

IDR(s)

Master's thesis
Goushani Kisoensingh

Supervisor: Gerard L.G. Sleijpen
Department of Mathematics
Universiteit Utrecht

Contents

1	Introduction	2
2	The background of Bi-CGSTAB	3
3	IDR(s)	4
3.1	IDR	5
3.2	IDR(s)	6
3.3	Initiating the matrix \mathbf{U} for the IDR(s) algorithm	7
3.3.1	Initial matrices $\mathbf{U} = \mathbf{r}$ and $\mathbf{U} = \text{Arnoldi}$	8
3.4	Numerical results for the IDR(s) ₁ and the IDR(s) ₂ algorithm	9
3.4.1	The relation between the IDR(s) ₁ and the IDR(s) ₂ algorithm	9
3.4.2	The primary and the secondary residuals	12
3.4.3	Comparing the four initiations of matrix \mathbf{U}	12
3.4.4	Comparing the IDR(s) ₁ and the IDR(s) ₂ algorithm	15
3.5	The convergence behaviour of the IDR(s) algorithm for growing subspace s	17
3.5.1	Stability	18
4	Bi-CGSTAB, IDR and the original IDR algorithm	18
4.1	The Bi-CGSTAB and the IDR algorithm	18
4.1.1	Convergence of the IDR and the Bi-CGSTAB algorithm	22
4.2	The IDR and the IDR(s) for $s = 1$	22
4.3	The original IDR algorithm	24
4.3.1	Comparing the IDR, the original IDR and the IDR(s) algorithm	25
5	Bi-CG	27
5.1	Numerical results for the Bi-CG algorithm	29
6	Bi-CGSTAB, the IDR(s) variant	31
6.1	Numerical results for the IDR(s) variant of the Bi-CGSTAB and the IDR(s)	34
6.1.1	The IDR(s) variant of the Bi-CGSTAB	34
6.1.2	The IDR(s) variant of the Bi-CGSTAB and the IDR(s)	35
7	Conclusion	38
8	Acknowledgement	39
9	References	40
10	Appendix	41

1 Introduction

In this thesis the focus will be on the IDR(s) algorithm. The IDR(s) is a new short recurrence method for solving large nonsymmetric systems of linear equations, based on the Induced Dimension Reduction (IDR) method. The IDR method was proposed in 1980 Sonneveld [4], but because of some instabilities it was left forgotten.

In Section 4.3 this IDR algorithm will be tested to detect the instabilities as stated in [1].

Recent research ([2],[3]) led to the development of the IDR(s) algorithm which showed that the choice of an s-dimensional “shadow” subspace \mathcal{S} leads to better effective search in the Krylov space for an approximate solution.

The IDR algorithm is equivalent with the Bi-CGSTAB algorithm (Section 4.1). Therefore we first describe in Section 2 how the Bi-CGSTAB algorithm is developed [1].

In Section 3 the behaviour of the IDR(s) algorithm will be tested for different starting strategies. Experimenting with this IDR(s) algorithm, led to two new variants of the IDR(s) (Section 3.2). IDR(s) selects the update for the approximate solution from an s-dimensional subspace. This subspace is updated at each step and spanned by the columns of a matrix, say, \mathbf{U} . We will initiate the matrix \mathbf{U} in four ways (Section 3.3). In Section 3.4 experiments will be done with different matrices to test which of the two IDR(s) variants gives the best convergence. From the two variants the best one will be chosen. We also try to choose the best initialization strategy for \mathbf{U} for the IDR(s) algorithm.

In Section 4.2 we will also see that for $s = 1$ the IDR(s) algorithm is equivalent with the IDR algorithm [3].

The Bi-CG algorithm [3] will be discussed in Section 5, which leads to the development of the IDR(s) variant of the Bi-CGSTAB. The IDR(s) variant of the Bi-CGSTAB is developed to understand the working of the IDR(s) algorithm [3]. In Section 6 we compare the IDR(s) and the IDR(s) variant of the Bi-CGSTAB.

2 The background of Bi-CGSTAB

In order to solve large linear systems of the form

$$\mathbf{Ax} = \mathbf{b},$$

where \mathbf{A} is a known $n \times n$ matrix, and \mathbf{b} is a given vector, iterative methods are used to find an accurate solution \mathbf{x} . The linear system could be real or complex, but for ease of discussion we will focus on a real linear system. Iterative methods like the Conjugate Gradient (CG) for symmetric systems and Bi-Conjugate Gradient (Bi-CG) and the GMRES for general systems can be used. These iterative methods search for the approximate solution in the Krylov subspaces $\mathbf{x}_0 + \mathcal{K}_k(\mathbf{A}, \mathbf{r}_0)$ where the space is generated by \mathbf{A} and the approximate solution \mathbf{x}_0 , where $\mathbf{r}_0 = \mathbf{b} - \mathbf{Ax}_0$ is the initial residual. Using GMRES the approximate solution can be found with the smallest residual norm. But the GMRES uses long recurrences that results in high memory storage and expensive computations if the number of iterative steps is not small. The iteration methods CG and Bi-CG use short recurrences, that lead to less memory storage and less computation work. Using short recurrences, we need a small number of vectors for the process. On the other hand, using short recurrence gives no optimal reduction of the residual norm.

The Bi-CG method is equivalent to the CG method for symmetrical problems. But the Bi-CG is twice more expensive per step than the CG method because it needs two matrix-vector multiplications per step, one with \mathbf{A} and one with \mathbf{A}^* .

A variant of the Bi-CG method, the Conjugate Gradient-Squares (CG-S) method, has been proved to work more effective for certain linear systems that are nonsymmetric. The CG-S converges faster than Bi-CG. The Bi-CG shows peaks in the convergence, and these peaks become squared in the the convergence of the CG-S. The peaks in the CG-S effect the accuracy of the solution and also the convergence.

In the CG method the residuals vectors \mathbf{r}_k are generated by a coupled 2-term recurrence relation. These residual vectors can also satisfy a 3-term recurrence relation. The norm $\|\mathbf{x} - \mathbf{x}_k\|_{\mathbf{A}}$ is minimized, where the matrix \mathbf{A} needs to be symmetric and positive definite. Approximations in the Bi-CG method are created by choosing the residual \mathbf{r}_j in such a way such that the residual \mathbf{r}_j is orthogonal with respect to another sequence of vectors $\hat{\mathbf{r}}_0, \hat{\mathbf{r}}_1, \dots, \hat{\mathbf{r}}_{j-1}$ and where $\hat{\mathbf{r}}_j$ is also orthogonal with respect to $\mathbf{r}_0, \mathbf{r}_1, \dots, \mathbf{r}_{j-1}$. We say that the residuals satisfy a *bi-orthogonality* relation. This can be accomplished by two 3-term recurrence relations, one for the sequence \mathbf{r}_j and one for $\hat{\mathbf{r}}_j$ or by two coupled 2-term recurrence relations. Theoretically the Bi-CG method finds an exact solution within n steps at most, when \mathbf{A} is a $n \times n$ matrix.

An observation made by Peter Sonneveld is that both the sequences (\mathbf{r}_j) and $(\hat{\mathbf{r}}_j)$ converge towards zero, where only (\mathbf{r}_j) is used. Sonneveld suggested to make a change in the Bi-CG so that the convergence effort is concentrated on the vectors \mathbf{r}_j only. The Bi-CG vectors can be written as $\mathbf{r}_j = \mathbf{P}_j(\mathbf{A})\mathbf{r}_0$ and $\hat{\mathbf{r}}_j = \mathbf{P}_j(\mathbf{A}^*)\hat{\mathbf{r}}_0$, where \mathbf{P}_j is a polynomial of degree j . From the bi-orthogonality relation we get the following equation $(\mathbf{r}_j, \hat{\mathbf{r}}_k) = (\mathbf{P}_j(\mathbf{A})\mathbf{r}_0, \mathbf{P}_k(\mathbf{A}^*)\hat{\mathbf{r}}_0) = (\mathbf{P}_k(\mathbf{A})\mathbf{P}_j(\mathbf{A})\mathbf{r}_0, \hat{\mathbf{r}}_0) = 0$ for $k < j$.

The vectors $\tilde{\mathbf{r}}_j \equiv \mathbf{P}_j^2(\mathbf{A})\mathbf{r}_0$ can be created using the latter form of the innerproduct for recovering the Bi-CG parameters. In this way there is no need to form the vectors $\hat{\mathbf{r}}_j$ (nor the vectors \mathbf{r}_j), and

thus also no multiplication with \mathbf{A}^* . The Bi-CG polynomial $\mathbf{P}_k(\mathbf{A})$ can be seen as a reductor working on \mathbf{r}_0 that is applied twice in the CG-S method: once on \mathbf{r}_0 (as in Bi-CG) and a second time as $\mathbf{P}_k(\mathbf{A})\mathbf{r}_0$. The reduction operator $\mathbf{P}_k(\mathbf{A})$ in Bi-CG is very dependent on the starting initial residual \mathbf{r}_0 , and that is why it is not likely to be an effective reduction operator for a vector, also for $\mathbf{P}_j(\mathbf{A})\mathbf{r}_0$.

As an alternative Van Der Vorst used a suitable polynomial $\tilde{\mathbf{P}}_k$, where \mathbf{x}_k is generated so that $\mathbf{r}_k = \tilde{\mathbf{P}}_k(\mathbf{A})\mathbf{P}_k(\mathbf{A})\mathbf{r}_0$. For $\tilde{\mathbf{P}}_k$ Van Der Vorst took a polynomial of the form

$$p_k(\mathbf{x}) = (1 - \omega_1\mathbf{x})(1 - \omega_2\mathbf{x}) \dots (1 - \omega_k\mathbf{x}),$$

with suitable constants ω_j . Van Der Vorst selected ω_j in the j -th iteration step to minimize \mathbf{r}_j where $\mathbf{r}_j = \tilde{\mathbf{P}}_{j-1}(\mathbf{A})\mathbf{P}_j(\mathbf{A})\mathbf{r}_0$. In Bi-CGSTAB, by doing this Van Der Vorst found that the method becomes mathematically equivalent with the IDR (Induced Dimension Reduction) method. More details about the IDR method can be found in Section 4.3.

The Bi-CGSTAB method proposed by Van Der Vorst can be found in his article [1]. The residual vector \mathbf{r}_k in Bi-CGSTAB is expressed as $\mathbf{r}_k = p_k(\mathbf{A})\mathbf{P}_k(\mathbf{A})\mathbf{r}_0$.

In Bi-CGSTAB, as well as in CG, CG-S and IDR, the vector \mathbf{x}_k can be easily determined as a by-product: if we update $\mathbf{r}_k = \mathbf{r}_{k-1} - \gamma\mathbf{A}\mathbf{y}$, then \mathbf{x}_k can be obtained as $\mathbf{x}_k = \mathbf{x}_{k-1} + \gamma\mathbf{y}$. The vector \mathbf{x}_k is not required for continuing the iteration, as in all CG-type methods.

It can be shown from the orthogonality property ($\mathbf{P}_k(\mathbf{A})\mathbf{r}_0, p_j(\mathbf{A}^*)\hat{\mathbf{r}}_0 = 0$), for $j < k$ that the Bi-CGSTAB is a finite method. This implies that the method in exact arithmetic will terminate after at most n iteration steps. For an $n \times n$ system $\mathbf{A}\mathbf{x} = \mathbf{b}$ the Bi-CGSTAB method requires 2 matrix vector multiplications (with \mathbf{A}), 6 vector updates and 4 innerproducts per iteration step. While the CG-S method requires 2 matrix vector products with \mathbf{A} , 6 vector updates, but only 2 innerproducts. The extra two inner products in the Bi-CGSTAB method causes small increase in computational work per iteration step. The memory space that is needed in the Bi-CGSTAB method is the same as in the CG-S method.

The advantage of the Bi-CGSTAB is that it produces smoother convergence and converges faster than the CG-S.

3 IDR(s)

The IDR(s) is a new short recurrence method for solving large nonsymmetric systems of linear equations, that is based on the Induced Dimension Reduction (IDR) method. The IDR was proposed in 1980 by Sonneveld [4]. The IDR method generates residuals in a sequence of shrinking subspaces. The residuals are generated in spaces \mathcal{G}_k . These spaces are recursively defined by $\mathcal{G}_{k+1} = (I - \omega_k A)(\mathcal{G}_k \cap \mathcal{S})$, where \mathcal{S} is a fixed proper subspace of \mathbb{C}^n , and $\omega_k \in \mathbb{C}$, $\omega_k \neq 0$.

The subspace \mathcal{S} is the space of vectors that are orthogonal to some vector $\tilde{\mathbf{r}}_0$. The exact solution can be computed in at most $2n$ matrix-vector multiplications.

The IDR method is also related with the Bi-CG method. The iteration polynomial constructed by IDR is the product of the Bi-CG polynomial with another locally minimizing polynomial. The IDR method led to the development of the CGS method. And from this method later on the Bi-CGSTAB was developed. The original IDR method is efficient as the Bi-CGSTAB method and is mathematically equivalent to the Bi-CGSTAB, but due to some instabilities the IDR method

became less attractive. The IDR method was left forgotten, and was over the years known only as the predecessor of the CG-S method.

Recent research ([2],[3]), as we will see in Section 3.2, showed that for \mathcal{S} a smaller fixed “shadow” subspace can be chosen. In this “shadow” subspace the residuals are constructed in such a way so that they are orthogonal to all the columns of an $n \times s$ matrix $\tilde{\mathbf{R}}_0$ with $s > 1$. By experimenting it appears that this choice of \mathcal{S} gives more effective search of the Krylov subspace and a appropriate solution. By using a subspace \mathcal{S} with co-dimension s the IDR(s) algorithm is developed.

If $s = 1$, the IDR(s) algorithm is mathematically equivalent with the original IDR algorithm. Only the new IDR(s) algorithm gives more stable results than the original algorithm [3]. In Section 4.3 the original IDR method will be tested for its numerical instability.

3.1 IDR

The residuals of the IDR method generally lie in a sequence (\mathcal{G}_k) of shrinking subspaces, i.e. $\mathcal{G}_{k+1} \subset \mathcal{G}_k$ and $\dim(\mathcal{G}_{k+1}) < \dim(\mathcal{G}_k)$ for all $k \in \mathbb{N}_0$ unless $\mathcal{G}_k = \{0\}$. The existence of such a sequence is guaranteed by the fundamental IDR theorem, see Theorem 3.1. Lemma 3.1 and Theorem 3.1 (that follows from the lemma) we describe here can also be found in [3, Lem. 2.1, Th. 2.2].

Lemma 3.1 *Let \mathcal{G}_0 and \mathcal{S} be subspaces and $\mu_0, \mu_1 \in \mathbb{C}$. Put $\mathcal{G}_1 \equiv (\mu_0 \mathbf{I} - \mathbf{A})(\mathcal{G}_0 \cap \mathcal{S})$.*

- 1) *If $\mathcal{G}_1 \subset \mathcal{G}_0$ then $(\mu_1 \mathbf{I} - \mathbf{A})(\mathcal{G}_1 \cap \mathcal{S}) \subset \mathcal{G}_1$*
- 2) *If $\mathcal{G}_1 = \mathcal{G}_0 \neq \{0\}$, then $\mathcal{G}_0 \cap \mathcal{S}$ contains an eigenvector of \mathbf{A} .*

Proof. Note that $(\mu_1 \mathbf{I} - \mathbf{A})(\mathcal{G}_1 \cap \mathcal{S}) \subset \mathcal{G}_1$ iff $(\mu_0 \mathbf{I} - \mathbf{A})(\mathcal{G}_1 \cap \mathcal{S}) \subset \mathcal{G}_1$. Therefore 1) follows from the inclusion $(\mu_0 \mathbf{I} - \mathbf{A})(\mathcal{G}_1 \cap \mathcal{S}) \subset (\mu_0 \mathbf{I} - \mathbf{A})(\mathcal{G}_0 \cap \mathcal{S}) = \mathcal{G}_1$.

If $\mathcal{G}_0 = \mathcal{G}_1$ then $\dim(\mathcal{G}_0) = \dim((\mu_1 \mathbf{I} - \mathbf{A})(\mathcal{G}_0 \cap \mathcal{S})) \leq \dim(\mathcal{G}_0 \cap \mathcal{S}) \leq \dim(\mathcal{G}_0)$.

In particular, $\dim(\mathcal{G}_0 \cap \mathcal{S}) = \dim(\mathcal{G}_0)$, whence $\mathcal{G}_0 \subset \mathcal{S}$ and $\mathcal{G}_0 = (\mu_0 \mathbf{I} - \mathbf{A})(\mathcal{G}_0 \cap \mathcal{S}) \subseteq (\mu_0 \mathbf{I} - \mathbf{A})\mathcal{G}_0$, which implies that $\mu_0 \mathbf{I} - \mathbf{A}$, and therefore \mathbf{A} , has an eigenvector in \mathcal{G}_0 unless $\mathcal{G}_0 = \{0\}$. \square

The IDR theorem has been formulated in terms of subspaces of a linear subspace \mathcal{S} . The theorem will be formulated in terms of complement space $\mathcal{S} = \tilde{\mathbf{R}}_0^\perp$.

Theorem 3.1 *Let $\tilde{\mathbf{R}}_0 = [\tilde{\mathbf{r}}_1, \dots, \tilde{\mathbf{r}}_s]$ be an $n \times s$ matrix and let (μ_j) be a sequence in \mathbb{C} . With $\mathcal{G}_0 \equiv \mathbb{C}^n$, define*

$$\mathcal{G}_{k+1} \equiv (\mu_k \mathbf{I} - \mathbf{A})(\mathcal{G}_k \cap \tilde{\mathbf{R}}_0^\perp) \quad (k = 0, 1, \dots) \quad (1)$$

If $\tilde{\mathbf{R}}_0^\perp$ does not contain an eigenvector of \mathbf{A} , then for all $k = 0, 1, \dots$, we have that

- 1) *$\mathcal{G}_{k+1} \subset \mathcal{G}_k$, and*
- 2) *$\dim \mathcal{G}_{k+1} < \dim \mathcal{G}_k$ unless $\mathcal{G}_k = \{0\}$*

Proof. Take $\mathcal{S} = \tilde{\mathbf{R}}_0^\perp$ and apply the lemma inductively. \square

3.2 IDR(s)

The residuals that are used by the IDR(s) method are constructed in the spaces \mathcal{G}_k . In each step the residuals are updated, where short recurrences are used. The update residuals \mathbf{r} for the IDR method are of the form $\mathbf{r}_+ = \mathbf{r} - \mathbf{c}\alpha$, with $\mathbf{c} = \mathbf{A}\mathbf{u}$, where \mathbf{u} is an available vector. By this the approximate solution $\tilde{\mathbf{x}}$, $\mathbf{r} = \mathbf{b} - \mathbf{A}\tilde{\mathbf{x}}$ can be updated. This will cost only one AXPY (vector update): from $\tilde{\mathbf{x}}_+ \equiv \tilde{\mathbf{x}} + \mathbf{u}\alpha$ we have that $\mathbf{r}_+ = \mathbf{b} - \mathbf{A}\tilde{\mathbf{x}}_+$.

For the computation of \mathbf{r} we need one MV (matrix-vector multiplication) for $\mathbf{c} = \mathbf{A}\mathbf{u}$. To update \mathbf{r} the scalar α needs to be computed, where we probably will need one (or a few) DOT product(s). The approximate solutions get almost a free ride. Further on we see that some vector updates can be combined: $\mathbf{r}_+ = \mathbf{r} - \mathbf{C}\vec{\alpha}$, with $\mathbf{C} = \mathbf{A}\mathbf{U}$, where \mathbf{U} , a $n \times s$ matrix is available, then $\tilde{\mathbf{x}}_+ = \tilde{\mathbf{x}} + \mathbf{U}\vec{\alpha}$. If a column of \mathbf{C} as $\mathbf{C}\vec{\gamma} + \omega\mathbf{A}\mathbf{v}$ is updated, then a corresponding column \mathbf{U} can be updated as $\mathbf{U}\vec{\gamma} + \omega\mathbf{v}$.

