\{Q_k, \rho_k\} \leftarrow \mathrm{qr}(Q_{k-1} - T_k \alpha_k)$
    $X_k \leftarrow X_{k-1} + P_{k-1} \alpha_k \delta_{k-1}$
    $P_k \leftarrow Q_k + P_{k-1} \rho_k^\dagger$
    $\delta_k \leftarrow \rho_k \delta_{k-1}$
end for

4 Shifted Block Solvers

4.1 SBCGrQ

The BCGrQ solver can be extended to solve multiple shifts in the same way as the CG solver was extended to the SCG solver, by relating the residuals of the shifted system to the unshifted ones. This leads to the SBCGrQ solver [4], described in Alg. 5. The formulation here is equivalent but uses a coupled two–term recursion in $\alpha_k^{\sigma_j}$ and $\beta_k^{\sigma_j}$, instead of the single three-term recursion in $\zeta_k^{\sigma_j}$, which can improve the numerical accuracy of the shifted solutions if the system is very badly conditioned [**?**]. The relative norm of the shifted residual is given by $\sqrt{\sum_l \delta_k^{\sigma_j}(i,l) / \sum_l \delta_0(i,l)}$, where $\delta_k^{\sigma_j} = \rho_k \alpha_k^{-1} \alpha_k^{\sigma_j}$.

Algorithm 5 SBCGrQ: Solve $(\mathbf{A}+\sigma_j)X^{\sigma_j}=B$, where $j=0,\ldots,N_{\text{shifts}}-1$

$X^{\sigma_j}, P^{\sigma_j}, Q, T \in \mathcal{C}^{L\times n_{\text{RHS}}}$; $\alpha,\rho,\delta,\alpha^{\sigma_j},\beta^{\sigma_j}\in\mathcal{C}^{n_{\text{RHS}}\times n_{\text{RHS}}}$
$X_0=T_0=0$, $\{Q_0,\delta_0\}=\text{qr}(B)$, $P_0=Q_0$; $\rho_0=\delta_0$, $\alpha_0=\alpha_0^{\sigma_j}=\beta_0^{\sigma_j}=1$
for $k=1,2,\ldots$ until $\sqrt{\sum_j \delta_k(i,j)/\sum_j \delta_0(i,j)}<\epsilon\,\forall i$ **do**
    $T_k \leftarrow \mathbf{A}P_{k-1}$
    $\alpha_k \leftarrow (P_{k-1}^\dagger T_k)^{-1}$
    $\{Q_k,\rho_k\}\leftarrow \text{qr}(Q_{k-1}-T_k\alpha_k)$
    $X_k \leftarrow X_{k-1}+P_{k-1}\alpha_k\delta_{k-1}$
    $P_k \leftarrow Q_k+P_{k-1}\rho_k^\dagger$
    $\delta_k\leftarrow\rho_k\delta_{k-1}$
    for $j=1,\ldots,N_{\text{shifts}}-1$ **do**
        $\beta_k^{\sigma_j}\leftarrow\left[1+(\sigma_j-\sigma_0)\alpha_k+\alpha_k\rho_{k-1}\alpha_{k-1}^{-1}\left\{1-\beta_{k-1}^{\sigma_j}\right\}\rho_{k-1}^\dagger\right]^{-1}$
        $\alpha_k^{\sigma_j}\leftarrow\beta_k^{\sigma_j}\alpha_k\rho_{k-1}\alpha_{k-1}^{-1}\alpha_{k-1}^{\sigma_j}$
        $X_k^{\sigma_j}\leftarrow X_{k-1}^{\sigma_j}+P_{k-1}^{\sigma_j}\alpha_k^{\sigma_j}$
        $P_k^{\sigma_j}\leftarrow Q_k+P_{k-1}^{\sigma_j}\beta_k^{\sigma_j}\rho_k^\dagger$
    end for
end for

References

[1] B. Jegerlehner, *Krylov space solvers for shifted linear systems*, hep-lat/9612014.

[2] D. P. O'Leary, *The block conjugate gradient algorithm and related methods*, *Linear Algebra and its Applications* **29** (1980) 293 – 322. Special Volume Dedicated to Alson S. Householder.

[3] A. A. Dubrulle, *Retooling the method of block conjugate gradients*, *Electronic Trans. on Num. Anal* (2001) 216–233, [http://eudml.org/doc/121814].

[4] Y. Futamura, T. Sakurai, S. Furuya, and J.-I. Iwata, *Efficient algorithm for linear systems arising in solutions of eigenproblems and its application to electronic-structure calculations*, in *High Performance Computing for Computational Science - VECPAR 2012*, (Berlin, Heidelberg), pp. 226–235, Springer Berlin Heidelberg, 2013.