Deflated CG

Zoé Weissbaum

May 2024

a)

Our goal is to solve a linear system of the form $Ax = b$, where $A \in \mathbb{R}^{n\times n}$ is a symmetric positive definite matrix and $b \in \mathbb{R}^n$. The conjugate gradient (CG) method (see section 4.1.2 and Algorithm 4.5 in [2]) is an iterative method which can be used to solve these types of linear systems. But the problem with CG is that its speed of convergence depends on the condition number of A (see Theorem 4.6 in [2]) and hence on the distribution of its eigenvalues. If the smallest eigenvalue of A is close to zero, its condition number is large and CG will converge very slowly. With the deflated-CG method, we are aiming to increase the convergence speed of the CG method by adding to the Krylov subspace a deflation subspace: we add to the Krylov subspace a few approximate eigenvectors associated with eigenvalues closest to zero. This is intended as a way to "hide" these eigenvalues to the CG method so that the convergence speed will increase.

Let $W = \{w_1, \dots, w_k\}$ be the block of k vectors we add into the Krylov subspace. We assume that they are linearly independent. Since A is symmetric positive definite, we note that the matrix $W^\top AW$ is non-singular. Let x_0 be an initial guess such that $r_0 = b - Ax_0 \perp W$. As in the CG method, we want to extract an approximate solution from the Krylov subspace that satisfies a Galerkin condition, that is find $x_j \in x_0 + \mathcal{K}_{k,j}(A, W, r_0)$ such that $r_j = b - Ax_j \perp \mathcal{K}_{k,j}(A, W, r_0)$. The Krylov subspace $\mathcal{K}_{k,j}(A, W, r_0)$ is the space spanned by the vectors $\{w_1, ..., w_k\}$ and the basis of $\mathcal{K}_j(A, r_0)$. Then the equations in (3.10) and (3.11) from [3] provide the steps of the Deflated-CG algorithm (Algorithm 3.5 in [3]), which is essentially the same as the CG algorithm except from the parts with W and $\hat{\mu}_j$.

The last step to get Algorithm 3.6 is to add preconditioning. Let $M = LL^\top \in \mathbb{R}^{n\times n}$ be an invertible matrix. The preconditioned Deflated-CG algorithm is the Deflated-CG method with split preconditioning. Instead of solving the linear system $Ax = b$, we solve the system $L^{-1}AL^{-\top}y = L^{-1}b$, with $x = L^{-\top}y$. The goal of preconditioning is to accelerate convergence and since Theorem 4.3 in [3] shows that the convergence depends on the condition number, we aim to choose a matrix L such that $\kappa(L^{-1}AL^{-\top}) < \kappa(A)$. Finally, by applying the Deflated-CG algorithm to $L^{-1}AL^{-\top}y = L^{-1}b$ and redefining the variables as follows:

$$\begin{aligned} L^{-\top}W &\to W, \qquad & Lr_j &\to r_j, \\ L^{-\top}y_j &\to x_j, \qquad & L^{-\top}p_j &\to p_j, \end{aligned}$$

we obtain Algorithm 3.6 in [3].

b)

Theorem 1. *Let κ be the condition number of $H^\top AH$, where $H = I - W(W^\top AW)^{-1}(AW)^\top$. Then $||x_* - x_j||_A \leq 2\left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^j ||x_* - x_0||_A$.*

Proof. We follow similar ideas as for the convergence of CG that we have seen during the lectures (Theorem 4.6 in [2]). We have that $r_j = b - Ax_j \perp \mathcal{K}_{k,j}(A, W, r_0)$ by construction of the residuals r_j. Then, since $b = Ax_*$, we can rewrite this as $A(x_* - x_j) \perp \mathcal{K}_{k,j}(A, W, r_0)$, meaning that the error $x_* - x_j$ is A-orthogonal to $\mathcal{K}_{k,j}(A, W, r_0)$. Thus we have

$$\begin{aligned} ||x_* - x_j||_A &= \min_{u \in \mathcal{K}_{k,j}(A,W,r_0)} ||x_* - x_0 - u||_A \\ &\leq \min_{u \in \mathcal{K}_j(H^\top AH, r_0)} ||x_* - x_0 - u||_A, \end{aligned}$$

because the minimum over $\mathcal{K}_{k,j}(A, W, r_0)$ is lower than the minimum over $\mathcal{K}_j(H^\top AH, r_0)$, due to Theorem 3.3 in [1]. Then, using Lemma 2.21 in the lecture notes, we get