In discussions on the derivation of the algorithms we will concentrate on the updates of the residuals of the form $\mathbf{A}\mathbf{U}$ with \mathbf{U} available.

The IDR(s) method requires at most $n + n/s$ matrix-vector multiplications, where n is the problem size, and s is the dimension of the subspace.

Another important part of the IDR(s) method is the bi-orthogonalization. If we want to bring $\mathbf{r} \in \mathcal{G}_k$ to $\mathbf{r}_+ \in \mathcal{G}_{k+1}$, first a residual vector \mathbf{v} in $\mathcal{G}_k \cap \tilde{\mathbf{R}}_0^\perp$ needs to be formed, we do that by subtracting a vector of the form $\mathbf{S}\vec{\gamma}$ and \mathbf{S} with columns in \mathcal{G}_k , from \mathbf{r} . So $\mathbf{v} = \mathbf{r} - \mathbf{S}\vec{\gamma}$ where $\mathbf{S} = \mathbf{A}\mathbf{U}$ and $\vec{\gamma} = (\tilde{\mathbf{R}}_0^*\mathbf{S})^{-1}\tilde{\mathbf{R}}_0^*\mathbf{r}$. Matrix \mathbf{S} is a $n \times s$ matrix and $\tilde{\mathbf{R}}_0^*\mathbf{S}$ needs to be a nonsingular matrix. The matrix \mathbf{S} is constructed as a difference of $s + 1$ residuals.

\mathbf{v} is multiplied by $\mathbf{I} - \omega\mathbf{A}$ for some appropriate ω , so we get that $\mathbf{r}_+ = \mathbf{v} - \omega\mathbf{A}\mathbf{v}$.

$\mathbf{r}_+ - \mathbf{r}$ is of the form $\mathbf{A}\mathbf{u}$, where $\mathbf{u} = \tilde{\mathbf{x}} - \tilde{\mathbf{x}}_+$ if $\mathbf{r} = \mathbf{b} - \mathbf{A}\tilde{\mathbf{x}}$ and $\mathbf{r}_+ = \mathbf{b} - \mathbf{A}\tilde{\mathbf{x}}_+$. Algorithm 3.1 gives us the IDR(s) algorithm.

The IDR(s) algorithm did undergo some changes compared to the algorithm described in article [3], because the IDR(s) algorithm from [3] had an error. In the IDR(s) algorithm of article [3] the matrix \mathbf{S} grows in dimension after one iteration. This is not supposed to happen. To correct this we introduced a new variable j in the IDR(s) algorithm.

In the IDR(s) algorithm (see Alg. 3.1) the parameter i is needed to update the matrices \mathbf{U} and \mathbf{S} and j is needed to compute ω once in a while.

Because we want to get rid of one parameter, so that the IDR(s) algorithm looks more simple we create the following variant of the IDR(s) method, see Alg. 3.2. This variant uses only one parameter j that updates the matrices \mathbf{U} and \mathbf{S} and computes ω .

For ease of discussion we give the two variants of the IDR(s) algorithm a name. We name Alg. 3.1 with the two parameters i and j IDR(s)₂ and Alg. 3.2 with only one parameter j we name IDR(s)₁. In Section 3.4 we will compare the IDR(s)₁ and the IDR(s)₂ algorithm.

```

Select an  $\mathbf{x}_0$ 
Select  $n \times s$  matrices  $\tilde{\mathbf{R}}_0$  and  $\mathbf{U}$ 
 $\mathbf{x} = \mathbf{x}_0$ ,  $\mathbf{r} = \mathbf{b} - \mathbf{A}\mathbf{x}$ 
Compute  $\mathbf{S} = \mathbf{A}\mathbf{U}$ 
 $i = s + 2$ 
 $j = i$ 
 $\sigma = \tilde{\mathbf{R}}_0^* \mathbf{S}$ 
repeat
  Solve  $\sigma \vec{\gamma} = \tilde{\mathbf{R}}_0^* \mathbf{r}$  for  $\vec{\gamma}$ 
   $\mathbf{v} = \mathbf{r} - \mathbf{S} \vec{\gamma}$ ,  $\mathbf{c} = \mathbf{A}\mathbf{v}$ 
  if  $i > s$ ,  $i = 1$ , end if
  if  $j > s + 1$ 
     $\omega \leftarrow \mathbf{c}^* \mathbf{v} / \mathbf{c}^* \mathbf{c}$ ,  $j = 1$ 
  end if
   $\mathbf{U}e_i \leftarrow \mathbf{U} \vec{\gamma} + \omega \mathbf{v}$ 
   $\mathbf{r}_1 \leftarrow \mathbf{v} - \omega \mathbf{c}$ ,  $\mathbf{S}e_i \leftarrow \mathbf{r} - \mathbf{r}_1$ ,  $\mathbf{r} \leftarrow \mathbf{r}_1$ 
   $\sigma e_i \leftarrow \tilde{\mathbf{R}}_0^* \mathbf{S}e_i$ 
   $i \leftarrow i + 1$ ,  $j \leftarrow j + 1$ 
end repeat

```

Alg. 3.1: *The IDR(s) algorithm with two parameters, i and j .
This algorithm will be also referred as IDR(s)₂.*

3.3 Initiating the matrix \mathbf{U} for the IDR(s) algorithm

The $n \times s$ matrix \mathbf{U} from the IDR(s) algorithm can be initialized in different ways. In this section we will discuss four ways in which we can initialize the matrix \mathbf{U} .

1. \mathbf{U} initiated by the Arnoldi iteration, where $\mathbf{U} = [\mathbf{r}, \mathbf{A}\mathbf{r}, \dots, \mathbf{A}^{s-1}\mathbf{r}]$ is orthogonalized, with $\mathbf{S} = \mathbf{A}\mathbf{U}$.
2. $\mathbf{U} = \mathbf{r}$, where $\mathbf{r} = \mathbf{b} - \mathbf{A}\mathbf{x}$ and \mathbf{x} is the initial guess. Taking $\mathbf{U} = \mathbf{r}$ is simple and saves computation work. This initiation of \mathbf{U} will be further discussed in Section 3.3.1.
3. $\mathbf{U} = \tilde{\mathbf{R}}_0$, $\mathbf{S} = \mathbf{A}\mathbf{U}$. The $n \times s$ matrix $\tilde{\mathbf{R}}_0$ is randomly chosen and orthogonalized. Because $\tilde{\mathbf{R}}_0$ is orthogonal, the computations are kept cheap.
4. $\mathbf{U} = \tilde{\mathbf{R}}_0$, $\mathbf{S} = \mathbf{A}\mathbf{U}$, $\tilde{\mathbf{R}}_0 = \mathbf{S}$. Initiating the matrix \mathbf{U} here is the same as in 3, but because $\tilde{\mathbf{R}}_0 = \mathbf{S}$, \mathbf{U} becomes also well-conditioned.

```

Select an  $\mathbf{x}_0$ 
Select  $n \times s$  matrices  $\tilde{\mathbf{R}}_0$  and  $\mathbf{U}$ 
 $\mathbf{x} = \mathbf{x}_0$ ,  $\mathbf{r} = \mathbf{b} - \mathbf{A}\mathbf{x}$ 
Compute  $\mathbf{S} = \mathbf{A}\mathbf{U}$ 
 $j = s + 1$ 
 $\sigma = \tilde{\mathbf{R}}_0^* \mathbf{S}$ 
repeat
  Solve  $\sigma \vec{\gamma} = \tilde{\mathbf{R}}_0^* \mathbf{r}$  for  $\vec{\gamma}$ 
   $\mathbf{v} = \mathbf{r} - \mathbf{S}\vec{\gamma}$ ,  $\mathbf{c} = \mathbf{A}\mathbf{v}$ 
  if  $j > s$ 
     $\omega \leftarrow \mathbf{c}^* \mathbf{v} / \mathbf{c}^* \mathbf{c}$ ,  $j = 0$ 
  end if
   $\mathbf{w} = \mathbf{U}\vec{\gamma} + \omega \mathbf{v}$ ,  $\mathbf{x} = \mathbf{x} + \mathbf{w}$ 
   $\mathbf{r}_1 = \mathbf{v} - \omega \mathbf{c}$ 
  if  $j \neq 0$ 
     $\mathbf{U}e_j \leftarrow \mathbf{w}$ 
     $\mathbf{S}e_j \leftarrow \mathbf{r} - \mathbf{r}_1$ 
     $\sigma e_j \leftarrow \tilde{\mathbf{R}}_0^* \mathbf{S}e_j$ 
  end
   $\mathbf{r} \leftarrow \mathbf{r}_1$ ,  $j \leftarrow j + 1$ 
end repeat

```

Alg. 3.2: Variant of the IDR(s) method with one parameter j : IDR(s)₁.

3.3.1 Initial matrices $\mathbf{U} = \mathbf{r}$ and $\mathbf{U} = \mathbf{Arnoldi}$

Note that the initiation of the matrix \mathbf{U} where $\mathbf{U} = \mathbf{r}$ works in Matlab, but mathematically this is not the case. By working with the IDR(s) algorithm in Matlab it showed that the method works well if $\mathbf{U} = \mathbf{r}$, where $\mathbf{r} = \mathbf{b} - \mathbf{A}\mathbf{x}_0$. But the question arises why does this work?

Looking at the IDR(s) algorithm we try to understand what is happening if $\mathbf{U} = \mathbf{r}$ for $s = 1$ in the first iteration step (we are in the space $\mathcal{G}_1 = (\mathbf{I} - \omega \mathbf{A})(\mathcal{G}_0 \cap \tilde{\mathbf{R}}_0^\perp)$):

When the matrix \mathbf{U} is initiated as $\mathbf{U} = \mathbf{r}$ we have that $\mathbf{S} = \mathbf{A}\mathbf{U}$

The Matlab command $\tilde{\mathbf{R}}_0 \mathbf{S} \backslash \tilde{\mathbf{R}}_0^* \mathbf{r}$ actually computes $\vec{\gamma} = \min_{\arg} \| \tilde{\mathbf{R}}_0^* \mathbf{S} \vec{\gamma} - \tilde{\mathbf{R}}_0^* \mathbf{r} \|$

For $\mathbf{U} = \mathbf{r}$ the IDR(s) algorithm computes:

$$\mathbf{v} = \mathbf{r} - \mathbf{S}\vec{\gamma} = \mathbf{r} - \mathbf{A}\mathbf{r}\vec{\gamma}$$

$$\mathbf{x} = \mathbf{x} + \mathbf{U}\vec{\gamma}$$

$$\mathbf{c} = \mathbf{A}\mathbf{v}, \omega = \frac{\mathbf{c}^* \mathbf{v}}{\mathbf{c}^* \mathbf{c}}$$

$$\mathbf{U}e_1 = \mathbf{U}\vec{\gamma} + \omega \mathbf{v} = \mathbf{r}\vec{\gamma} + \omega \mathbf{v} = \mathbf{r}\vec{\gamma} + \omega(\mathbf{r} - \mathbf{A}\mathbf{r}\vec{\gamma})$$

$$\mathbf{S}e_1 = \mathbf{A}\mathbf{U}e_1 = \mathbf{A}(\mathbf{U}\vec{\gamma} + \omega \mathbf{v}) = \mathbf{A}\mathbf{r}\vec{\gamma} + \omega \mathbf{A}\mathbf{v} = \mathbf{A}\mathbf{r}\vec{\gamma} + \omega \mathbf{A}(\mathbf{r} - \mathbf{A}\mathbf{r}\vec{\gamma}) = \mathbf{A}\mathbf{r}\vec{\gamma} + \omega(\mathbf{A}\mathbf{r} - \mathbf{A}^2 \mathbf{r}\vec{\gamma})$$

$$\mathbf{r} = \mathbf{v} - \omega \mathbf{c}, \mathbf{x} = \mathbf{U} + \omega \mathbf{v}$$

When the matrix \mathbf{U} is initialized as $\mathbf{U} = \mathbf{r}$ for $s = 1$ we get that $\mathbf{U}e_1 = \hat{\mathbf{r}}$ where $\hat{\mathbf{r}} = \mathbf{r}\vec{\gamma} + \omega(\mathbf{r} - \mathbf{A}\mathbf{r}\vec{\gamma})$. When $s > 1$, for \mathbf{U} we have that the Krylov space is spanned by the vectors: $[\hat{\mathbf{r}}, \mathbf{A}\hat{\mathbf{r}}, \dots, \mathbf{A}^s\hat{\mathbf{r}}]$, where $\hat{\mathbf{r}} = \mathbf{r}\vec{\gamma} + \omega(\mathbf{r} - \mathbf{A}\mathbf{r}\vec{\gamma})$. When $\mathbf{U} = \mathbf{r}$, the $\text{Span}(\mathbf{U}) \subseteq \text{Span}(\hat{\mathbf{r}}, \mathbf{A}\hat{\mathbf{r}}, \dots, \mathbf{A}^s\hat{\mathbf{r}})$.

Initiating the matrix \mathbf{U} by the Arnoldi iteration for $s = 1$ we get that $\mathbf{U}e_1 = \mathbf{r}$ and for $s > 1$ we have that for \mathbf{U} the Krylov space is spanned by the vectors $[\mathbf{r}, \mathbf{A}\mathbf{r}, \dots, \mathbf{A}^{s-1}\mathbf{r}]$.

In further experiments we will see that the initial matrices $\mathbf{U} = \mathbf{r}$ and $\mathbf{U} = \text{Arnoldi}$ have different convergence. This is because the Krylov space for \mathbf{U} is spanned by different vectors for the two initiations, which leads to different convergence for $s > 1$. In Section 3.4.3 we will discuss which of these two initiations give more stable result for the IDR(s).

3.4 Numerical results for the IDR(s)₁ and the IDR(s)₂ algorithm

The IDR(s)₁ and the IDR(s)₂ algorithm are tested in Matlab, to measure which of the two algorithms produces better results. First the IDR(s)₁ and the IDR(s)₂ algorithm are compared to find out if there exists any relation between these two algorithms. Then the four different initiations for the matrix \mathbf{U} (Section 3.3) will be compared, and we try to decide which initial matrix \mathbf{U} leads to an optimal convergence for the IDR(s) algorithm. After finding the best initiation for \mathbf{U} we will decide which of two IDR(s) algorithm is more effective.

3.4.1 The relation between the IDR(s)₁ and the IDR(s)₂ algorithm

Experimenting with the IDR(s)₁ and the IDR(s)₂ algorithm we found that the two IDR(s) variants share a relation when the matrix \mathbf{U} is initiated by the Arnoldi iteration for any subspace s . The relation implies that for every $s + 1$ step the two IDR(s) variants produce the same residuals. We call this relation an $s + 1$ relation. Initiating the matrix \mathbf{U} as $\mathbf{U} = \tilde{\mathbf{R}}_0$ with $\mathbf{S} = \mathbf{A}\mathbf{U}$ shows also an $s + 1$ relation for the two IDR(s) algorithms. When $\mathbf{U} = \tilde{\mathbf{R}}_0$, $\mathbf{S} = \mathbf{A}\mathbf{U}$, $\tilde{\mathbf{R}}_0 = \mathbf{S}$ we also found an $s + 1$ relation for the IDR(s)₁ and the IDR(s)₂ algorithm.

For every $s + 1$ step the two IDR(s) variants give the same result for the matrix \mathbf{U} initiated in these three ways. We use the same matrices $\tilde{\mathbf{R}}_0$ and \mathbf{U} for the two IDR(s) variants. For the two IDR(s) variants the same residual vector \mathbf{r} is also computed and stored, where $\mathbf{r} = \mathbf{v} - \omega\mathbf{A}\mathbf{v}$ (ω is available). These similarities between the IDR(s)₁ and the IDR(s)₂ algorithm could explain why the two IDR(s) variants have the same residuals every $s + 1$ step.

It can be shown that $s + 1$ steps of $\mathbf{v} \equiv \mathbf{r}_j - \mathbf{S}_j\vec{\gamma}$, $\mathbf{r}_{j+1} = \mathbf{v} - \omega\mathbf{A}\mathbf{v}$ and $\mathbf{s}_j \equiv \mathbf{r}_j - \mathbf{r}_{j+1}$ is needed to bring $s + 1$ residuals in \mathcal{G}_k to $s + 1$ residuals in \mathcal{G}_{k+1} . The value for ω is selected at step 1 of a $s + 1$ cycle. In the remaining s steps of the cycle the same ω is used. By this at every $s + 1$ steps s innerproducts can be saved.

When the matrix \mathbf{U} is initiated as $\mathbf{U} = \mathbf{r}$ there only exists a $s + 1$ relation for the IDR(s)₁ and the IDR(s)₂ algorithm when subspace $s = 1$. For $s > 1$ there is no $s + 1$ relation between two IDR(s) algorithms. The reason why there is no $s + 1$ relation between the two IDR(s) variants for the initiation $\mathbf{U} = \mathbf{r}$ is because we have that for $s > 1$ the Krylov space for \mathbf{U} is spanned by the

vectors $[\hat{\mathbf{r}}, \mathbf{A}\hat{\mathbf{r}}, \dots, \mathbf{A}^s\hat{\mathbf{r}}]$, where $\hat{\mathbf{r}} = \mathbf{r}\vec{\gamma} + \omega(\mathbf{r} - \mathbf{A}\mathbf{r}\vec{\gamma})$ (see Section 3.3).

In figure 1 we plot the graphs for the $\text{IDR}(s)_1$ and the $\text{IDR}(s)_2$ algorithm, using the initial matrix $\mathbf{U} = \mathbf{r}$ for $s = 4$. This figure shows us that the graphs of the two $\text{IDR}(s)$ algorithms run closely, but without a $s + 1$ relation.

Figure 2 shows the $s + 1$ relation between the $\text{IDR}(s)_1$ and the $\text{IDR}(s)_2$ algorithm for \mathbf{U} initiated by the Arnoldi iteration for $s = 4$. By a close look it can be seen that for every 5 steps ($s + 1$ step, where $s = 4$) the two variants of the $\text{IDR}(s)$ algorithm are the same.

For figure 1 and figure 2 the matrix \mathbf{A} is chosen as a 100×100 diagonal matrix ($\mathbf{A} = \text{diag}(1 : n)$). The $n \times s$ matrix \mathbf{R}_0 is a randomly chosen and orthogonalized, where $n = 100$ and $s = 4$.

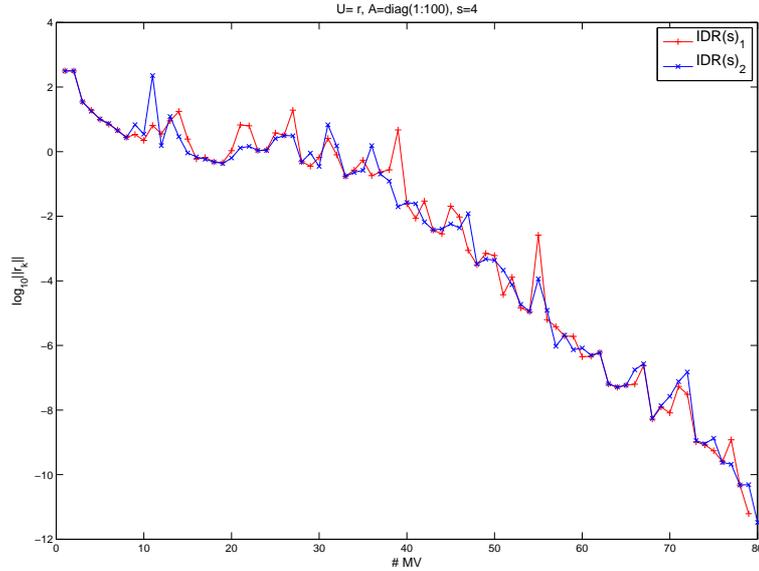


Figure 1: *The $\text{IDR}(s)_1$ and the $\text{IDR}(s)_2$ algorithm do not produce the same residuals at every $s + 1$ step when \mathbf{U} initiated as $\mathbf{U} = \mathbf{r}$.*

In search for a $s + 1$ relation for the $\text{IDR}(s)_1$ and the $\text{IDR}(s)_2$ algorithm, where the matrix \mathbf{U} is initiated as $\mathbf{U} = \mathbf{r}$, some changes are made in the $\text{IDR}(s)_1$ algorithm. The resulting convergence graph of the changed $\text{IDR}(s)_1$ algorithm and the $\text{IDR}(s)_2$ algorithm can show if the changes made lead to a $s + 1$ relation.

In the $\text{IDR}(s)_2$ algorithm the matrices \mathbf{U} and \mathbf{S} are filled in the first s steps. To fill the matrices \mathbf{U} and \mathbf{S} in the $\text{IDR}(s)_1$ algorithm also in the first s steps, we set $\mathbf{U} = \mathbf{0}$, that leads to $\mathbf{S} = \mathbf{A}\mathbf{U} = \mathbf{0}$. For $j = s + 1$ σ will be computed, where $\sigma = \mathbf{R}_0^* \mathbf{S}$. Because we have $\mathbf{S} = \mathbf{0}$ for the first step, $\sigma = 0$. If $\sigma = 0$, we can not proceed. To correct this, we take $\sigma = 1$ if the norm of $\mathbf{U} = \mathbf{0}$.

γ is computed followed by \mathbf{v} and \mathbf{c} , where $\mathbf{c} = \mathbf{A}\mathbf{v}$. For $j > s$, ω is computed and j is set 0. As in Alg. 3.2 the vectors \mathbf{w} , \mathbf{x} and \mathbf{r}_1 are computed. To fill the matrices \mathbf{U} and \mathbf{S} in the first steps the $\text{IDR}(s)_1$ algorithm will be changed as follows: Instead of $j \neq 0$, we put $j \neq 0, i = j$. From here on

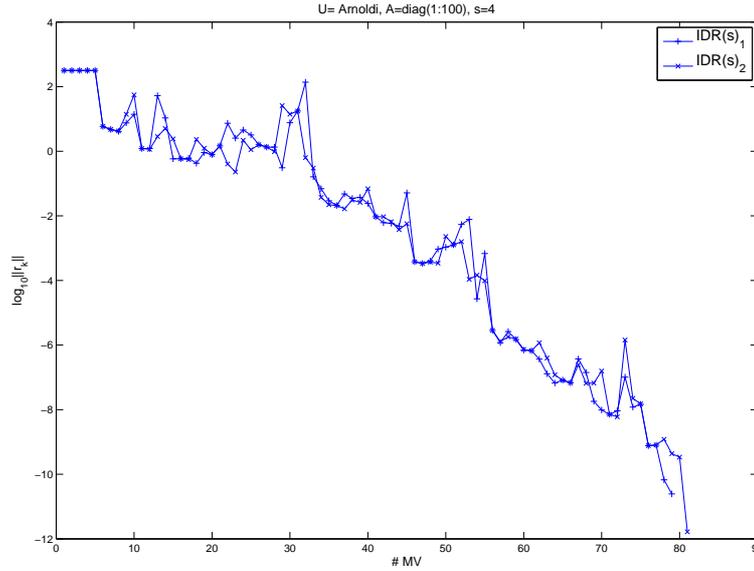


Figure 2: *The IDR(s)₁ and the IDR(s)₂ algorithm produce the same residuals at every s + 1 step. The matrix \mathbf{U} is initiated by the Arnoldi iteration and s = 4.*

the last part of the IDR(s)₁ algorithm would be like:

```

if j ≠ 0, i = j
   $\mathbf{U}e_i \leftarrow \mathbf{w}$ 
   $\mathbf{S}e_i \leftarrow \mathbf{r} - \mathbf{r}_1$ 
   $\sigma e_i \leftarrow \tilde{\mathbf{R}}_0^* \mathbf{S}e_i$ 
end
 $\mathbf{r} \leftarrow \mathbf{r}_1, j \leftarrow j + 1$ 
end repeat

```

By the changes made in the IDR(s)₁ algorithm as described above we hope that the IDR(s)₁ and the IDR(s)₂ algorithm produce the same residuals every s + 1 step. In figure 3 the graph of the changed IDR(s)₁ algorithm is plotted together with the graph of the IDR(s)₂ for s = 4 and \mathbf{U} initiated as $\mathbf{U} = \mathbf{r}$. But from the resulting plots in figure 3 there is no existence of any s + 1 relation for the two IDR(s) algorithms.

From these results we have to accept that there is no s + 1 relation between the two variants of the IDR(s) algorithm when the matrix \mathbf{U} is initiated as $\mathbf{U} = \mathbf{r}$.

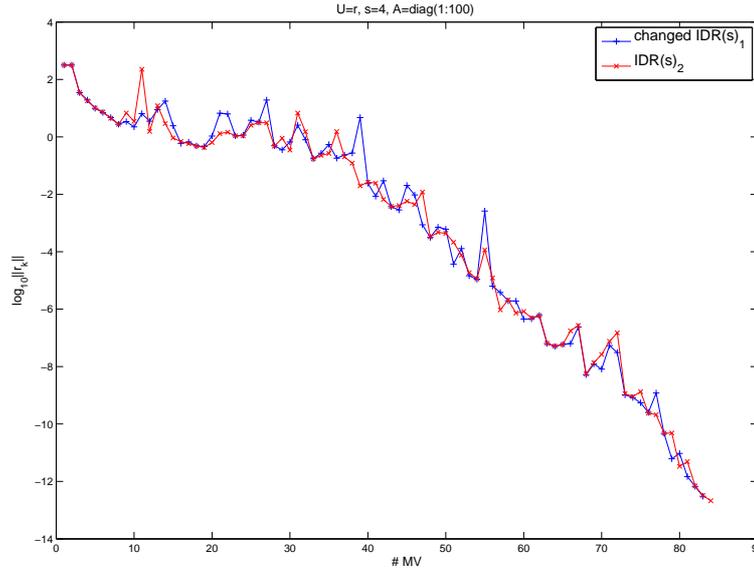


Figure 3: After the changes made in the $\text{IDR}(s)_1$ algorithm, the changed $\text{IDR}(s)_1$ and the $\text{IDR}(s)_2$ still not have the same residual every $s + 1$ step when \mathbf{U} is initiated as $\mathbf{U} = \mathbf{r}$.

3.4.2 The primary and the secondary residuals

From the figures (Section 3.4.1) it is clear that the two variants of the $\text{IDR}(s)$ method, with matrix \mathbf{U} initiated in one of the four ways, as described in Section 3.3, have the same start but after a while the convergence differs. As an example we take figure 2. We see that the graphs of the two $\text{IDR}(s)$ variants have the exact same beginning, but after some time they behave different from each other.

We assume that the convergence plot of the two $\text{IDR}(s)$ variants are the same in the beginning, because the primary residuals are the same for the variants $\text{IDR}(s)_1$ and $\text{IDR}(s)_2$. The primary residual is defined by the residual that is created for the first time in \mathcal{G}_k . All the other residuals (like σ , see Alg. 3.1 and Alg. 3.2) are called the secondary residuals.

Beside that the two $\text{IDR}(s)$ variants produce the same residuals every $s + 1$ step (except for the initiation $\mathbf{U} = \mathbf{r}$), the convergence graphs of the two variants the $\text{IDR}(s)$ algorithm produce also the same primary residuals.

3.4.3 Comparing the four initiations of matrix \mathbf{U}

In this part the four initiation of the matrix \mathbf{U} , as described in Section 3.3, will be tested for the $\text{IDR}(s)$ algorithm. We will select the initial matrix \mathbf{U} which gives us the best convergence for the $\text{IDR}(s)$ algorithm.

From the previous section we know that the $\text{IDR}(s)_1$ and the $\text{IDR}(s)_2$ produce the same residuals every $s + 1$ step. Therefore, we present the results for one variant of the $\text{IDR}(s)$ algorithm, the

IDR(s)₂ algorithm.

To study the convergence behaviour of the IDR(s)₂ algorithm for different initial matrices \mathbf{U} , some other matrices than the matrix $\mathbf{A} = \text{diag}(1 : n)$ will be used. The results we get using the diagonal matrix $\mathbf{A} = \text{diag}(1 : n)$ is not enough to decide which initial matrix \mathbf{U} gives the best convergence for the IDR(s) algorithm. The convergence of the IDR(s) algorithm is compared with the convergence of the GMRES. When the convergence graph of the IDR(s) runs closely to the graph of the GMRES at comparable error rates, we can say that the IDR(s) algorithm converges well. We have to notice that convergence of an algorithm is also dependent on the test matrix \mathbf{A} we experiment with. In this section we make use of the test matrices MEIER01 [6] and TFQMR001 [6].

MEIER01

The MEIER01 matrix is a result of an advection dominated 2nd order partial differential equation, with Dirichlet boundary conditions on the unit cube [5]:

$$-u_{xx} - u_{yy} - u_{zz} + 1000u_x = f,$$

where the function f is defined by the solution $u(x, y, z) = \exp(xyz)\sin(\pi x)\sin(\pi y)\sin(\pi z)$. The equation has been discretized using $10 \times 10 \times 10$ finite volumes and central differences for u_x . This gives a seven diagonal linear system of order 1000.

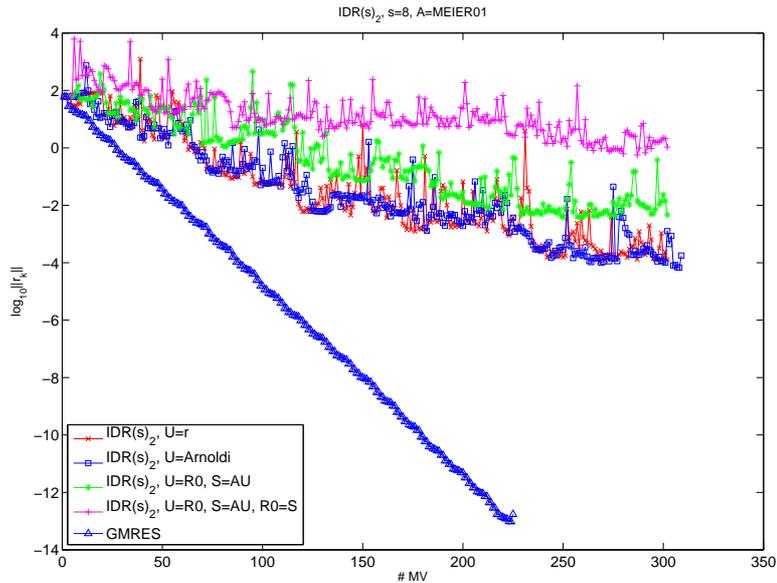


Figure 4: The initiation $\mathbf{U} = \text{Arnoldi}$ and the initiation $\mathbf{U} = \mathbf{r}$ produce comparable convergence for the IDR(s)₂. The convergence produced by these two initiations is also better than the convergence produced by initiations: $\mathbf{U} = \tilde{\mathbf{R}}_0$, $\mathbf{S} = \mathbf{AU}$ and $\mathbf{U} = \tilde{\mathbf{R}}_0$, $\mathbf{S} = \mathbf{AU}$, $\tilde{\mathbf{R}}_0 = \mathbf{S}$. Matrix $\mathbf{A} = \text{MEIER01}$ and $s = 8$.

Experimenting with the MEIER01 matrix, we found that the matrix \mathbf{U} initiated as $\mathbf{U} = \mathbf{r}$ and \mathbf{U}

initiated by Arnoldi converges better than the initiations $\mathbf{U} = \tilde{\mathbf{R}}_0$, $\mathbf{S} = \mathbf{AU}$ and $\mathbf{U} = \tilde{\mathbf{R}}_0$, $\mathbf{S} = \mathbf{AU}$, $\tilde{\mathbf{R}}_0 = \mathbf{S}$.

Figure 4 shows the convergence behaviour of the four initiations of \mathbf{U} for the $\text{IDR}(s)_2$ algorithm, where we use matrix MEIER01 for subspace $s = 8$. The matrix $\tilde{\mathbf{R}}_0$ is a randomly chosen $n \times s$ orthogonalized matrix.

Comparing the convergence graph of the initiation $\mathbf{U} = \mathbf{r}$ and \mathbf{U} initiated by the Arnoldi iteration, we see that the two graphs runs very close to each other.

These two graphs are the closest to the convergence of the GMRES. It is hard to decide which of these two initial matrices \mathbf{U} ($\mathbf{U} = \mathbf{r}$ or \mathbf{U} initiated by the Arnoldi) produces better convergence. The matrix \mathbf{U} initiated by the Arnoldi iteration and the initial matrix where $\mathbf{U} = \mathbf{r}$ gives comparable results for the $\text{IDR}(s)_2$ algorithm.

TFQMR001

The TFQMR001 is a result of 63×63 finite volume discretization, with

$$-u_{xx} - u_{yy} + \gamma(xu_x + yu_y) + \beta u = f \quad (2)$$

on the unit cube with Dirichlet boundary conditions. Here $\gamma = 100$ and $\beta = -200$ [5]. The resulting matrix we get is of order $n = 3969$.

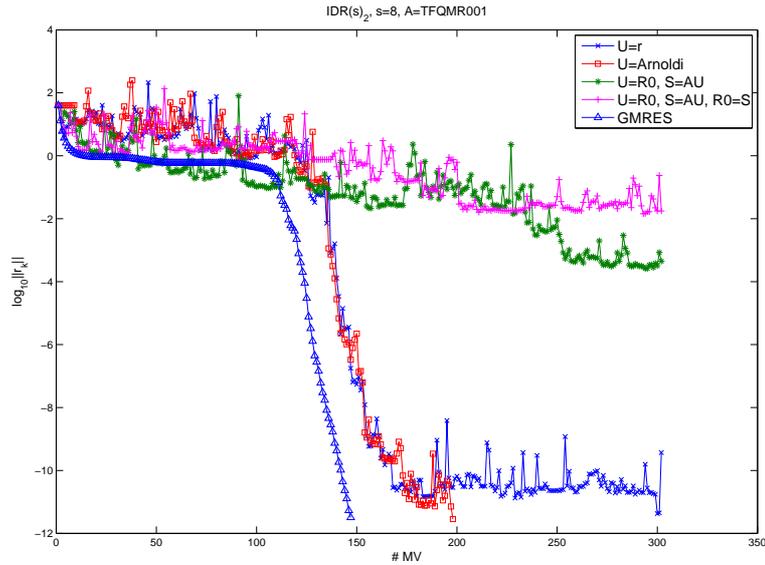


Figure 5: The matrix \mathbf{U} initiated by the Arnoldi iteration and \mathbf{U} initiated as $\mathbf{U} = \mathbf{r}$ produces comparable convergence for the $\text{IDR}(s)_2$ algorithm. Again these two initiation produce the best convergence compared to the other two initiations: $\mathbf{U} = \tilde{\mathbf{R}}_0$, $\mathbf{S} = \mathbf{AU}$ and $\mathbf{U} = \tilde{\mathbf{R}}_0$, $\mathbf{S} = \mathbf{AU}$, $\tilde{\mathbf{R}}_0 = \mathbf{S}$. The matrix $\mathbf{A} = \text{TFQMR001}$ and $s = 8$.

The results from the TFQMR001 matrix for the four initial matrices \mathbf{U} are comparable with the results of the MEIER01 matrix. Figure 5 presents the results for the $\text{IDR}(s)_2$ algorithm using the

TFQMR001 matrix for $s = 8$. The matrix $\tilde{\mathbf{R}}_0$ is a randomly chosen $n \times s$ orthogonalized matrix. For the TFQMR001 we found clear evidence that the initiations $\mathbf{U} = \tilde{\mathbf{R}}_0$, $\mathbf{S} = \mathbf{A}\mathbf{U}$ and $\mathbf{U} = \tilde{\mathbf{R}}_0$, $\mathbf{S} = \mathbf{A}\mathbf{U}$, $\tilde{\mathbf{R}}_0 = \mathbf{S}$ do not produce optimal convergence for the IDR(s)₂ algorithm.

In figure 5 it is visible that the convergence of graphs, where \mathbf{U} generated as $\mathbf{U} = \mathbf{r}$ and \mathbf{U} initiated by Arnoldi, approach the convergence graph of the GMRES. The convergence produced by the initiation $\mathbf{U} = \mathbf{r}$ and \mathbf{U} initiated by the Arnoldi produce comparable convergence for the IDR(s)₂ algorithm till a residual norm of 10^{-10} .

From the test results using the matrices MEIER01 and TFQMR001 we could not decide which of the two initiations of \mathbf{U} : $\mathbf{U} = \mathbf{r}$ or \mathbf{U} initiated by the Arnoldi, produce better convergence. Both initiations for \mathbf{U} produces stable convergence. The matrix \mathbf{U} initiated by the Arnoldi iteration and the initiation $\mathbf{U} = \mathbf{r}$ produce comparable convergence for the IDR(s)₂ algorithm. Because of the comparable convergence it is hard to choose between the two initiations of \mathbf{U} .

Experimenting with the matrices MEIER01 and TFQMR001 for the IDR(s)₁ algorithm we found that the matrices \mathbf{U} initiated as $\mathbf{U} = \mathbf{r}$ and \mathbf{U} initiated by Arnoldi also produce comparable convergence. For the IDR(s)₁ the initial matrices $\mathbf{U} = \tilde{\mathbf{R}}_0$, $\mathbf{S} = \mathbf{A}\mathbf{U}$ and $\mathbf{U} = \tilde{\mathbf{R}}_0$, $\mathbf{S} = \mathbf{A}\mathbf{U}$, $\tilde{\mathbf{R}}_0 = \mathbf{S}$ also produce a poor convergence.

From now on in experiments where we will make use of the IDR(s) algorithm, the matrix \mathbf{U} will be initiated by the Arnoldi iteration or \mathbf{U} will be initiated as $\mathbf{U} = \mathbf{r}$. We do this because these two initiations for \mathbf{U} give the best convergence of the IDR(s) algorithm.

3.4.4 Comparing the IDR(s)₁ and the IDR(s)₂ algorithm

We want to select the IDR(s) algorithm which produces the best convergence behaviour. For the selection we need to compare the convergence behaviour of the IDR(s)₁ and the IDR(s)₂ algorithm.

By only experimenting with the diagonal matrix $\mathbf{A} = \text{diag}(1 : 100)$, it is difficult to decide which of the two IDR(s) algorithms is more effective. Therefore we experiment with more interesting matrices like the TFQMR001 matrix.

We present figure 6 as a nice example where the difference of the convergence behaviour is visible for the IDR(s)₁ and the IDR(s)₂ algorithm. In this example matrix \mathbf{A} is the TFQMR001 matrix and subspace $s = 8$. The matrix \mathbf{U} is initiated by the Arnoldi iteration.

The convergence of both the IDR(s)₁ and the IDR(s)₂ algorithm runs close with the GMRES. When the residual norm is about 10^{-9} , the GMRES uses 139 MV's. At this same residual norm the IDR(s)₁ algorithms uses 172 MV's and the IDR(s)₂ algorithm uses 155 MV's. From these results we get that the convergence of the IDR(s)₂ algorithm is better than the IDR(s)₁ algorithm. Figure 6 shows that the convergence of the IDR(s)₂ is better than the convergence of the IDR(s)₁ algorithm.