$$\begin{aligned} ||x_* - x_j||_A &\leq \min_{p \in \Pi_{j-1}} ||x_* - x_0 - p(H^\top AH) r_0||_A \\ &= \min_{p \in \Pi_{j-1}} ||x_* - x_0 - (H^\top AH) p(H^\top AH)(x_* - x_0)||_A \\ &= \min_{\tilde{p} \in \Pi_j,\ \tilde{p}(0)=1} ||\tilde{p}(H^\top AH)(x_* - x_0)||_A \\ &\leq ||x_* - x_0||_A \min_{\tilde{p} \in \Pi_j,\ \tilde{p}(0)=1} ||\tilde{p}(H^\top AH)||_2 \\ &= ||x_* - x_0||_A \min_{\tilde{p} \in \Pi_j,\ \tilde{p}(0)=1} \max_{\lambda \in \{\lambda_1, \ldots, \lambda_n\}} |\tilde{p}(\lambda)| \\ &\leq ||x_* - x_0||_A \min_{\tilde{p} \in \Pi_j,\ \tilde{p}(0)=1} \max_{\lambda \in [\lambda_{\min}, \lambda_{\max}]} |\tilde{p}(\lambda)|. \end{aligned}$$

where, in the last inequality, we have replaced the discrete set $\{\lambda_1, \ldots, \lambda_n\}$ by the interval $[\lambda_{\min}, \lambda_{\max}]$. Lastly, the $\tilde{p}$ that solves the optimization problem in the last line takes the form of a scaled and shifted Chebyshev polynomial and thus we obtain

$$||x_* - x_j||_A \leq 2\left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^j ||x_* - x_0||_A.$$

□

c)

The idea of Algorithm 5.3 is to use the Deflated PCG algorithm (Algorithm 3.6 in [3]) to solve linear systems that feature the same symmetric positive definite matrix A but different right-hand sides:

$$Ax^{(s)} = b^{(s)},\ s = 1, 2, \ldots, \nu.$$

In order to achieve this, the first system is simply solved by using standard preconditioned CG. This corresponds to step 2 of Algorithm 5.3. In step 1, we define l to be the number of steps of PCG for which we keep the data $(\alpha_i, \beta_i, \hat{\mu}_i, p_i)$. Then for each of the other systems, a few approximate eigenvectors are added into the Krylov subspace. The number of eigenvectors is

given by k. The eigenvectors are refined with each new system being solved. In this way, convergence is expected to be improved as more systems are solved. To compute the approximate eigenvectors, we have to solve a generalized eigenvalue problem $G^{(s)}y_i - \theta_i F^{(s)}y_i = 0$. The matrices $F^{(s)}$ and $G^{(s)}$ are computed based on A, W and the data of the l steps of Algorithm 3.6 (or PCG if $s = 1$) used to solve the previous system, so we indeed get an approximation of the eigenvectors. In practice, using relations (5.2) and (5.4) in [3], we only need to keep p_i instead of $(\alpha_i, \beta_i, \hat{\mu}_i, p_i)$ for $i = 1, \ldots, l$ to compute $F^{(s)}$ and $G^{(s)}$. With the solution to this eigenvalue problem, we can compute the new W, which we use in Algorithm 3.6 to solve the new system. Finally, we compute the new $G^{(s)}, F^{(s)}$. Thus we obtain the following algorithm:

Algorithm 5.3: Deflated PCG for multiple right-hand sides.
Input: $A \in \mathbb{R}^{n\times n}$ symmetric positive definite, $B = [b^{(1)}, \ldots, b^{(\nu)}] \in \mathbb{R}^{n\times\nu}$ the matrix of right-hand sides, k the number of eigenvectors to deflate with, $l \geq k$ the number of PCG steps for which we keep the data, M a preconditioner.
Output: $X = [x^{(1)}, \ldots, x^{(\nu)}] \in \mathbb{R}^{n\times\nu}$, each column of which is a solution to $Ax^{(s)} = b^{(s)}$.