Comparing the convergence of the IDR(s)₁ with the IDR(s)₂, the convergence of IDR(s)₁ shows the

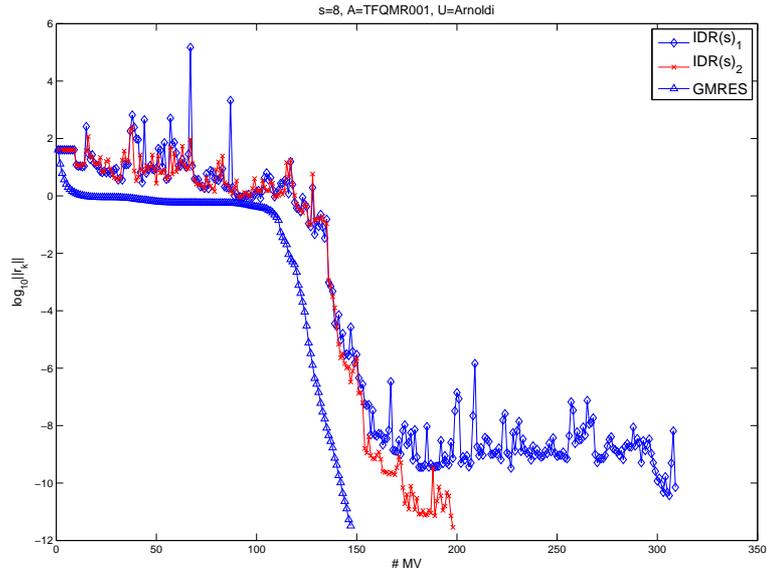


Figure 6: *The convergence of the $IDR(s)_2$ algorithm is better than the convergence of the $IDR(s)_1$ algorithm. $\mathbf{A} = \text{TFQMR001}$, \mathbf{U} is initiated by Arnoldi and $s = 8$. The existence of the high peaks in the convergence of the $IDR(s)_1$ leads to errors and causes the convergence to stop at early stage.*

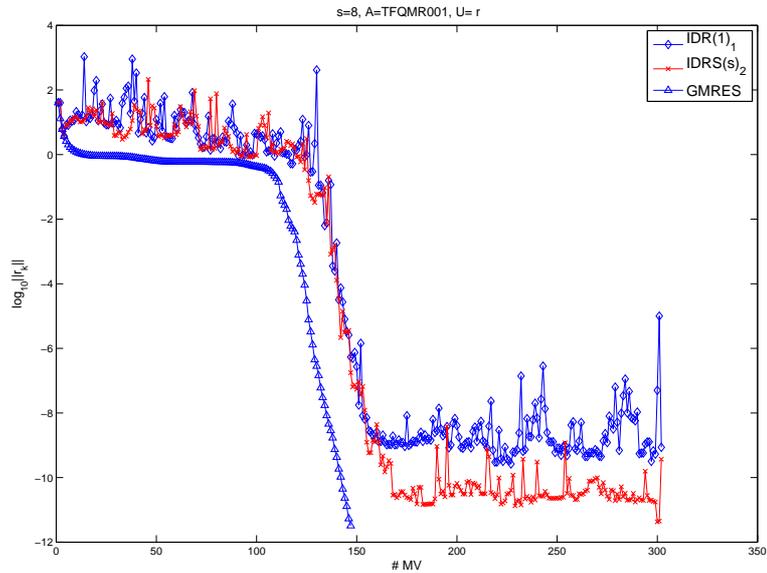


Figure 7: *The convergence of the $IDR(s)_2$ algorithm is better than the convergence of the $IDR(s)_1$ algorithm. $\mathbf{A} = \text{TFQMR001}$, \mathbf{U} is initiated as $\mathbf{U} = \mathbf{r}$ and $s = 8$. Again we see high peaks in the convergence of the $IDR(s)_1$ that leads to errors in further residual steps and effects the convergence of the $IDR(s)_1$ algorithm.*

existence of high peaks. The development of such a peak leads to errors in further residual steps [7, Section 3.1].

Figure 6 shows a high peak where the absolute difference between the residual norms is about 10^4 . In the next steps of the $\text{IDR}(s)_1$ algorithm this relatively large error affects the new residuals. The errors in the residuals that follows can become bigger.

The existence of a relatively big error in the beginning steps of the convergence of the $\text{IDR}(s)_1$ affects the rest of the convergence and causes the $\text{IDR}(s)_1$ to terminate at an early stage.

When the matrix \mathbf{U} is initiated as $\mathbf{U} = \mathbf{r}$, the convergence of the $\text{IDR}(s)_1$ algorithm also show relatively high peaks, compared with the convergence of the $\text{IDR}(s)_2$ algorithm, see figure 7. The peaks lead to errors in further residual steps and effects the convergence of the $\text{IDR}(s)_1$ algorithm. The $\text{IDR}(s)_2$ has again a better convergence when we use the initiation $\mathbf{U} = \mathbf{r}$.

Because we found that the $\text{IDR}(s)_2$ algorithm converges better than the $\text{IDR}(s)_1$ algorithm for \mathbf{U} initiated as $\mathbf{U} = \mathbf{r}$ and \mathbf{U} initiated by the Arnoldi, we will use the $\text{IDR}(s)_2$ algorithm in further examples.

3.5 The convergence behaviour of the $\text{IDR}(s)$ algorithm for growing subspace s

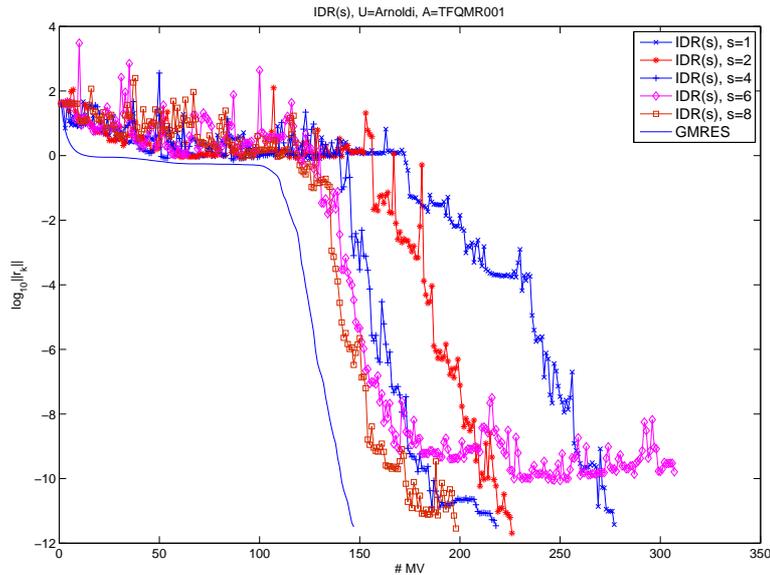


Figure 8: When the subspace s gets bigger, the convergence of the $\text{IDR}(s)$ algorithm becomes close to the GMRES. For this example we choose the TFQMR001 matrix and \mathbf{U} is initiated by the Arnoldi iteration.

For the experiments in the previous sections, we worked with a relative big subspace for the $\text{IDR}(s)$ algorithm, where $s = 8$. The reason why we did not use small subspaces is because of the con-

vergence behaviour. If the subspace gets bigger, usually the convergence of the IDR(s) algorithm becomes better.

Figure 8 shows the relation between the convergence behaviour and the growing of subspaces. For this experiment the matrix TFQMR001 is used, for different subspace s and \mathbf{U} is initiated by the Arnoldi iteration. This example shows that the convergence gets better when the subspace gets bigger. The better the convergence of the IDR(s) algorithm, the closer the graph of the IDR(s) gets to the GMRES.

3.5.1 Stability

The stability of the the IDR(s) algorithm depends on the the number of the iteration steps (MV's) and how accurate the algorithm is. When the IDR(s) algorithm is stable, the algorithm will also result in a good convergence, but this is also dependent on the test problem. For the IDR(s) method, where $s = 1$, there is a relation between the eigenvalues of a k -dimension matrix and the convergence.

The eigenvalues of the diagonal matrix \mathbf{A} are given as: $\mathbf{A}^* = \mathbf{A} = \text{diag}(\lambda_1, \dots, \lambda_k)$. By applying Lanczos, we get in step k a tridiagonal matrix \mathbf{T}_k , where the eigenvalues of this matrix is given as: μ_1^k, \dots, μ_j^k . The eigenvalues of the matrix \mathbf{T}_k are the Ritz values of the matrix \mathbf{A} . When the Ritz value $\mu_j^{(k,\lambda)} \sim 0$, peaks in the k -step of the convergence history will appear in Bi-CG based methods: these peaks correspond to small Ritz values.

To obtain a convergence with peaks we can choose $\mu_j^{(k,\lambda)}$ such that $\mu_j^{(k,0)} - \lambda \sim 0$.

The Bi-Lanczos method can be used to find the Ritz values for the IDR(s) method, where $s = 1$. For $s > 1$ there is yet no theory about the convergence of the Ritz-values.

The Bi-Lanczos method uses the Bi-CG polynomial to compute the Ritz values. For $s = 1$ the Bi-CG polynomial can be computed, but when $s > 1$ there is no theoretical information available for the computation of the Bi-CG polynomials. Therefor we can not say anything about the stability of the IDR(s) method.

4 Bi-CGSTAB, IDR and the original IDR algorithm

In this chapter we will discuss a variant of the Bi-CGSTAB algorithm [3], the IDR algorithm [3] and the original IDR algorithm [4]. We will see that the Bi-CGSTAB is equivalent with the IDR algorithm. And we will also show that the IDR(s) algorithm is equal to the IDR algorithm for $s = 1$.

The stability of the original IDR method will also be tested.

4.1 The Bi-CGSTAB and the IDR algorithm

For $s = 1$ the IDR and the Bi-CGSTAB are closely related. There is a theorem, [3, Th. 4.2], that suggests that the IDR and the Bi-CGSTAB are related. From the previous sections we know that residual vector of the IDR belongs to \mathcal{G}_k .

From the theorem it can be shown that the residual vector of the Bi-CGSTAB, $\mathbf{r}_k^{Bi-CGSTAB}$, also belongs to \mathcal{G}_k , what creates a relation between the Bi-CGSTAB and the IDR. More details are given further.

Theorem 4.1 *Let $\tilde{\mathbf{R}}_0$, (μ_j) and \mathcal{G} be as in Theorem 3.1. Consider the polynomial p of degree k given by $p_k(\lambda) \equiv \prod_{j=1}^k (\mu_j - \lambda)$ ($\lambda \in \mathbb{C}$). Then*

$$\mathcal{G}_k = \{p_k(\mathbf{A})\mathbf{v} \mid \mathbf{v} \perp \mathcal{K}_k(\mathbf{A}^*\tilde{\mathbf{R}}_0)\} \quad (3)$$

Proof. The claim for $k = 0$ is trivial. When $k = 0$, we get that $\mathcal{G}_0 \equiv \mathbb{C}^n$. $\mathcal{K}_0(\mathbf{A}^*\tilde{\mathbf{R}}_0) = \{\mathbf{0}\}$ and $\{\mathbf{v} \mid \mathbf{v} \perp \mathcal{K}_0\} = \mathbb{C}^n$. Since $p_0 = \mu_0$, we get that $\mathcal{G}_0 = \{\mu_0\mathbf{v} \mid \mathbf{v} \perp \mathcal{K}_0\} = \mathbb{C}^n$.

We use an induction argument to prove the theorem. Assume that (3) is correct. Then,

$$\mathcal{G}_k \cap \tilde{\mathbf{R}}_0^\perp = \{p_k(\mathbf{A})\mathbf{v} \mid p_k(\mathbf{A})\mathbf{v} \perp \tilde{\mathbf{R}}_0, \mathbf{v} \perp \mathcal{K}_k(\mathbf{A}^*\tilde{\mathbf{R}}_0)\}.$$

Since $\mathbf{v} \perp \mathcal{K}_k(\mathbf{A}^*\tilde{\mathbf{R}}_0)$, we have that $q(\mathbf{A})\mathbf{v} \perp \tilde{\mathbf{R}}_0$ for all polynomials q of degree $< k$. Hence $p_k(\mathbf{A})\mathbf{v} \perp \tilde{\mathbf{R}}_0$ if and only if $\mathbf{A}^k\mathbf{v} \perp \tilde{\mathbf{R}}_0$, which is equivalent to $\mathbf{v} \perp (\mathbf{A}^*)^k\tilde{\mathbf{R}}_0$, or equivalently, $\mathbf{v} \perp (\mathbf{A}^*)^k\tilde{\mathbf{R}}_0\vec{\gamma}_k$ for all s vectors $\vec{\gamma}_k$. Apparently

$$\mathcal{G}_k \cap \tilde{\mathbf{R}}_0^\perp = \{p_k(\mathbf{A})\mathbf{v} \mid \mathbf{v} \perp \mathcal{K}_{k+1}(\mathbf{A}^*\tilde{\mathbf{R}}_0)\},$$

whence

$$\mathcal{G}_{k+1} = (\mu_k\mathbf{I} - \mathbf{A})(\mathcal{G}_k \cap \tilde{\mathbf{R}}_0^\perp) = \{(\mu_k\mathbf{I} - \mathbf{A})p_k(\mathbf{A})\mathbf{v} \mid \mathbf{v} \perp \mathcal{K}_{k+1}(\mathbf{A}^*\tilde{\mathbf{R}}_0)\}.$$

Since $(\mu_{k+1}\mathbf{I} - \mathbf{A})p_k(\mathbf{A}) = p_{k+1}(\mathbf{A})$ this proves the theorem. \square

We now will describe how the residual vector in the Bi-CGSTAB is created, followed by the derivation of the Bi-CGSTAB algorithm and the IDR algorithm. Instead of $\tilde{\mathbf{R}}_0$ we will use the notation $\tilde{\mathbf{r}}_0$. As we know, the Bi-CGSTAB is a transpose free variant of the Bi-CG method. The k th Bi-CG residual \mathbf{r}_k^{Bi-CG} is of the form

$$\mathbf{r}_k^{Bi-CG} = q_k(\mathbf{A})\mathbf{r}_0 \quad (4)$$

with q_k a polynomial of degree k such that

$$q_k(0) = 1 \quad \text{and} \quad q_k(\mathbf{A})\mathbf{r}_0 \perp \mathcal{K}_k(\mathbf{A}^*, \tilde{\mathbf{R}}_0). \quad (5)$$

By the first property q_k becomes a “residual polynomial”, i.e., $q_k(\mathbf{A})\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_k$ for some \mathbf{x}_k ($\mathbf{x}_k = \mathbf{x}_0 + \tilde{q}(\mathbf{A})\mathbf{r}_0$), where \tilde{q} is such that $(q(\lambda) = 1 - \lambda\tilde{q}(\lambda))$. The two properties in equation (5) determine q_k uniquely.

In the Bi-CGSTAB an auxiliary polynomial p_k of degree k is used for further reduction of the Bi-CG residual:

$$\mathbf{r}_k^{Bi-CGSTAB} = p_k(\mathbf{A})\mathbf{r}_k^{Bi-CG} \quad (6)$$

The polynomial p_k is of the form $p_k(\lambda) = (1 - \omega_k \lambda) \cdots (1 - \omega_1 \lambda)$, with

$$\omega_k = \text{minarg}_\omega \|(1 - \omega \mathbf{A})p_{k-1}(\mathbf{A})\mathbf{r}_k^{Bi-CG}\|_2$$

If the Bi-CGSTAB process does not break off, then the ω_j will be non-zero and with $\mu_j \equiv 1/\omega_j$, we have that $p_k(\lambda) = \frac{1}{\mu_k \cdots \mu_1}(\mu_k - \lambda) \cdots (\mu_1 - \lambda)$. By (4) and Theorem 4.1 it can be concluded that $\mathbf{r}_k^{Bi-CGSTAB}$ belongs to \mathcal{G}_k . From the uniqueness remarks on the form of the Bi-CG residual and the selection of the ω_j in IDR and Bi-CGSTAB (to minimize residual norms), there can be concluded that IDR is equivalent to Bi-CGSTAB for $s = 1$. By assuming exact arithmetic, when the initializations of \mathbf{r}_0 and $\tilde{\mathbf{r}}_0$ are the same, at every second step the same residuals will be produced. The Bi-CG depends on a coupled two term recurrences. The scalars α_k and β_{k+1} in the following equations are determined by the orthogonality condition

$$\begin{aligned} \mathbf{r}_{k+1}^{Bi-CG} &= \mathbf{r}_k^{Bi-CG} - \mathbf{c}_{k+1}^{Bi-CG} \alpha_k \perp \tilde{\mathbf{r}}_k \\ \mathbf{u}_{k+1}^{Bi-CG} &= \mathbf{r}_{k+1}^{Bi-CG} - \mathbf{u}_k^{Bi-CG} \beta_{k+1} \end{aligned} \quad \text{such that} \quad \mathbf{c}_{k+1}^{Bi-CG} \equiv \mathbf{A} \mathbf{u}_{k+1}^{Bi-CG} \perp \tilde{\mathbf{r}}_k. \quad (7)$$

$\tilde{\mathbf{r}}_0, \dots, \tilde{\mathbf{r}}_j$ is a basis of $\mathcal{K}_{j+1}(\mathbf{A}^*, \tilde{\mathbf{r}}_0)$ for all $j \leq k$. If we multiply the second recurrence relation with \mathbf{A} we get the following equation:

$$\begin{aligned} \mathbf{r}_{k+1}^{Bi-CG} &= \mathbf{r}_k^{Bi-CG} - \mathbf{c}_{k+1}^{Bi-CG} \alpha_k \perp \tilde{\mathbf{r}}_k \\ \mathbf{c}_{k+1}^{Bi-CG} &= \mathbf{A} \mathbf{r}_{k+1}^{Bi-CG} - \mathbf{c}_k^{Bi-CG} \beta_{k+1} \perp \tilde{\mathbf{r}}_k. \end{aligned} \quad (8)$$

With these equations \mathbf{c}_{k+1}^{Bi-CG} and \mathbf{u}_{k+1}^{Bi-CG} can be computed. At first β_{k+1} and \mathbf{c}_{k+1}^{Bi-CG} is computed, where $\mathbf{c}_{k+1}^{Bi-CG} \perp \tilde{\mathbf{r}}_k$. Then \mathbf{u}_{k+1}^{Bi-CG} , where $\mathbf{u}_{k+1}^{Bi-CG} = \mathbf{r}_{k+1}^{Bi-CG} - \mathbf{u}_{k+1}^{Bi-CG} \beta_{k+1}$, can be updated, and this will cost one additional AXPY (per step). Because this formulation is close to the derivation of the IDR method, we will use this to derive the Bi-CGSTAB. By an induction argument, (8) indicates that local bi-orthogonality implies global bi-orthogonality where $\mathbf{r}_{k+1}^{Bi-CG}, \mathbf{c}_{k+1}^{Bi-CG} \perp \tilde{\mathbf{r}}_k, \tilde{\mathbf{r}}_{k-1}, \tilde{\mathbf{r}}_0$

Bi-CGSTAB can be derived from the Bi-CG method when we put $\mathbf{P}_k \equiv p_k(\mathbf{A})$, where p_k is a polynomial of degree k with $p_k(0) = 1$. If we take $\tilde{\mathbf{r}}_k = \bar{p}_k(\mathbf{A}^*)\tilde{\mathbf{r}}_0$, with a suitable α_k , then the first relation of (8) gives us

$$\mathbf{v}_k \equiv \mathbf{P}_k \mathbf{r}_{k+1}^{Bi-CG} = \mathbf{P}_k \mathbf{r}_k^{Bi-CG} - \mathbf{P}_k \mathbf{c}_k^{Bi-CG} \alpha_k \perp \tilde{\mathbf{r}}_0.$$

When $\mathbf{r}_k \equiv \mathbf{P}_k \mathbf{r}_k^{Bi-CG}$ and $\mathbf{c}_k \equiv \mathbf{P}_k \mathbf{c}_k^{Bi-CG}$, we have that

$$\mathbf{v}_k = \mathbf{r}_k - \mathbf{c}_k \alpha_k \perp \tilde{\mathbf{r}}_0.$$

$p_{k+1}(\lambda) \equiv (1 - \omega_{k+1} \lambda) p_k(\lambda)$ allows us to get the following relation

$$\mathbf{r}_{k+1} = \mathbf{P}_k \mathbf{r}_{k+1}^{Bi-CG} = (\mathbf{I} - \omega_{k+1} \mathbf{A}) \mathbf{P}_k \mathbf{r}_{k+1}^{Bi-CG} = (\mathbf{I} - \omega_{k+1} \mathbf{A}) \mathbf{v}_k.$$

Further we can compute $\mathbf{P}_k \mathbf{c}_{k+1}^{Bi-CG}$ from the second relation of (8).

$$\mathbf{P}_k \mathbf{c}_{k+1}^{Bi-CG} = \mathbf{A} \mathbf{P}_k \mathbf{r}_{k+1}^{Bi-CG} - \mathbf{P}_k \mathbf{c}_k^{Bi-CG} \beta_k = \mathbf{A} \mathbf{v}_k - \mathbf{c}_k \beta_k \perp \tilde{\mathbf{r}}_0$$