1. Solve $Ax^{(1)} = b^{(1)}$ with standard PCG, keep the data of the first l steps.
2. Compute $G^{(1)}, F^{(1)}$ according to their definitions. Set $W^{(1)} = \emptyset$.
3. For $s = 2, \ldots, \nu$:
4. Solve $G^{(s-1)}y_i - \theta_i F^{(s-1)}y_i = 0$ for k eigenvectors $y_1^{(s-1)}, \ldots, y_k^{(s-1)}$.
5. Set $W^{(s)} = [W^{(s-1)}, P_l(s-1)]Y^{(s-1)}$ with $P_l^{(s-1)} = [p_0^{(s-1)}, \ldots, p_{l-1}(s-1)]$ the search directions computed in Algorithm 3.6 (or PCG if $s = 2$).
6. Solve $Ax^{(s)} = b^{(s)}$ with Algorithm 3.6 with $W = W^{(s)}$, keep the data of the first l steps.
7. Compute $G^{(s)}, F^{(s)}$ according to their definitions.

d)

We reproduce the observations of Example 1 in [3]. The goal of this example is to investigate the numerical convergence behavior of Algorithm 3.6 explained in part a) and b). The matrix A is constructed using a 5-point centered difference discretization of a Laplacian on a $(N + 2)\times(N+2)$ mesh, where N represents the number of points in each direction. Thus the matrix A is a $n \times n$ matrix with $n = N^2$. The vector b is randomly generated with independent and identically distributed entries from the standard normal distribution.

We will analyse three scenarios with varying values of n: $n = 400$ (corresponding to $N = 20$), $n = 1600$ (corresponding to $N = 40$), and $n = 4624$ (corresponding to $N = 68$).

For each n, we start by computing three eigenvectors w_1, w_2, and w_3, associated to the three smallest eigenvalues of the matrix A. We then perform the Deflated-CG method and compare it with the standard CG method. For the Deflated-CG method, we use an initial guess of $x_{-1} = 0$ and compare three different deflation subspace choices: $W = [w_1]$, $W = [w_1, w_2]$, and $W = [w_1, w_2, w_3]$. For the CG method, we use an initial guess of $x_0 = 0$. To analyse the results, we plot the relative residuals $||b - Ax_j||_2/||b||_2$ versus the number of iterations. We terminate when the residual is less than 10^{-7}.

For the first case, $n = 400$ and $N = 20$, the largest eigenvalue of the matrix A is 7.9553 and the four smallest eigenvalues are 0.0447, 0.1112, 0.1112 and 0.1777. The plot can be seen on Figure 1. As expected, Deflated-CG converges in less iterations than CG. We obtain the best convergence when using all three smallest eigenvectors w_1, w_2, w_3. Convergence with $W = [w_1, w_2]$ or $W = [w_1]$ is almost identical, and standard CG is the slowest. These observations can be explained from the condition number of the systems solved by these different methods. The condition number of the matrix A is $\kappa_A = \frac{7.9553}{0.0447} = 178.0643$. On the other hand, as we have explained in part a) and from the Theorem proved in part b), the Deflated-CG with $W = [w_1]$, $W = [w_1, w_2]$, and $W = [w_1, w_2, w_3]$ solves a system with condition numbers $\kappa_1 = \frac{7.9553}{0.1112} = 71.5454$, $\kappa_2 = \frac{7.9553}{0.1112} = 71.5454$ and $\kappa_3 = \frac{7.9553}{0.1777} = 44.7661$ respectively. A smaller condition number then implies a faster convergence.

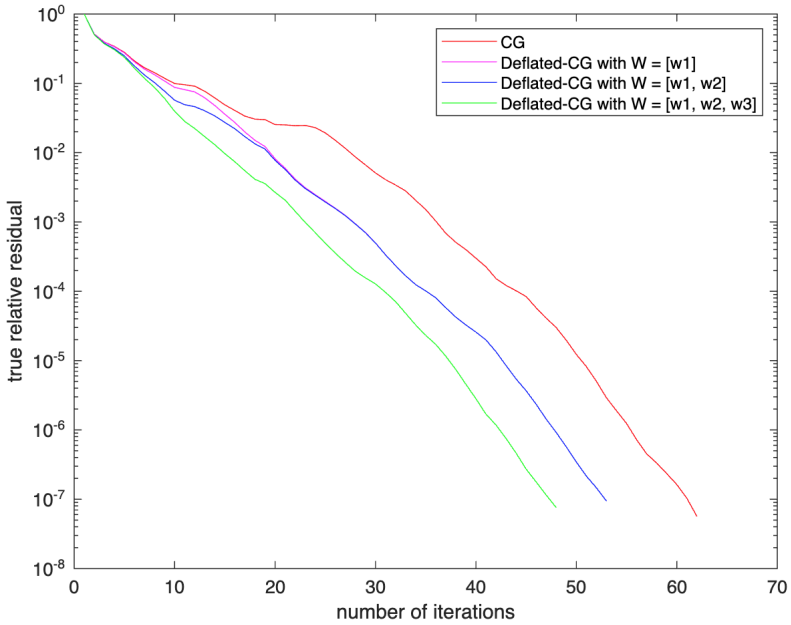

Figure 1: Convergence of Deflated Conjugate Gradient method, n = 400

For the two other cases, $n = 1600$ (Figure 2) and $n = 4624$ (Figure 3), we observe analogous results. The matrix A has a similar structure and hence a similar eigenvalue distribution, except it is bigger, so its condition number is larger. This leads to an increase in the number of iterations necessary to reach convergence, but the general behaviours and comparison between the four computations are equivalent to the first case.