Now with $\mathbf{u}_k \equiv \mathbf{P}_k \mathbf{u}_k^{Bi-CG}$ and $\mathbf{w} = \mathbf{v}_k - \mathbf{u}_k \beta_k$, gives us $\mathbf{A}\mathbf{w} = \mathbf{P}_k \mathbf{c}_{k+1}^{Bi-CG}$ and $\mathbf{u}_{k+1} = (\mathbf{I} - \omega_{k+1} \mathbf{A})\mathbf{w} = \mathbf{w} - \omega_{k+1} \mathbf{A}\mathbf{w}$. Alg. 4.1 gives us this variant of the Bi-CGSTAB algorithm. This Bi-CGSTAB algorithm is different than the Bi-CGSTAB algorithm given in article [1]. The Bi-CGSTAB algorithm given here computes the vector \mathbf{u}_k in a different way than it is done in the original Bi-CGSTAB. In this variant the Bi-CGSTAB depends on two explicit orthogonalization on $\tilde{\mathbf{r}}_0$. In the standard Bi-CGSTAB one of the orthogonalization is implicit. Compared with the standard Bi-CGSTAB the orthogonalization is made explicit at the cost of one additional AXPY.

```

Select an  $\mathbf{x}_0$ 
Select an  $\tilde{\mathbf{r}}_0$ 
Compute  $\mathbf{r} = \mathbf{b} - \mathbf{A}\mathbf{x}$ , set  $\mathbf{u} = \mathbf{r}$ 
repeat
  Compute  $\mathbf{A}\mathbf{u}$ 
   $\alpha$  such that  $\mathbf{v} = \mathbf{r} - \mathbf{A}\mathbf{u}\alpha \perp \tilde{\mathbf{r}}_0$ 
  Compute  $\mathbf{A}\mathbf{v}$ 
  Select  $\omega$ ,  $\mathbf{r} = \mathbf{v} - \omega\mathbf{A}\mathbf{v}$ 
   $\beta$  such that  $\mathbf{A}\mathbf{w} = \mathbf{A}\mathbf{v} - \mathbf{A}\mathbf{u}\beta \perp \tilde{\mathbf{r}}_0$ 
   $\mathbf{w} = \mathbf{v} - \mathbf{u}\beta$ 
   $\mathbf{u} = \mathbf{w} - \omega\mathbf{A}\mathbf{w}$ 
end repeat

```

Alg. 4.1: Bi-CGSTAB [3].

```

Select an  $\mathbf{x}_0$ 
Select an  $\tilde{\mathbf{r}}_0$ 
Compute  $\mathbf{r} = \mathbf{b} - \mathbf{A}\mathbf{x}$ , set  $\mathbf{u} = \mathbf{r}$ 
repeat
  Compute  $\mathbf{A}\mathbf{u}$ ,  $\mathbf{r}' = \mathbf{r}$ 
   $\alpha$  such that  $\mathbf{v} = \mathbf{r} - \mathbf{A}\mathbf{u}\alpha \perp \tilde{\mathbf{r}}_0$ 
  Compute  $\mathbf{A}\mathbf{v}$ 
  Select  $\omega$ ,  $\mathbf{r} = \mathbf{v} - \omega\mathbf{A}\mathbf{v}$ 
   $\mathbf{s} = \mathbf{r}' - \mathbf{r}$ ,  $\mathbf{u}' = \mathbf{u}\alpha + \omega\mathbf{v}$ 
   $\beta$  such that  $\mathbf{v}' = \mathbf{r} - \mathbf{s}\beta \perp \tilde{\mathbf{r}}_0$ 
   $\mathbf{u} = \mathbf{u}'\beta - \omega\mathbf{v}'$ 
end repeat

```

Alg. 4.2: IDR [3].

The IDR algorithm for $s = 1$ is given in Alg. 4.2.

The IDR algorithm is mathematically equivalent with the IDR(s) for $s = 1$ (see Section 4.2). In Section 4.3 we will see that the IDR algorithm is also related with the original IDR algorithm [4].

Observing the IDR and the Bi-CGSTAB algorithm, we see that the computing of the vectors \mathbf{r} and \mathbf{v} are theoretically the same. The two vectors \mathbf{r} and \mathbf{v} are computed the same time and from Alg. 4.1 and Alg. 4.2 it is clear that the IDR and the Bi-CGSTAB are mathematically equivalent.

4.1.1 Convergence of the IDR and the Bi-CGSTAB algorithm

In figure 9 we show the convergence of the IDR (Alg. 4.2) and the Bi-CGSTAB algorithm (Alg. 4.1). The convergence of the Bi-CGSTAB is equivalent with the convergence of the IDR, except for the ending of the convergence. The computation of the vectors \mathbf{v} and \mathbf{r} for the two algorithms are theoretically the same. Both vectors are used to generate convergence for the IDR and the Bi-CGSTAB, and this gives an equivalent convergence.

Figure 9 shows that after an error rate of 10^{-11} the convergence of these two algorithms slightly begins to vary from each other. This could be caused by numerical errors.

We used the diagonal matrix $\mathbf{A} = \text{diag}(1 : 100)$. The $n \times s$ matrix $\tilde{\mathbf{R}}_0$ is a randomly chosen and orthogonalized, where $n = 100$ and $s = 1$.

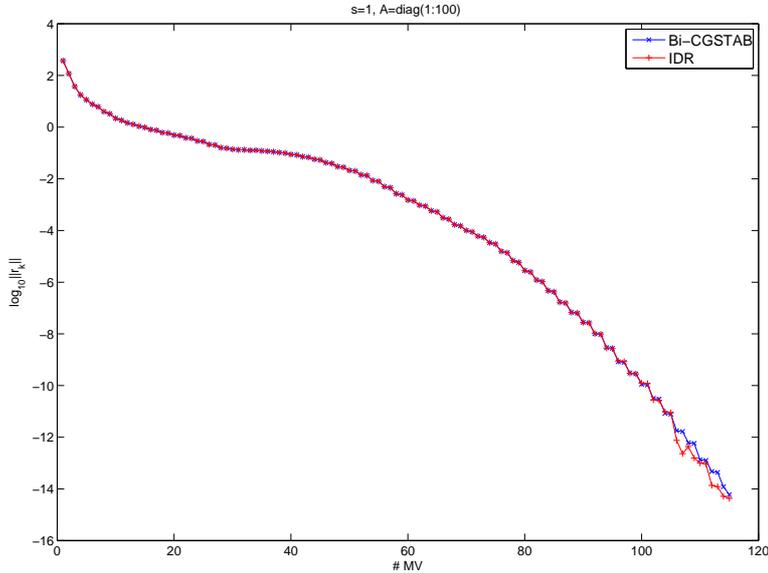


Figure 9: We plot the vectors \mathbf{v} and \mathbf{r} for the IDR and the Bi-CGSTAB algorithm. This gives an equivalent convergence for the IDR and the Bi-CGSTAB till a residual norm of 10^{-11} .

4.2 The IDR and the IDR(s) for $s = 1$

For $s = 1$ the IDR(s) algorithm (Alg. 3.1) algorithm is mathematically equivalent with the IDR algorithm (Alg. 4.2). The two algorithms are equivalent because the vectors \mathbf{v} and \mathbf{r} of the IDR are the same as the vectors \mathbf{v} and \mathbf{r} calculated in the IDR(s) for $s = 1$. For $s = 1$ the residuals of the IDR(s) lie in the space $\mathcal{G}_1 = (\mathbf{I} - \omega \mathbf{A})(\mathcal{G}_0 \cap \tilde{\mathbf{R}}_0^\perp)$

For the IDR(s) first the matrices \mathbf{U} and $\tilde{\mathbf{R}}_0$ are selected, followed by the next computations:

$$\begin{aligned}\mathbf{S} &= \mathbf{A}\mathbf{U} \\ \gamma &= \tilde{\mathbf{R}}_0^* \mathbf{S} \tilde{\mathbf{R}}_0^* \mathbf{r}\end{aligned}$$

$$\begin{aligned}\boxed{\mathbf{v}} &= \mathbf{r}_{old} - \mathbf{S}\vec{\gamma}, & \text{here } \mathbf{r} \in \mathcal{G}_k, \mathbf{v} \in \mathcal{G}_k \cap \mathbf{R}_0^\perp \\ \mathbf{x} &= \mathbf{x} + \mathbf{U}\vec{\gamma} \\ \mathbf{c} &= \mathbf{A}\mathbf{v}, \omega = \mathbf{c}^* \mathbf{v} \backslash \mathbf{c}^* \mathbf{c} \\ \mathbf{U}e_1 &= \mathbf{U}e_1 \vec{\gamma} + \omega \mathbf{v} \\ \boxed{\mathbf{r}}_{new} &= \mathbf{v} - \omega \mathbf{c} & \mathbf{r}_{new} = (\mathbf{I} - \omega_k \mathbf{A}) \mathbf{v} (\mathcal{G}_k \cap \tilde{\mathbf{R}}_0^\perp) = \mathcal{G}_{k+1} \\ \mathbf{S}e_1 &= \mathbf{r}_{old} - \mathbf{r}_{new} = \mathbf{A}\mathbf{U}e_1 & \mathbf{r}_{old} \in \mathcal{G}_k \text{ and } \mathbf{r}_{new} \in \mathcal{G}_{k+1}, \text{ so } \mathbf{S}e_1 \in \mathcal{G}_k \\ \mathbf{x} &= \mathbf{U} + \omega \mathbf{v}\end{aligned}$$

Comparing the IDR(s) with the IDR algorithm we can see the similarities:

For the IDR $\mathbf{A}\mathbf{U}$ is first computed followed by the computation of $\alpha = \mathbf{r}_0^* \mathbf{A}\mathbf{u} \backslash \mathbf{r}_0^* \mathbf{r}$, where in the IDR(s) also first $\mathbf{S} = \mathbf{A}\mathbf{u}$ and $\gamma = \tilde{\mathbf{R}}_0^* \mathbf{S} \tilde{\mathbf{R}}_0^* \mathbf{r}$ are computed. The scalars α and γ are just the same, where $\tilde{\mathbf{R}}_0^*$ and \mathbf{r}_0^* are the same vectors.

The vector $\mathbf{v} = \mathbf{r} - \mathbf{A}\mathbf{u}\alpha$ in IDR is the same as vector $\mathbf{v} = \mathbf{r}_{old} - \mathbf{S}\vec{\gamma}$ of IDR(s) where $\mathbf{r}_{old} = \mathbf{r}$, $\mathbf{S} = \mathbf{A}\mathbf{u}$ and $\alpha = \gamma$.

The computation of $\omega = \mathbf{A}\mathbf{v}^* \mathbf{v} \backslash \mathbf{A}\mathbf{v}^* \mathbf{A}\mathbf{v}$ in IDR is the same as ω in IDR(s) with $\mathbf{c} = \mathbf{A}\mathbf{v}$.

The update of \mathbf{U} in IDR(s) is similar to the computing of \mathbf{u}' in IDR, where $\mathbf{u}' = \mathbf{u}\alpha + \omega \mathbf{v}$.

Then we compute the vector \mathbf{r}_{new} in IDR(s) which is similar to the vector \mathbf{r} in IDR, where $\mathbf{r} = \mathbf{v} - \omega \mathbf{A}\mathbf{v}$. In the IDR(s) we compute the vector $\mathbf{S} = \mathbf{r}_{old} - \mathbf{r}_{new}$ which is the same computation in the IDR for $\mathbf{s} = \mathbf{r}' - \mathbf{r}$ where $\mathbf{r}_{old} = \mathbf{r}$ and $\mathbf{r}_{new} = \mathbf{r}'$.

In the last step of the IDR(s) we update \mathbf{x} by $\mathbf{x} = \mathbf{U}e_1 - \omega \mathbf{v}$. In the IDR finally \mathbf{u} computed as $\mathbf{u} = \mathbf{u}'\beta + \omega \mathbf{v}'$ and β such that $\mathbf{v}' - \mathbf{r} - \mathbf{s}\beta \perp \tilde{\mathbf{r}}_0$.

For the IDR algorithm (Alg. 4.2) we do comparable computation as for the IDR(s) (Alg. 3.1) for $s = 1$, what implies that the IDR and the IDR(s) are mathematically equivalent. The vectors \mathbf{v} and \mathbf{r} are computed in both the IDR and the IDR(s) for $s = 1$. Generally the vectors \mathbf{v} and \mathbf{r} of the IDR algorithm are used to create a convergence plot. For the IDR(s) algorithm only the vector \mathbf{r} is stored to create a convergence plot.

In figure 10 the convergence of the IDR (Alg. 4.2) and the IDR(s) (Alg. 3.1) is plotted. For the IDR algorithm we store the residuals vectors \mathbf{v} and \mathbf{r} and for the IDR(s) algorithm we store the residual vector \mathbf{r} to obtain convergence.

Because different residual vectors are stored for the IDR and the IDR(s), the convergence plots of the two algorithms is not exactly the same. The convergence plot shows us that the IDR and the IDR(s) algorithm produce the same residuals at every $s + 1$ step till an residual norm of 10^{-9} . After this residual norm the IDR and the IDR(s) are affected by numerical errors, that lead to different residuals at every $s + 1$ step.

We used the diagonal matrix $\mathbf{A} = \text{diag}(1 : 100)$. For the IDR(s) the matrix \mathbf{U} is initiated by the Arnoldi iteration. The $n \times s$ matrix $\tilde{\mathbf{R}}_0$ is a randomly chosen and orthogonalized, where $n = 100$

and $s = 1$.

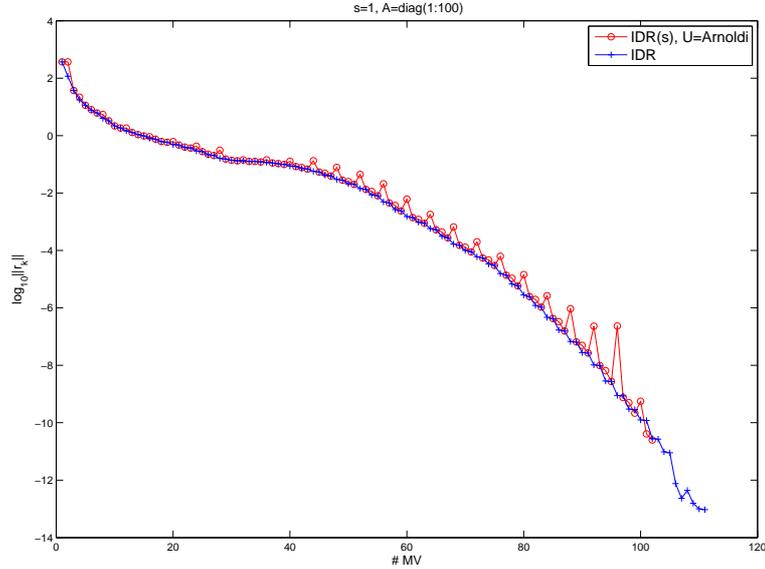


Figure 10: We plot the vectors \mathbf{v} and \mathbf{r} for the IDR and the vector \mathbf{r} for the IDR(s) algorithm ($s = 1$). The IDR and the IDR(s) produce the same residuals at every $s + 1$ step till a residual norm of 10^{-9} .

4.3 The original IDR algorithm

We present the original IDR algorithm (Alg. 4.3) as proposed by P. Sonneveld [4]. This IDR algorithm creates $n \times s$ matrices \mathbf{U} and \mathbf{S} that are zero in the beginning. For the IDR algorithm (Alg. 4.2) that we described in the previous section and the IDR(s) algorithm (Alg. 3.1), the $n \times s$ matrices \mathbf{U} and \mathbf{S} are also created, but not set zero. For the IDR algorithm (Alg. 4.2) we take $\mathbf{U} = \mathbf{r}$, and $\mathbf{S} = \mathbf{A}\mathbf{U}$ and for the IDR(s) algorithm \mathbf{U} is initiated in four different ways, where $\mathbf{U} \neq 0$ and $\mathbf{S} = \mathbf{A}\mathbf{U}$. When comparing the original IDR algorithm (Alg. 4.3) and the IDR algorithm (Alg. 4.2) we see that there are no major differences. Only the computation of α does not take place simultaneously. These two algorithms produce the same vectors \mathbf{v} and \mathbf{r} .

The IDR algorithm of Sonneveld starts with $\mathbf{U} = 0$ and $\mathbf{S} = 0$, $m = 1$ and $\mathbf{r} = \mathbf{A}\mathbf{x} - \mathbf{b}$. For $\mathbf{v} = \mathbf{r} - \alpha\mathbf{S}$, we get that $\mathbf{v} = \mathbf{r}$, because $\mathbf{S} = 0$. $\mathbf{c} = \mathbf{A}\mathbf{v}$ and ω is computed for $k = 0$, where $\omega = (\mathbf{c}^*\mathbf{v})/(\mathbf{c}^*\mathbf{c})$. We have residual $\mathbf{r} = \mathbf{r} - \omega\mathbf{c}$ and $\mathbf{S} = \mathbf{d}\mathbf{r}$ where $\mathbf{d}\mathbf{r} = \omega\mathbf{c}$. At the end α is computed.

In the next step \mathbf{v} is computed and becomes: $\mathbf{v} = \mathbf{r} - \alpha(\omega\mathbf{c})$. This vector is the same as vector $\mathbf{v} = \mathbf{r} - \mathbf{A}\mathbf{u}\alpha$ produced in the IDR algorithm (Alg. 4.2).

After the computation of the vector \mathbf{v} , $\mathbf{c} = \mathbf{A}\mathbf{v}$ is computed followed by ω . This leads us to the computation of the vector \mathbf{r} , where $\mathbf{r} = \mathbf{r} - \omega\mathbf{c}$. This vector is the same vector $\mathbf{r} = \mathbf{v} - \omega\mathbf{A}\mathbf{v}$ that

is produced in the IDR algorithm (Alg. 4.2).

```

Select an  $\mathbf{x}_0$ 
Select  $n \times s$  matrices  $\tilde{\mathbf{R}}_0$ ,  $\mathbf{U}$  and  $\mathbf{S}$ 
where  $\mathbf{U} = \mathbf{0}$  and  $\mathbf{S} = \mathbf{0}$ 
 $\mathbf{r} = \mathbf{Ax} - \mathbf{b}$ 
 $m = 1, \alpha = 0, k = 0$ 
repeat
   $\mathbf{v} = \mathbf{r} - \alpha\mathbf{S}, \mathbf{c} = \mathbf{Av}$ 
  if  $m = 0$  or  $k = 0$ 
     $\omega \leftarrow \mathbf{c}^*\mathbf{v}/\mathbf{c}^*\mathbf{c}, k = 1$ 
  end if
   $\mathbf{dx} = \alpha\mathbf{U} + \omega\mathbf{v}$ 
   $\mathbf{dr} = \alpha\mathbf{S} + \omega\mathbf{c}$ 
   $\mathbf{x} = \mathbf{x} + \mathbf{dx}$ 
   $\mathbf{r} = \mathbf{r} + \mathbf{dr}$ 
  if  $m = 1$ 
     $\mathbf{S} = \mathbf{dr}$ 
     $\mathbf{U} = \mathbf{dx}$ 
  end if
   $\alpha = (\tilde{\mathbf{R}}_0^*\mathbf{r})/(\tilde{\mathbf{R}}_0^*\mathbf{S})$ 
   $m = 1 - m$ 
end repeat

```

Alg. 4.3: *The original IDR* [4].

4.3.1 Comparing the IDR, the original IDR and the IDR(s) algorithm

In figure 11 we present the convergence plot of the original IDR algorithm (Alg. 4.3), the IDR algorithm (Alg. 4.2), the IDR(s)algorithm (Alg. 3.1) and the GMRES. Because the IDR algorithm and the Bi-CGSTAB algorithm are equivalent, the convergence of the Bi-CGSTAB will not be plotted in the figure. For this experiment we want to find out what relation the original IDR algorithm shares with the IDR and the IDR(s) algorithm.

The diagonal matrix $\mathbf{A} = \text{diag}(1 : n)$ is used. $\tilde{\mathbf{R}}_0$ is a randomly chosen orthogonalized $n \times s$ matrix, where $n = 100$ and $s = 1$. For the IDR(s), the matrix \mathbf{U} is initiated by the Arnoldi iteration.

From the results we have that the convergence of the original IDR, the IDR and the IDR(s) algorithm produce the same residuals at every $s + 1$ step. This means that the original IDR and the Bi-CGSTAB algorithm also produce the same residuals at every $s + 1$ step, because the IDR and the Bi-CGSTAB are mathematically equivalent. The original IDR, the IDR and the IDR(s) have the same residual every $s + 1$ step in common because different residual vectors are being plot as explained in Section 4.2.

Figure 11 shows that the original IDR produces a smooth convergence for $\mathbf{A} = \text{diag}(1 : 100)$. The IDR also produces a smooth convergence.

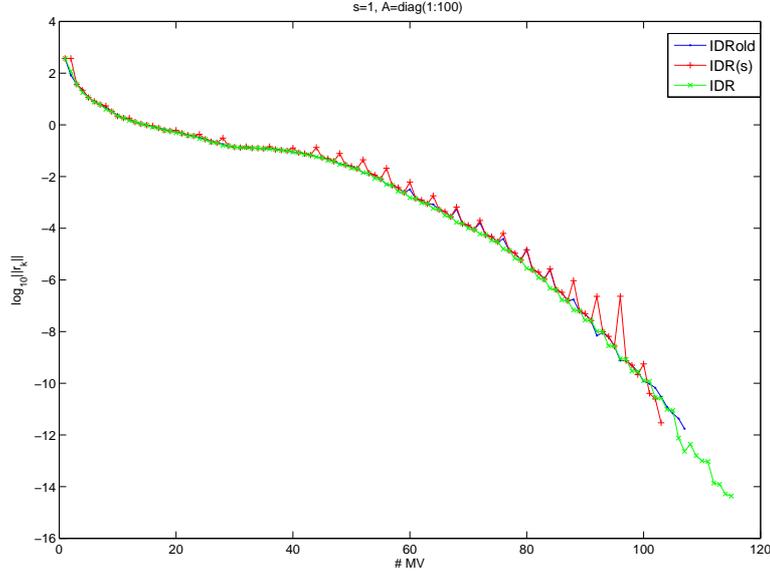


Figure 11: *The original IDR algorithm (IDR_{old}), IDR and the IDR(s) produce the same residuals at every $s + 1$ step (\mathbf{U} initiated by the Arnoldi) for $s = 1$. Matrix $\mathbf{A} = \text{diag}(1 : 100)$.*

To test the stability of the original IDR algorithm we also experimented with the matrix TFQMR001. In figure 12 we present the convergence behaviour of the original IDR and the IDR algorithm where the matrix TFQMR001 is used. The convergence of the original IDR shows high peaks. The high peaks in the convergence of the original IDR leads to loss in convergence rate which explains why the graph of the original IDR stops at an early stage, compared to the graph of the IDR.

The stability of the original IDR also depends on the loss in accuracy. For this we need to measure $\|\mathbf{r}_k\|$ and $\|\mathbf{b} - \mathbf{A}\mathbf{x}_k\|$. For the matrix TFQMR001 we have that $\|\mathbf{r}_k\| = 3.5036 \cdot 10^{-12}$ and $\|\mathbf{b} - \mathbf{A}\mathbf{x}_k\| = 3.1920 \cdot 10^{-12}$. From these result we can conclude that there is no big accuracy loss for the original IDR.

Experimenting with the original IDR algorithm for the MEIER01 we also detect high peaks in the convergence, but here was no remarkable loss in accuracy. For both of the matrices, TFQMR001 and MEIER001, the original IDR algorithm, the IDR, IDR(s) and the Bi-CGSTAB algorithm had the same residuals every $s + 1$ step.

From our results we did not observe that the original IDR algorithm is unstable as stated in article [1], except the high peaks that leads to loss in convergence rate. The convergence of the original IDR algorithm behaves in the same way as the convergence IDR, the Bi-CGSTAB and the IDR(s) algorithm.

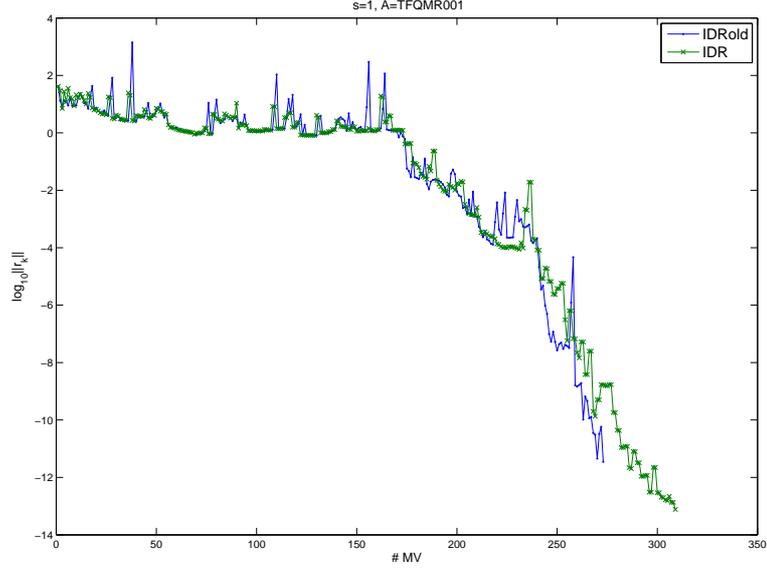


Figure 12: *The original IDR algorithm (IDR_{old}) and the IDR for $s = 1$. Compared to the IDR, the original IDR consists of high peaks, but no unstable convergence, because there is no big loss in accuracy. Matrix $\mathbf{A} = \text{TFQMR001}$.*

5 Bi-CG

We will try to explain how Bi-CG can be formulated for an s -dimensional initial shadow residual in order to understand how an s -dimensional initial shadow residual $\tilde{\mathbf{R}}_0$ can be included in Bi-CGSTAB. We take $\tilde{\mathbf{R}}_0$ as an $n \times s$ matrix of full rank. $\mathcal{K}_k(\mathbf{A}^*, \tilde{\mathbf{R}}_0)$ is the space with all the vectors $\sum_{j < k} (\mathbf{A}^*)^j \tilde{\mathbf{R}}_0 \tilde{\beta}_j$ with $\tilde{\beta}_j$ in \mathbb{C}^s .

$\mathcal{K}_k(\mathbf{A}^*, \tilde{\mathbf{R}}_0)$ is the block Krylov subspace of order k generated by the \mathbf{A}^* and $\tilde{\mathbf{R}}_0$. Choose $\tilde{\mathbf{R}}_i$ as an $n \times s$ matrix where the columns of the matrices $\tilde{\mathbf{R}}_0, \dots, \tilde{\mathbf{R}}_{i-1}$ span the space $\mathcal{K}_i(\mathbf{A}^*, \tilde{\mathbf{R}}_0)$ for all $i \leq k$. Now a residual vector \mathbf{r}_k needs to be constructed in $\mathcal{K}_K(\mathbf{A}, \mathbf{r}_0)$ with $K = ks + 1$ that is orthogonal to $\mathcal{K}_k(\mathbf{A}^*, \tilde{\mathbf{R}}_0)$.

Let \mathbf{C}_0 be the $n \times s$ matrix with columns $\mathbf{A}\mathbf{r}_0, \dots, \mathbf{A}^s\mathbf{r}_0$. So $\mathbf{C}_0 = \mathbf{A}\mathbf{U}_0$, where $\mathbf{U}_0 = [\mathbf{r}_0, \dots, \mathbf{A}^{s-1}\mathbf{r}_0]$ and \mathbf{U}_0 is explicitly available. We have to find a vector $\tilde{\alpha}_0 \in \mathbb{C}^s$ such that $\mathbf{r}_1 \equiv \mathbf{r}_0 - \mathbf{C}_0\tilde{\alpha}_0 \perp \tilde{\mathbf{R}}_0 \in \mathcal{K}_{s+1}(\mathbf{A}, \mathbf{r}_0)$ and update \mathbf{x}_0 , where $\mathbf{x}_1 = \mathbf{x}_0 + \mathbf{U}_0\tilde{\alpha}_0$.

If we take $\sigma_0 \equiv \tilde{\mathbf{R}}_0^* \mathbf{C}_0$, then any vector of the form $\mathbf{w} - \mathbf{C}_0\sigma_0^{-1}\tilde{\mathbf{R}}_0^*\mathbf{w}$ is orthogonal to $\tilde{\mathbf{R}}_0$: $\pi_1 \equiv \mathbf{I} - \mathbf{C}_0\sigma_0^{-1}\tilde{\mathbf{R}}_0^*$ is a skew projection onto the orthogonal complement of $\tilde{\mathbf{R}}_0$. In particular $\mathbf{r}_1 = \pi_1\mathbf{r}_0$. We assume that σ_0 is $s \times s$ non-singular.

For the Bi-CG algorithm, we take $\mathbf{v} = \mathbf{r}_1$, then a $n \times s$ matrix \mathbf{C}_1 that is orthogonal to $\tilde{\mathbf{R}}_0$ can be constructed for $j = 1, \dots, s$ like: $\mathbf{s} = \mathbf{A}\mathbf{v}$ with $\mathbf{s} = \mathbf{s} - \mathbf{C}_0(\sigma_0^{-1}\tilde{\mathbf{R}}_0^*\mathbf{s})$, $\mathbf{C}_1\mathbf{e}_j = \mathbf{s}$ and $\mathbf{v} = \mathbf{s}$.

By $\mathbf{C}_1\mathbf{e}_j = \mathbf{s}$ we point out that the j -column of \mathbf{C}_1 is set to the vector \mathbf{s} and \mathbf{e}_j is thus the j th (s -dimensional) standard basis vector. So \mathbf{C}_1 is orthogonal to $\tilde{\mathbf{R}}_0$ and the columns of \mathbf{C}_1 form a basis

for the Krylov subspace of $\mathcal{K}_s(\pi_1 \mathbf{A}, \pi_1 \mathbf{A} \mathbf{r}_1)$ order s generated by $\mathbf{A}_1 \equiv \pi_1 \mathbf{A} \equiv (\mathbf{I} - \mathbf{C}_0 \sigma_0^{-1} \tilde{\mathbf{R}}_0^*) \mathbf{A}$ and $\mathbf{A}_1 \mathbf{r}_1$. There is also a matrix \mathbf{U}_1 such that $\mathbf{C}_1 = \mathbf{A} \mathbf{U}_1$. The columns of \mathbf{U}_1 can be computed at the same time with the columns of \mathbf{C}_1 : $\mathbf{U}_1 e_j = \mathbf{v} - \mathbf{U}_0 (\sigma_0^{-1} \tilde{\mathbf{R}}_0^* \mathbf{s})$. With $\pi_0 \equiv \mathbf{I} - \mathbf{U}_0 (\sigma_0^{-1} \tilde{\mathbf{R}}_0^* \mathbf{A})$ we have that $\mathbf{A} \pi_0 = \pi_1 \mathbf{A}$ and $\pi_0 \mathbf{v} = \mathbf{v} - \mathbf{U}_0 \sigma_0^{-1} \tilde{\mathbf{R}}_0^* \mathbf{s}$, where $\mathbf{s} = \mathbf{A} \mathbf{v}$. With $\sigma_1 \equiv \tilde{\mathbf{R}}_1^* \mathbf{C}_1$, the vector $\mathbf{r}_2 \equiv \mathbf{r}_1 - \mathbf{C}_1 (\sigma_0^{-1} \tilde{\mathbf{R}}_1^* \mathbf{r}_1)$ is orthogonal to $\tilde{\mathbf{R}}_0$ and also to $\tilde{\mathbf{R}}_1$. It belongs to Krylov subspace $\mathcal{K}_{2s+1}(\mathbf{A}, \mathbf{r}_0)$ and $\mathbf{x}_2 = \mathbf{x}_1 + \mathbf{U}_1 (\sigma_0^{-1} \tilde{\mathbf{R}}_1^* \mathbf{r}_1)$ is the associated approximate solution.

By repetition the residual \mathbf{r}_k can be computed: $\mathbf{r}_k \in \mathcal{K}_{ks+1}(\mathbf{A}, \mathbf{r}_0)$, $\mathbf{r}_k \perp \mathcal{K}_k(\mathbf{A}^*, \tilde{\mathbf{R}}_0)$. The $\vec{\alpha}_k$ and $\vec{\beta}_j \equiv \vec{\beta}_j^{(k)}$ can be computed as $\vec{\alpha}_k = \sigma_k^{-1} (\tilde{\mathbf{R}}_k^* \mathbf{r}_k)$ and $\vec{\beta}_j = \sigma_k^{-1} (\tilde{\mathbf{R}}_k^* \mathbf{s})$, where $\sigma_k \equiv \tilde{\mathbf{R}}_k^* \mathbf{C}_k$. With the computation of the columns of \mathbf{C}_{k+1} , at the same time, the columns of the matrix \mathbf{U}_{k+1} can be computed, and the vector \mathbf{x}_k can also be updated. Here the columns of \mathbf{C}_{k+1} form a Krylov basis of the Krylov subspace generated by $\mathbf{A}_{k+1} \equiv (\mathbf{I} - \mathbf{C}_k \sigma_k^{-1} \tilde{\mathbf{R}}_k^*) \mathbf{A}$ and $\mathbf{A}_{k+1} \mathbf{r}_k$.

```

Select an  $\mathbf{x}_0$ 
Select an  $\tilde{\mathbf{R}}_0$ 
Compute  $\mathbf{r}_0 = \mathbf{b} - \mathbf{A} \mathbf{x}_0$ 
 $k = -1, \sigma_k = \mathbf{I}$ 
Set  $\mathbf{U}_k = 0, \mathbf{C}_k = 0, \tilde{\mathbf{R}}_k = 0$ 
repeat
   $\mathbf{s} = \mathbf{r}_k$ 
  for  $j = 1, \dots, s$ 
     $\mathbf{u} = \mathbf{s}$ , Compute  $\mathbf{s} = \mathbf{A} \mathbf{u}$ 
     $\beta_k = \sigma_k^{-1} (\tilde{\mathbf{R}}_k^* \mathbf{s})$ 
     $\mathbf{u} = \mathbf{u} - \mathbf{U}_k \beta_k, \mathbf{U}_{k+1} e_j = \mathbf{u}$ 
     $\mathbf{s} = \mathbf{s} - \mathbf{C}_k \beta_k, \mathbf{C}_{k+1} e_j = \mathbf{s}$ 
  end for
  if  $k = 1$ 
     $\tilde{\mathbf{R}} = \tilde{\mathbf{R}}_0$ 
  else
     $\mathbf{S} = \mathbf{A}^* \tilde{\mathbf{R}}_0$ 
     $\mu = (\tilde{\mathbf{R}}^* \tilde{\mathbf{R}}) / (\tilde{\mathbf{R}}^* \mathbf{S})$ 
     $\tilde{\mathbf{R}} = \mathbf{S} - (\tilde{\mathbf{R}} \mu)$ 
  end
   $k \leftarrow k + 1$ 
   $\sigma_k = \tilde{\mathbf{R}}_k^* \mathbf{C}_k, \alpha_k = \sigma_k^{-1} (\tilde{\mathbf{R}}_k^* \mathbf{r}_k)$ 
   $\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{U}_k \alpha_k, \mathbf{r}_{k+1} = \mathbf{r}_k - \mathbf{C}_k \alpha_k$ 
end repeat

```

Alg. 5.1: Bi-CG.

The Bi-CG algorithm described above (Alg. 5.1) is an instance of the Bi-CG algorithm from article [3]. The $n \times s$ matrix $\tilde{\mathbf{R}}_k$ is chosen in a way such that $\text{Span}(\tilde{\mathbf{R}}_k) + \mathcal{K}_k(\mathbf{A}^*, \tilde{\mathbf{R}}_0)$ equals $\mathcal{K}_{k+1}(\mathbf{A}^*, \tilde{\mathbf{R}}_0)$.

At first we choose $\tilde{\mathbf{R}}_k$ as $\tilde{\mathbf{R}}_k = \mathbf{A}^* \tilde{\mathbf{R}}_{k-1}$ but this choice leads to instable results. To correct this problem, the matrix $\tilde{\mathbf{R}}_k$ is chosen as $\tilde{\mathbf{R}}_k = \mathbf{A}^* \tilde{\mathbf{R}}_{k-1} - \tilde{\mathbf{R}}_{k-1} \mu$, with $\mu = (\tilde{\mathbf{R}}_{k-1}^* \tilde{\mathbf{R}}_{k-1}) / (\tilde{\mathbf{R}}_{k-1}^* \mathbf{A}^* \tilde{\mathbf{R}}_{k-1})$.

5.1 Numerical results for the Bi-CG algorithm

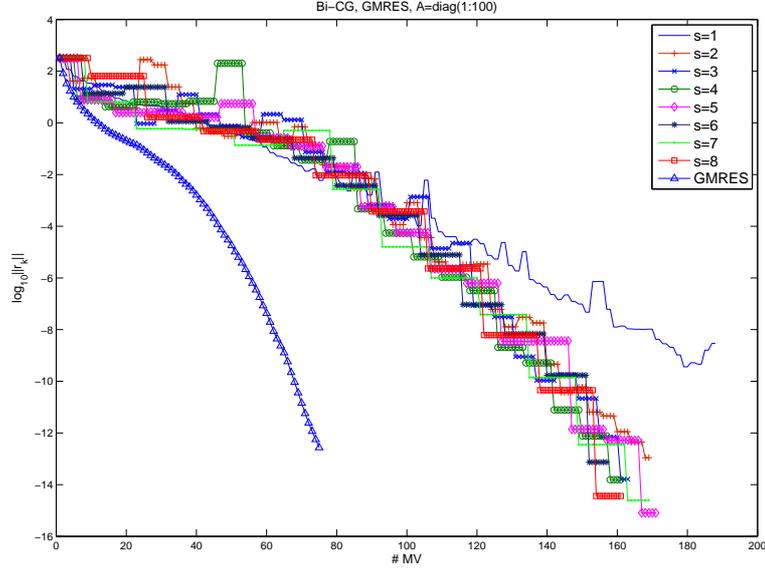


Figure 13: GMRES and Bi-CG for different s , where $\mathbf{A} = \text{diag}(1 : n)$ and $n = 100$.

In figure 13 we see the convergence plot of the GMRES with the Bi-CG for different s . We work with the diagonal matrix \mathbf{A} , where $\mathbf{A} = \text{diag}(1 : n)$ and $n = 100$. The $n \times s$ matrix $\tilde{\mathbf{R}}_0$ is orthogonal and randomly chosen. Here the MV's are counted for the multiplication with \mathbf{A} and \mathbf{A}^* .

For $s = 1$ the Bi-CG algorithm uses 178 MV's and for $s = 2$ there are 140 MV's used at a residual norm of 10^{-9} . Comparing the convergence of $s = 1$ and $s = 2$, we have here a reduction of 38 MV's. The convergence graph of $s = 3$ shows that at a residual error of 10^{-9} 131 MV's are needed, where $s = 4$ uses 132 MV's, $s = 5$ uses 146 MV's. For $s = 8$ there are 137 MV's needed at a residual error of 10^{-9} . The reduction of the MV's for the Bi-CG algorithm is very little when $s > 2$. For $s = 8$ the Bi-CG ends with 161 MV's. It is visible that the convergence of the GMRES is a lot faster than the convergence of the Bi-CG. For the GMRES 65 MV's are needed at a residual error of 10^{-9} .

The convergence of the Bi-CG for the different s does not approach the convergence of the GMRES.

In figure 14 the Bi-CG is plotted for the dimension of the Krylov space ($ks + 1$), along with the IDR(s) algorithm where \mathbf{U} is initiated by the Arnoldi iteration and the GMRES for subspace $s = 1$. The diagonal matrix $\mathbf{A} = \text{diag}(1 : n)$ is used, for $n = 100$. Again the $n \times s$ orthogonal matrix $\tilde{\mathbf{R}}_0$ is randomly chosen.