More specifically, in the case $n = 1600$ and $N = 40$, the largest eigenvalue of the matrix A is 7.9883 and the four smallest eigenvalues are 0.0117, 0.0293, 0.0293 and 0.0469. This results in systems with condition numbers $\kappa_A = \frac{7.9883}{0.0117} = 680.6171$, $\kappa_1 = \kappa_2 = \frac{7.9883}{0.0293} = 272.5667$ and $\kappa_3 = \frac{7.9883}{0.0469} = 170.4043$.

Finally, in the case $n = 4624$ and $N = 68$, the largest eigenvalue of the matrix A is 7.9959 and the four smallest eigenvalues are 0.0041, 0.0104, 0.0104 and 0.0166. The systems then have condition numbers $\kappa_A = \frac{7.9883}{0.0117} = 1928.9$, $\kappa_1 = \kappa_2 = \frac{7.9883}{0.0104} = 771.8776$ and $\kappa_3 = \frac{7.9883}{0.0166} = 482.4735$.

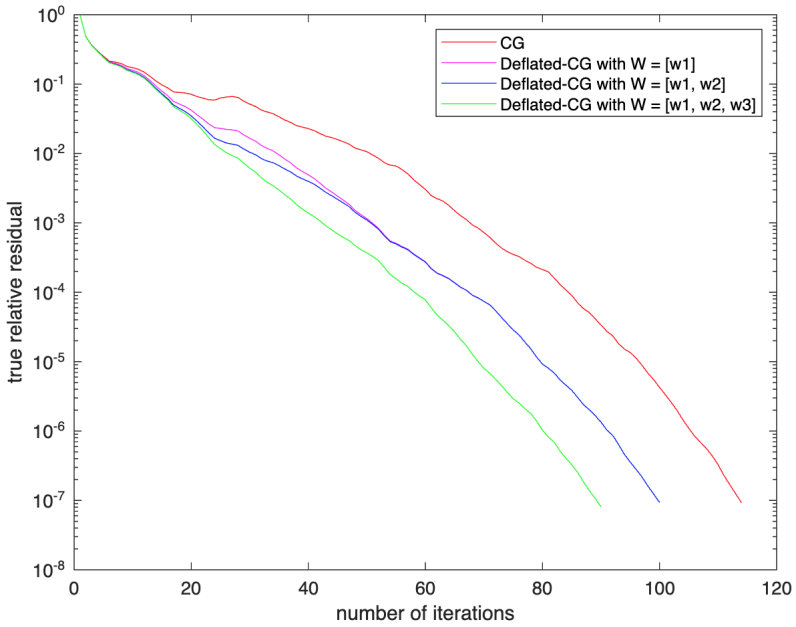

Figure 2: Convergence of Deflated Conjugate Gradient method, n = 1600

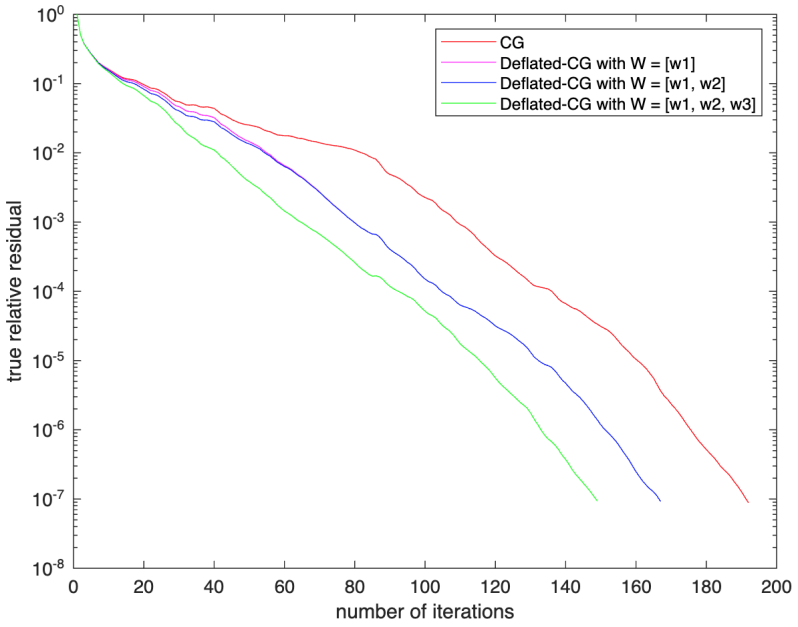

Figure 3: Convergence of Deflated Conjugate Gradient method, n = 4624

e)