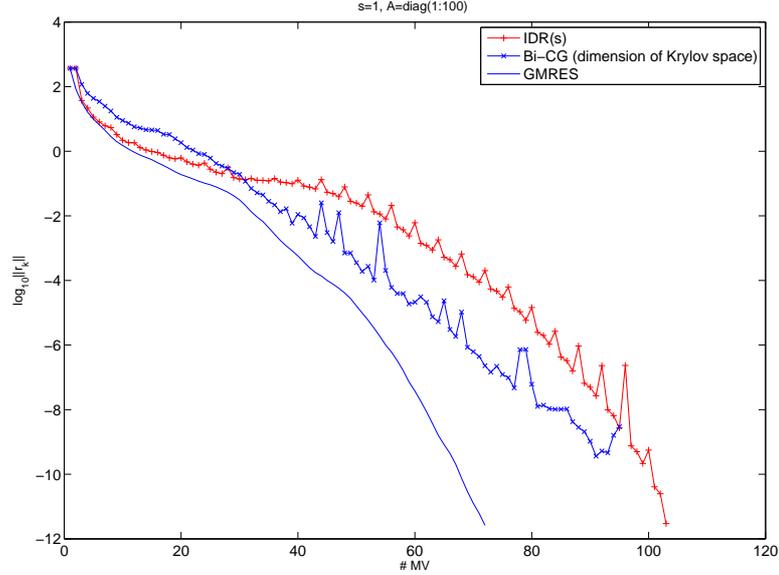


Figure 14: IDR(s) for $\mathbf{U} = \text{Arnoldi}$, GMRES and Bi-CG (dimension of the Krylov space is plotted) for $\mathbf{A} = \text{diag}(1 : n)$, $n = 100$ and $s = 1$.

The graph of the Bi-CG where the dimension of the Krylov space is plotted runs close to the convergence graph of the IDR(s) compared with the graph of the Bi-CG where the MV's are plotted for $s = 1$.

We see that the convergence graph of the Bi-CG algorithm, where the Krylov space dimension $(ks + 1)$ is plotted, runs close to the GMRES. This is made possible by our choice of matrix \mathbf{A} and the way $\tilde{\mathbf{R}}_0$ is chosen. The graph of the Bi-CG algorithm could look different when we worked with a different matrix. Because $\mathbf{A}^* = \mathbf{A}$ and $\tilde{\mathbf{R}}_0 = \mathbf{r}_0$, the Bi-CG algorithm behaves like the CG algorithm. The CG algorithm produces a graph that is close to the graph of the GMRES algorithm. GMRES and CG search the same Krylov subspace $\mathcal{K}_k(\mathbf{A}, \mathbf{r}_0)$. GMRES finds the approximation with the smallest residual norm ($\|\mathbf{b} - \mathbf{A}\mathbf{x}\|$), where CG finds the approximation with the smallest \mathbf{A}^{-1} residual norm ($\|\mathbf{b} - \mathbf{A}\mathbf{x}\|_{\mathbf{A}^{-1}}$). And because of the relation of the Bi-CG and the CG algorithm we can explain why the graphs of the Bi-CG algorithm and the GMRES are running close to each other.

The Bi-CG variant finds an approximation with a residual \mathbf{r}_k in the Krylov space $\mathcal{K}_{ks+1}(\mathbf{A}, \mathbf{r}_0)$. So the residuals $\mathbf{r}_k \in \mathcal{K}_{ks+1}(\mathbf{A}, \mathbf{r}_0)$ for the Bi-CG method, where $\mathbf{r}_k \perp \mathcal{K}_k(\mathbf{A}^*, \tilde{\mathbf{R}}_0)$.

When \mathbf{r}_k becomes \mathbf{r}_{k+1} in Bi-CG, the cost will be $2s$ MV's. \mathbf{r}_{k+1} will be in the Krylov space $\mathcal{K}_{(k+1)s+1}(\mathbf{A}, \mathbf{r}_0)$. The $2s$ MV's are needed for the multiplication with \mathbf{A}^* and during the calculation of $\tilde{\mathbf{R}}_{k=1} = \tilde{\mathbf{R}}_k = (\mathbf{A}^* \tilde{\mathbf{R}}_k)$.

6 Bi-CGSTAB, the IDR(s) variant

In this chapter we describe the Bi-CGSTAB variant of the IDR method for $s > 1$.

Like the Bi-CG we select a polynomial p_k of exact degree k and $\mathbf{P}_k \equiv p_k(\mathbf{A})$. In the Bi-CGSTAB the residual vector $\mathbf{r}_k = \mathbf{r}_k^{Bi-CG}$ and $\tilde{\mathbf{R}}_k = \tilde{p}_k(\mathbf{A}^*)\tilde{\mathbf{R}}_0$. Some of the notations of the vectors and matrices in Bi-CG are being replaced to obtain the Bi-CGSTAB algorithm. In article [3, Sect. 7] the exact details are given. Below in Alg. 6.1 we present the Bi-CGSTAB algorithm. Compared with the Bi-CGSTAB algorithm in article [3, Alg. 7.4], our algorithm has some changes. The changes are made because we were having problems with the stability of the algorithm. For $s > 2$ the results of the algorithm were not stable, because $\mathbf{R}_0^*\mathbf{S}$ was ill-conditioned.

In order to solve this problem we put the next part in the algorithm:

```

for  $j = 1, \dots, s$ 
   $J = [1, \dots, j - 1]$ 
   $\vec{\beta} = \sigma^{-1}(\tilde{\mathbf{R}}_0^*\mathbf{s})$ 
   $\mathbf{u} = \mathbf{v} - \mathbf{U}\vec{\beta}, \mathbf{v} = \mathbf{s} - \mathbf{S}\vec{\beta}$ 
   $h = \mathbf{S}'_k(:, J)^*\mathbf{v}, \mathbf{v} = \mathbf{v} - \mathbf{S}'_k(:, J)h, \mathbf{v} = \mathbf{v}/\rho$ 
   $\mathbf{u} = \mathbf{u} - \mathbf{U}'_k(:, J)h, \mathbf{u} = \mathbf{u}/\rho$ 
   $\mathbf{S}'(:, j) = \mathbf{v}, \mathbf{U}'(:, j) = \mathbf{u}$ 
   $\mathbf{s} = \mathbf{A}\mathbf{v}, \mathbf{SS}(:, j) = \mathbf{s}$ 
end for
 $\mathbf{U} = \mathbf{U} - \omega\mathbf{S}', \mathbf{S} = \mathbf{S}' - \omega\mathbf{SS}$ 

```

We introduced orthogonalization in the Bi-CGSTAB algorithm. First the vector \mathbf{v} is created by projecting the vector \mathbf{r} . Then the vector \mathbf{v} is multiplied with the matrix \mathbf{A} , where we get the vector \mathbf{s} , so that $\mathbf{s} = \mathbf{A}\mathbf{v}$. Now the vector \mathbf{s} is projected to obtain vector \mathbf{v} , followed by a multiplication with the matrix \mathbf{A} . The new vector \mathbf{s} is created and the process of the projection and the multiplication with \mathbf{A} continues until s steps are reached.

By the orthogonalization method the convergence of the Bi-CGSTAB algorithm becomes stable.

The same method is used in the first *for*-loop of Alg. 6.1. We start with the vector \mathbf{v} that is a projection of \mathbf{u} . Then the vector \mathbf{v} is multiplied with the matrix \mathbf{A} , where $\mathbf{v} = \mathbf{A}\mathbf{v}$. The vector \mathbf{v} will repeatedly be projected and multiplied until s steps are reached.

We will show two examples where it is visible that the orthogonalization introduced in the Bi-CGSTAB algorithm leads to stable convergence. The convergence of Bi-CGSTAB with orthogonalization (Alg. 6.1) will be compared to the convergence of Bi-CGSTAB where no orthogonalization is used [3, Alg. 7.4].

In figure 15 we use the TFQMR001 matrix, where $\tilde{\mathbf{R}}_0$ is an $n \times s$ randomly chosen and orthogonalized matrix. Here we see that the convergence of the Bi-CGSTAB variant with orthogonalization

for $s = 8$ runs close to the GMRES and converges fast. The Bi-CGSTAB algorithm without orthogonalization converges very slowly. For a residual norm about 10^{-12} the GMRES needs 148 MV's, the Bi-CGSTAB with orthogonalization needs 315 MV's and the Bi-CGSTAB without orthogonalization uses 1568 MV's.

This example shows that the orthogonalization used for the Bi-CGSTAB leads to faster convergence.

```

Select an  $\mathbf{x}$ 
Select an  $\tilde{\mathbf{R}}_0$ 
Compute  $\mathbf{r} = \mathbf{b} - \mathbf{A}\mathbf{x}$ 
Set  $\mathbf{U}_k = 0, \mathbf{S}_k = 0, \mathbf{U}'_k = 0, \mathbf{S}'_k = 0, \mathbf{SS}_k = 0$ 
for  $j = 1, \dots, s$ 
   $J = [1, \dots, j - 1]$ 
   $\mathbf{u} = \mathbf{v}$ , Compute  $\mathbf{v} = \mathbf{A}\mathbf{v}$ 
   $h = \mathbf{S}_k(:, J)^* \mathbf{v}$ ,  $\mathbf{v} = \mathbf{v} - \mathbf{S}_k(:, J)h$ ,  $\rho = \|\mathbf{v}\|$ ,  $\mathbf{v} = \mathbf{v}/\rho$ 
   $\mathbf{u} = \mathbf{u} - \mathbf{U}(:, J)h$ ,  $\mathbf{u} = \mathbf{u}/\rho$ 
   $\mathbf{S}(:, j) = \mathbf{v}$ ,  $\mathbf{U}(:, j) = \mathbf{u}$ 
end
repeat
   $\sigma = \tilde{\mathbf{R}}_0^* \mathbf{S}$ ,  $\vec{\alpha} = \sigma^{-1}(\tilde{\mathbf{R}}_0^* \mathbf{r})$ 
   $\mathbf{x} = \mathbf{x} + \mathbf{U}\vec{\alpha}$ ,  $\mathbf{v} = \mathbf{r} + \mathbf{S}\vec{\alpha}$ 
   $\mathbf{s} = \mathbf{A}\mathbf{v}$ 
  Select an  $\omega$ 
   $\mathbf{x} = \mathbf{x} - \omega\mathbf{v}$ ,  $\mathbf{r} = \mathbf{v} - \omega\mathbf{s}$ 
  for  $j = 1, \dots, s$ 
     $J = [1, \dots, j - 1]$ 
     $\vec{\beta} = \sigma^{-1}(\tilde{\mathbf{R}}_0^* \mathbf{s})$ 
     $\mathbf{u} = \mathbf{v} - \mathbf{U}\vec{\beta}$ ,  $\mathbf{v} = \mathbf{s} - \mathbf{S}\vec{\beta}$ 
     $h = \mathbf{S}'_k(:, J)^* \mathbf{v}$ ,  $\mathbf{v} = \mathbf{v} - \mathbf{S}'_k(:, J)h$ ,  $\mathbf{v} = \mathbf{v}/\rho$ 
     $\mathbf{u} = \mathbf{u} - \mathbf{U}'_k(:, J)h$ ,  $\mathbf{u} = \mathbf{u}/\rho$ 
     $\mathbf{S}'(:, j) = \mathbf{v}$ ,  $\mathbf{U}'(:, j) = \mathbf{u}$ 
     $\mathbf{s} = \mathbf{A}\mathbf{v}$ ,  $\mathbf{SS}(:, j) = \mathbf{s}$ 
  end for
   $\mathbf{U} = \mathbf{U} - \omega\mathbf{S}'$ ,  $\mathbf{S} = \mathbf{S}' - \omega\mathbf{SS}$ 
end repeat

```

Alg. 6.1: Bi-CGSTAB, the IDR(s) variant.

For $\mathbf{S}(:, j)$, $\mathbf{U}(:, j)$, $\mathbf{S}'(:, j)$, $\mathbf{U}'(:, j)$, $\mathbf{SS}(:, j)$ Matlab notation is used.

Figure 16 uses the SAAD3 matrix to illustrate the convergence of the Bi-CGSTAB with and without orthogonalization and the GMRES.

The matrix SAAD3 [6] is a result of a 66×66 finite volume discretization of equation (2), where $\gamma = 1000$ and $\beta = 10$. The matrix SAAD3 is of size $n = 4096$.

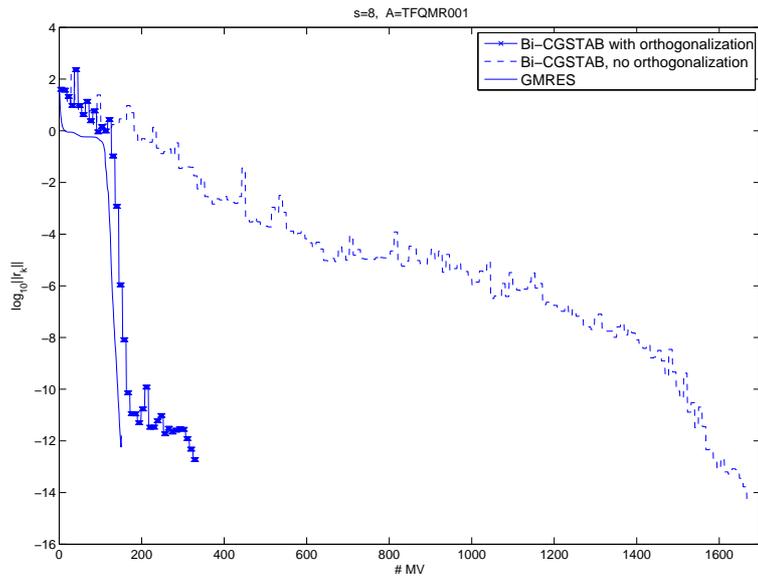


Figure 15: *Orthogonalization in the Bi-CGSTAB leads to a faster convergence for the matrix TFQMR001.*

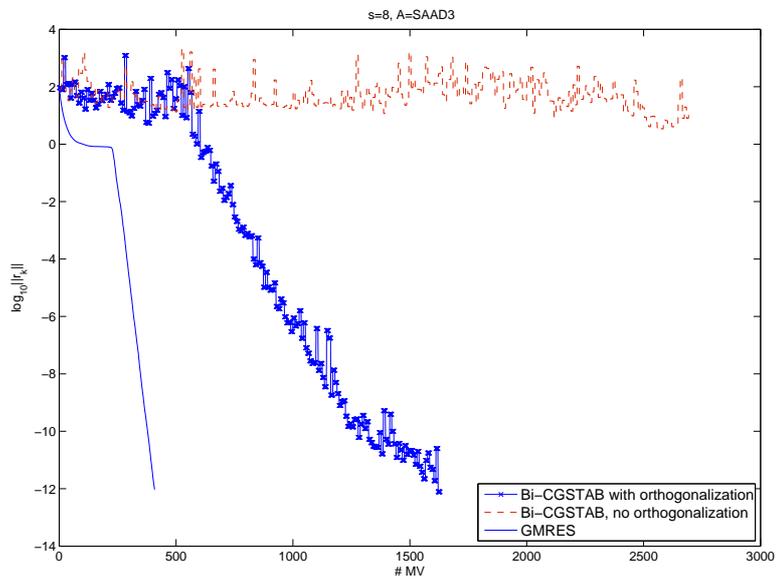


Figure 16: *Orthogonalization in the Bi-CGSTAB leads to convergence for the matrix SAAD3. The Bi-CGSTAB with no orthogonalization does not converge at all.*

For figure 16 the $n \times s$ matrix $\tilde{\mathbf{R}}_0$ is randomly chosen and orthogonalized, where $s = 8$. The figure shows that the Bi-CGSTAB with orthogonalization slowly converges compared with the convergence of the GMRES. At a residual norm about 10^{-12} the GMRES uses 407 MV's, where the Bi-CGSTAB with orthogonalization uses 1621 MV's. We see that the Bi-CGSTAB where no orthogonalization is used does not converge at all.

From the experiments we can conclude that orthogonalization in the Bi-CGSTAB algorithm leads to optimal convergence. For the TFQMR001 matrix we found that the orthogonalization leads to a faster convergence. The example where SAAD3 is used, orthogonalization in the Bi-CGSTAB leads to convergence. Using no orthogonalization for the Bi-CGSTAB gives poor convergence when we experiment with SAAD3 .

We notice that the convergence of the Bi-CGSTAB depends on the choice of matrix \mathbf{A} , but orthogonalization in the Bi-CGSTAB leads always to improvement of the convergence.

IDR(s) and Bi-CGSTAB

The IDR(s) (Alg 3.1) and the IDR(s) variant of the Bi-CGSTAB (Alg. 6.1) are equivalent. If these two algorithms produce the same $\mathbf{v}_k, \mathbf{r}_k$ in exact arithmetic then they will produce also the same $\mathbf{r}_{k+1}, \mathbf{v}_{k+s+1}$ and \mathbf{r}_{k+s+1} . And with the same initialization of the IDR(s) and the Bi-CGSTAB they will produce the same residuals and approximations every $s + 1$ step. The difference between the Bi-CGSTAB and the IDR(s) method is that the Bi-CGSTAB keeps the matrices \mathbf{S}_k and \mathbf{U}_k in a fixed loop and the update takes place outside the loop. In the IDR(s) algorithm these matrices are updated in each step.

6.1 Numerical results for the IDR(s) variant of the Bi-CGSTAB and the IDR(s)

In Section 6.1.1 we first plot the convergence of the IDR(s) variant of the Bi-CGSTAB for different subspaces s . In Section 6.1.2 we compare the convergence of the IDR(s) with the IDR(s) variant of the Bi-CGSTAB.

6.1.1 The IDR(s) variant of the Bi-CGSTAB

In figure 17 the convergence graphs of the Bi-CGSTAB (Alg. 6.1) for different s are plotted. We work with the diagonal matrix $\mathbf{A} = \text{diag}(1 : n)$, where $n = 100$. The $n \times s$ matrix $\tilde{\mathbf{R}}_0$ is randomly chosen and orthogonalized. With the growth of the subspace s we observe that the convergence of the Bi-CGSTAB approaches the GMRES, what implicates a good convergence. At a residual norm of 10^{-12} , the GMRES stops at 73 MV's and the Bi-CGSTAB for $s = 8$ stops with 82 MV's and $s = 1$ stops with 108 MV's. The convergence of $s = 8$ is the closest to GMRES.

Comparing the Bi-CGSTAB with the Bi-CG algorithm for $s = 8$, where the Bi-CG algorithm uses 154 MV's at a residual norm of 10^{-12} in figure 13, we can conclude that the convergence of the Bi-CGSTAB is faster.

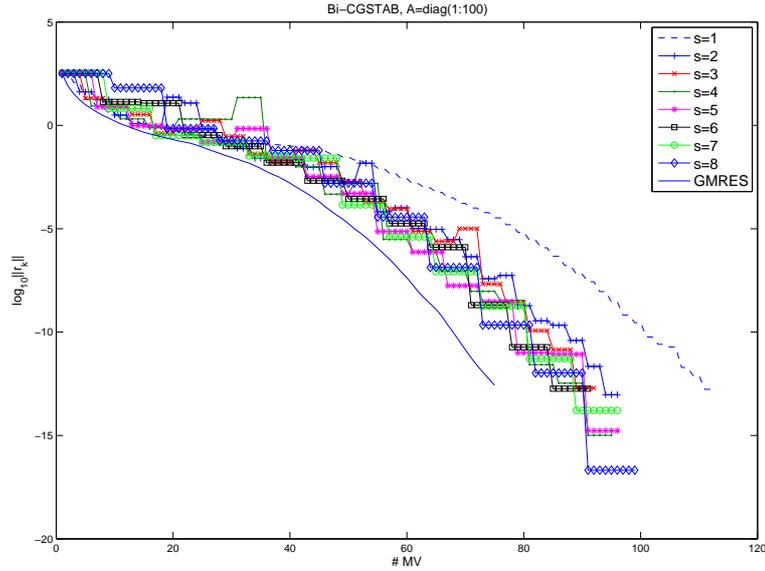


Figure 17: GMRES and the Bi-CGSTAB (Alg. 6.1) for different s . $\mathbf{A} = \text{diag}(1 : n)$ and $n = 100$

6.1.2 The IDR(s) variant of the Bi-CGSTAB and the IDR(s)

As we now know, the IDR(s) variant of the Bi-CGSTAB (Alg. 6.1) and the IDR(s) (Alg. 3.1) produce the same residuals every $s + 1$ step, because the two algorithms produce the same \mathbf{v} and \mathbf{r} vectors.