We reproduce the observations of Example 3 instead of Example 2, because the $ic(0)$ preconditioner does not work for Example 2 and we do not have access to the incomplete Cholesky factorization $ic(1)$ in Matlab. The goal of this example is to study the numerical convergence behaviour of Algorithm 5.3.

We are using the matrix 1138bus which is a sparse 1138×1138 matrix. We precondition the

system by incomplete Cholesky factorization $ic(0)$. We use $\nu = 10$ different right-hand sides which are independent random vectors with entries from the standard normal distribution. We take $l = 20$ steps and $k = 5$ approximate eigenvectors. As in Example 1, we plot the relative residuals $||b^{(s)} - Ax_j^{(s)}||_2/||b^{(s)}||_2$ versus the number of iterations for each s and terminate when the residual is less than 10^{-7}.

The results can be viewed on Figure 4. The convergence of the first system, computed with standard PCG, is plotted in red. For the other 9 systems $s = 2, \ldots, 10$, which are solved with Deflated-CG, we used blue lines. The second system converges a bit faster and then we observe a large gap between the second and the third, and all remaining systems are solved in approximately the same number of iterations. This is because the approximate eigenvectors quickly reach their limits. The plot is slightly different than the one in [3], where the approximate eigenvectors do not reach their limits as fast. This can be due to the fact that we used the function `eigs` in Matlab, as suggested by our supervisor, instead of the QZ algorithm, to solve the generalized eigenvalue problem (step 4 in the algorithm in c)). But the general behaviour we observe is still very similar to what is mentioned in [3]. Once the approximate eigenvectors are no longer refined, convergence speed no longer increases.

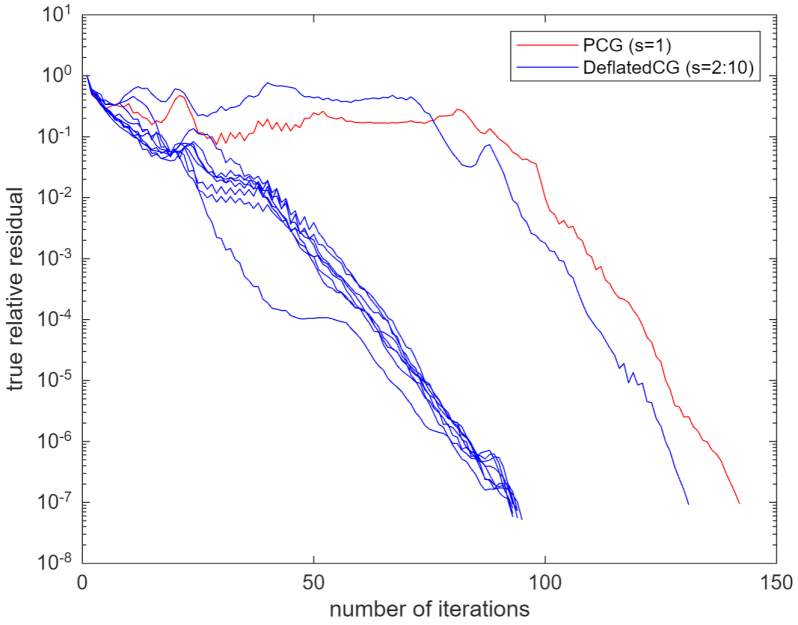

Figure 4: Convergence of Deflated-CG for multiple right-hand sides

References

[1] J. Erhel and F. Guyomarc'h. *An Augmented Subspace Conjugate Gradient.* Reasearch report RR-3278 INRIA, Domaine de Voluceau, Rocquencourt, B.P. 105, 78150 Le Chesnay, France, 1997.

[2] Daniel Kressner. *Computational Linear Algebra.* Lecture Notes, 2024.

[3] Y. Saad, M. Yeung, and F. Guyomarc'h. *A deflated version of the conjugate gradient algorithm.* SIAM J. Sci. Comput. 21 (2000), no. 5, 1909-1926.