We will now show an example, see figure 18, where the IDR(s) and the IDR(s) variant of the Bi-CGSTAB produce the same residuals every $s + 1$ step. For the example we used the matrix TFQMR001. The $n \times s$ matrix $\tilde{\mathbf{R}}_0$ is randomly chosen and orthogonalized, where $n = 3969$ and $s = 8$. The matrix \mathbf{U} for IDR(s) algorithm is initiated by the Arnoldi iteration.

The graphs of the Bi-CGSTAB and the IDR(s) in figure 18 do not always produce the same residuals at every $s + 1$ step, but the residuals are very close to each other at every $s + 1$ step. The reason why these two algorithms do not always produce an exact same residual every $s + 1$ step could be caused by numerical errors.

It can be noticed that for the matrix TFQMR001 the Bi-CGSTAB and the IDR(s) have similar convergence.

In figure 19 we plot the GMRES, the Bi-CGSTAB and the IDR(s) where \mathbf{U} is initiated as $\mathbf{U} = \mathbf{r}$ using the TFQMR001 matrix. The $n \times s$ matrix $\tilde{\mathbf{R}}_0$ is randomly chosen and orthogonalized, where $n = 3969$ and $s = 8$. Here the Bi-CGSTAB and the IDR(s) do not produce the same residuals at every $s + 1$ step.

Experimenting with other matrices like the SHERMAN3 [6], we found that the Bi-CGSTAB and the IDR(s) also produce the same $s + 1$ residuals every $s + 1$ step, when \mathbf{U} is initiated by the Arnoldi iteration. When \mathbf{U} is initiated as $\mathbf{U} = \mathbf{r}$, for the IDR(s) algorithm we again found that the

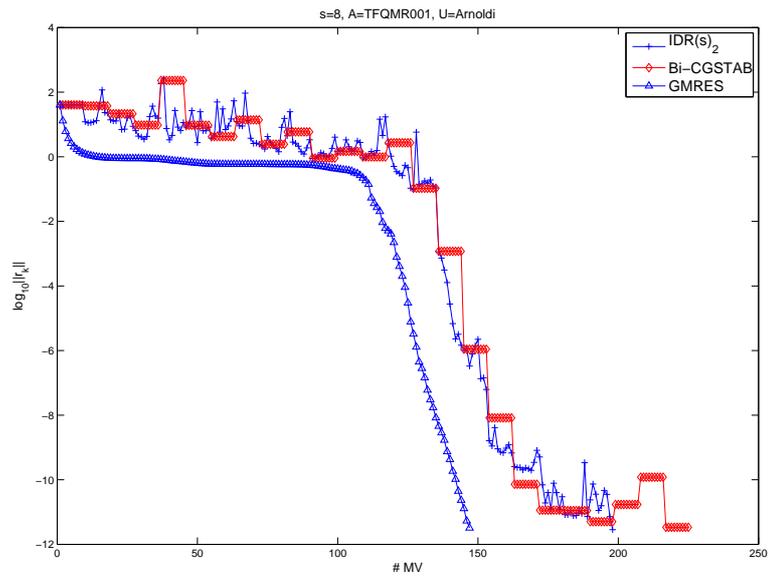


Figure 18: The Bi-CGSTAB and the IDR(s) produce the same residuals every $s + 1$ step when \mathbf{U} is initiated by Arnoldi for $\mathbf{A} = \text{TFQMR001}$ and $s = 8$ because of the \mathbf{v} and \mathbf{r} vectors. The convergence of the two algorithms approach the GMRES.

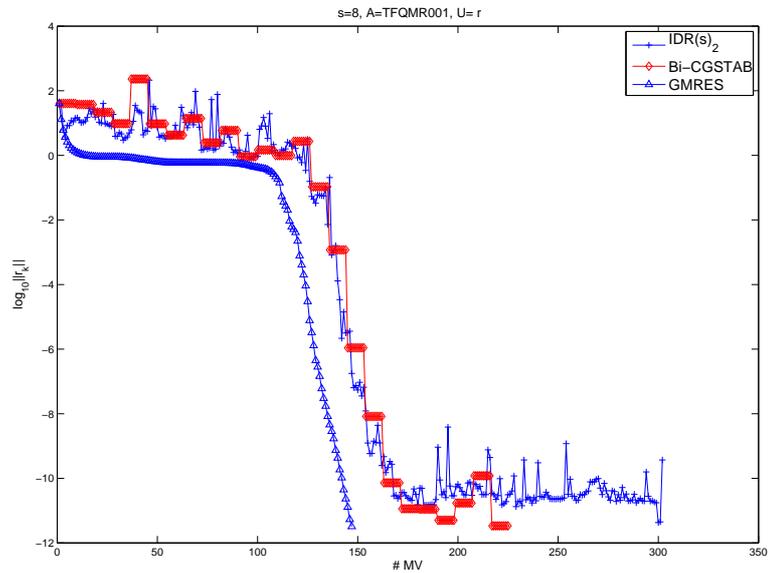


Figure 19: The Bi-CGSTAB and the IDR(s) do not produce the same residuals every $s + 1$ step when \mathbf{U} is initiated as $\mathbf{U} = \mathbf{r}$ for $\mathbf{A} = \text{TFQMR001}$ and $s = 8$. But the convergence of the two algorithms run close to each other and approach the GMRES.

Bi-CGSTAB and the IDR(s) do not share the same residuals at every $s + 1$ step.

Figure 20 shows a plot of the convergence of the GMRES, IDR(s), Bi-CGSTAB and the Bi-CG algorithm. The matrix $\mathbf{A} = \text{SAAD3}$, \mathbf{U} is initiated by the Arnoldi and $s = 8$. The $n \times s$ matrix $\tilde{\mathbf{R}}_0$ is randomly chosen and orthogonalized. The figure shows that the IDR(s) and the Bi-CG algorithm do not converge at all, but the Bi-CGSTAB surprisingly converges well.

If we compare the GMRES with the Bi-CGSTAB, the GMRES uses 407 MV's at a residual norm about 10^{-12} . The Bi-CGSTAB uses 1621 MV's at a residual norm 10^{-12} . The IDR(s) and the Bi-CG stays constant at a residual norm 10^2 .

The reason why the Bi-CGSTAB converges better than the IDR(s) algorithm is because of the orthogonalization we introduced in the Bi-CGSTAB algorithm.

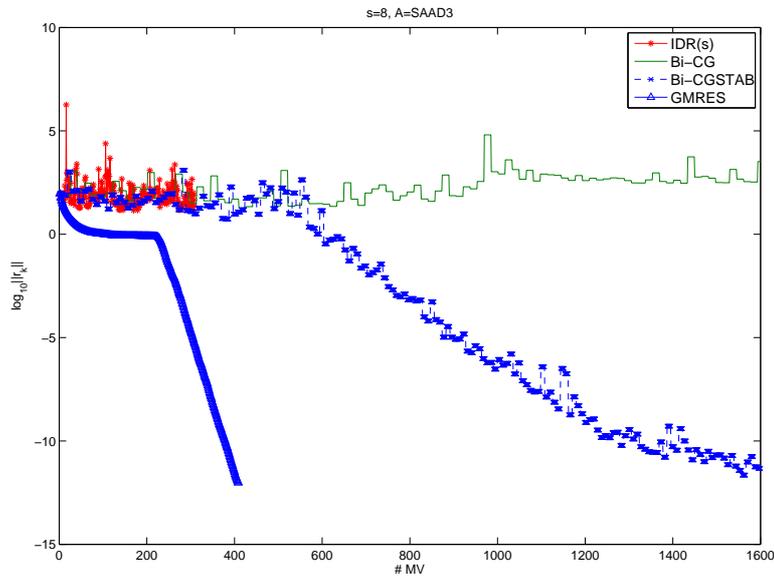


Figure 20: *The Bi-CGSTAB converges because of orthogonalization. The IDR(s) and the Bi-CG do not converge well. We used the matrix $\mathbf{A} = \text{SAAD3}$ and $s = 8$.*

Experimenting with the matrix MEIER01, results showed that the Bi-CGSTAB also converges better than the IDR(s) and the Bi-CG algorithm. The convergence of the IDR(s) terminates with 309 MV's where the residual norm is about 10^{-4} and the convergence of the Bi-CG lies around a residual norm 10^{-4} . At a residual norm 10^{-11} the GMRES uses 193 MV's, where the Bi-CGSTAB uses 1918 MV's. These results show that the Bi-CGSTAB has a better convergence than the IDR(s). The orthogonalization we introduced in the Bi-CGSTAB leads to better convergence.

From the results we can conclude that the convergence of the Bi-CGSTAB algorithm is similar or better than the convergence of the IDR(s), when using orthogonalization. The Bi-CGSTAB and the IDR(s) produced similar convergence for the matrices TFQMR001 and the SHERMAN4. For the matrices SAAD3 and MEIER01 the results showed that the Bi-CGSTAB had a better convergence

than the IDR(s). We notice that the convergence is always dependent on the choice of the matrix \mathbf{A} .

7 Conclusion

In this thesis we did a lot of numerical experiments where the IDR(s) algorithm is compared with a variant of the Bi-CGSTAB (Alg. 4.1), the IDR (Alg. 4.2), original IDR (Alg. 4.3), the Bi-CG (Alg. 5.1) and the IDR(s) variant of the Bi-CGSTAB (Alg. 6.1).

We examine two variants of the IDR(s) algorithm, i.e. the IDR(s)₁ (Alg. 3.1) and the IDR(s)₂ (Alg. 3.2) algorithm. These two variants of the IDR(s) are a result of the IDR(s) algorithm from [3] that had an error. The IDR(s)₁ and the IDR(s)₂ are tested and compared for different starting strategies, which of the two produce better convergence.

We compare the two IDR(s) variants for an $n \times s$ matrix \mathbf{U} that we initiate in four different ways: $\mathbf{U} = \mathbf{r}$, \mathbf{U} is initiated by the Arnoldi, $\mathbf{U} = \tilde{\mathbf{R}}_0, \mathbf{S} = \mathbf{AU}$ and $\mathbf{U} = \tilde{\mathbf{R}}_0, \mathbf{S} = \mathbf{AU}, \tilde{\mathbf{R}}_0 = \mathbf{S}$. The $n \times s$ matrix $\tilde{\mathbf{R}}_0$ is randomly chosen and orthogonalized.

From experiments with the IDR(s)₁ and the IDR(s)₂ algorithm, where we used the diagonal matrix $\mathbf{A} = \text{diag}(1 : 100)$, we get that the two IDR(s) variants produce the same residuals every $s + 1$ step when the matrix \mathbf{U} is initiated in the following three ways: \mathbf{U} is initiated by the Arnoldi, $\mathbf{U} = \tilde{\mathbf{R}}_0, \mathbf{S} = \mathbf{AU}$ and $\mathbf{U} = \tilde{\mathbf{R}}_0, \mathbf{S} = \mathbf{AU}, \tilde{\mathbf{R}}_0 = \mathbf{S}$.

The two IDR(s) variants produce the same residuals every $s + 1$ step because the same matrices $\tilde{\mathbf{R}}_0$ and \mathbf{U} are used in both variants and for both IDR(s) variants the vector \mathbf{r} is computed and used to generate convergence.

When the matrix \mathbf{U} is initiated as $\mathbf{U} = \mathbf{r}$, the two IDR(s) variants do not produce the same residuals at every $s + 1$ step. This is because for the initiation $\mathbf{U} = \mathbf{r}$ the Krylov space for \mathbf{U} is spanned by the vectors $[\hat{\mathbf{r}}, \mathbf{A}\hat{\mathbf{r}}, \dots, \mathbf{A}^s\hat{\mathbf{r}}]$, where $\hat{\mathbf{r}} = \mathbf{r}\vec{\gamma} + \omega(\mathbf{r} - \mathbf{A}\mathbf{r}\vec{\gamma})$ when $s > 1$.

Experiments with different matrices \mathbf{A} , like the MEIER01 and TFQMR001 showed that we get the best convergence when \mathbf{U} is initiated as $\mathbf{U} = \mathbf{r}$ or \mathbf{U} is initiated by the Arnoldi. The initiations $\mathbf{U} = \tilde{\mathbf{R}}_0, \mathbf{S} = \mathbf{AU}$ and $\mathbf{U} = \tilde{\mathbf{R}}_0, \mathbf{S} = \mathbf{AU}, \tilde{\mathbf{R}}_0 = \mathbf{S}$ produce very poor convergence.

The convergence produced by initial matrix $\mathbf{U} = \mathbf{r}$ and the matrix \mathbf{U} initiated by the Arnoldi is almost equal, and therefore we can not choose the best performing initial matrix \mathbf{U} .

We examine which of the two IDR(s) variants produce better convergence. Experimenting with the TFQMR001 matrix showed that the IDR(s)₂ has better convergence than the IDR(s)₁ algorithm for \mathbf{U} initiated as $\mathbf{U} = \mathbf{r}$ and \mathbf{U} initiated by Arnoldi.

The convergence of the IDR(s)₁ algorithm shows large peaks, that is developed in early residual steps. These peaks lead to errors that affect the convergence rate and cause the convergence to terminate at a early stage. Now that we found that the IDR(s)₂ produces better convergence, we use this IDR(s) variant in further experiments. When the subspace s gets bigger e.g $s = 8$, we see that the convergence of the IDR(s) algorithm approaches the GMRES.

Comparing the IDR(s) (Alg. 3.1) with the IDR (Alg. 4.2) we can show that the IDR(s) is mathe-

matically equivalent with the IDR algorithm [3] for $s = 1$. The two vectors \mathbf{v} and \mathbf{r} are computed in both the IDR and the IDR(s). The IDR(s) and the IDR produce the same residuals at every $s + 1$ step. This is because in the IDR the vectors \mathbf{v} and \mathbf{r} are used to obtain convergence and in the IDR(s) only the vector \mathbf{r} is used to obtain convergence.

The IDR (Alg. 4.2) is also equivalent with the Bi-CGSTAB (Alg. 4.1), because the vectors \mathbf{v} and \mathbf{r} also are computed in both the IDR and Bi-CGSTAB. Therefore the IDR(s) (Alg. 3.1) also equivalent with the Bi-CGSTAB for $s = 1$.

We examine the original IDR algorithm (Alg. 4.3) proposed by Sonneveld [4]. Experimenting with matrix $\mathbf{A} = \text{diag}(1 : 100)$, the original IDR showed a stable convergence that is comparable with the convergence of the IDR, IDR(s) and the Bi-CGSTAB. Experimenting with other matrices like the TFQMR001 we detect that the convergence of the original IDR has large peaks that lead to loss in convergence rate. There is also no big loss in accuracy for the original IDR and from these results we can conclude that the original IDR is not unstable. Beside the peaks in the convergence, there are no other instabilities in the original IDR as stated in [1].

The Bi-CG (Alg. 5.1) is formulated for an s -dimensional initial shadow residual in order to understand how an s -dimensional initial shadow residual $\tilde{\mathbf{R}}_0$ can be included in Bi-CGSTAB. Then the IDR(s) variant of the Bi-CGSTAB (Alg. 6.1) is formulated for $s > 1$.

Comparing the IDR(s) (Alg. 3.1) with the IDR(s) variant of the Bi-CGSTAB (Alg. 6.1), we found that the IDR(s) variant of the Bi-CGSTAB produce comparable or sometimes better convergence than the IDR(s). The IDR(s) variant of the Bi-CGSTAB converges better, because of the orthogonalization we introduced in the algorithm (Alg. 6.1).

Experimenting with the matrix TFQMR001 we get that orthogonalization causes the Bi-CGSTAB to converge faster where the convergence approaches the GMRES. Experiments with matrix SAAD3 showed that orthogonalization in the Bi-CGSTAB leads to convergence, where no orthogonalization in the Bi-CGSTAB resulted in no convergence at all.

For matrices TFQMR001 and SHERMAN4 the IDR(s) and the Bi-CGSTAB produce the same residuals every $s + 1$ step when the matrix \mathbf{U} is initiated by the Arnoldi. For the matrix SAAD3 the IDR(s) algorithm does not converge. But we see that the Bi-CGSTAB algorithm with orthogonalization shows a convergence that can be compared with the GMRES.

From our results we have that the IDR(s) variant of the Bi-CGSTAB produces similar or better convergence than the IDR(s) algorithm, when using orthogonalization. But still we have to keep in mind that the convergence is dependent on the choice of the matrix \mathbf{A} .

8 Acknowledgement

Special thanks goes to Gerard Sleijpen who assisted me during my research period.

9 References

- [1] H.A. van der Vorst, Bi-CGSTAB: A fast and smoothly converging variant of Bi-C for the solution of nonsymmetric linear systems, *SIAM J. Sci. Stat. Comput.*, Vol 13(2), pp. 631-644, 1992.
- [2] Peter Sonneveld and Martin B. van Gijzen, IDR(s): a family of simple and fast algorithms for solving large nonsymmetric linear systems of linear equations. Delft University of Technology, (march, 2007).
- [3] Gerard L.G. Sleijpen, Peter Sonneveld, and Martin B. van Gijzen, Bi-CGSTAB as an induced dimension reduction method. Preprint 1369, Dep. Math., University Utrecht (April, 2008).
- [4] Piet Wesseling and Peter Sonneveld, Numerical experiments with a multiple grid- and a preconditioned lanczostype method, *Lecture Notes in Mathematics*, vol. 771, pp. 543-562, Springer Verlag, Berlin, Heidelberg, New York, 1980.
- [5] Gerard L.G. Sleijpen and Henk A. van der Vorst, Mantaining convergence properties of Bi-CGSTAB methods in finite precision arithmetic, *Numerical Algorithms* 10 (1995), pp.203-223.
- [6] <http://www.math.uu.nl/people/sleijpen/>
- [7] Gerard L.G. Sleijpen and Henk A. van der Vorst, Reliable updated residuals in hybrid Bi-CG methods. Preprint nr. 886, Dep. Math., University Utrecht (November, 1994).

10 Appendix

The algorithms are programmed in Matlab. The Matlab files are given here.

```
1 function[x,hist]=idrs2(A,b,x0,R0,s,tol,kmax)
2
3 % Alg 3.1
4 % IDR(s)_2
5
6 x=x0;
7 r=b-(A*x);
8 rho=norm(r); hist=rho;
9 tol=tol*norm(b);
10
11 %Initiation 1: Computing U by the Arnoldi iteration
12 n=size(A,1);
13 U=zeros(n,s);
14
15 if 1
16 H=zeros(s+1,s);
17 U(:,1)=r/norm(r);
18 for m=1:s-1
19     v1=A*U(:,m);
20     hist=[hist,rho];
21     for j=1:m
22         H(j,m)=U(:,j)'\*v1;
23         v1=v1-U(:,j)*H(j,m);
24     end
25     H(m+1,m)=norm(v1);
26     U(:,m+1)=v1/H(m+1,m);
27 end
28 end
29
30 %Initiation 2: Set U=r
31 %U=r;
32
33 %Initiation 3: Set U=R0, S=A*U
34 %Initiation 4: Set U=R0, S=A*U, R0=S
35 %U=R0;
36 S=A*U;
37 hist=[hist,rho];
38 %R0=S;
39
40 i=s+2;
41 j=i;
42 sigma=R0'\*S
43
44 for k=1:kmax
45     if rho<tol, rho;, break, end
46
47 gamma=sigma\(R0'\*r);
48 C=cond(R0'\*S);
49 if (C> 1000)
50     C;, k;, end
```

```

51
52 v=r-S*gamma;
53 c=A*v;
54
55 if (i>s), i=1; end
56 if (j> s+1)
57     omega = (c'*v)/(c'*c);
58     j=1;
59 end
60
61 U(:,i)=(U*gamma)+(omega*v);
62 x=x+U(:,i);
63 r1=v-omega*c;
64 S(:,i)=r-r1;
65 sigma(:,i)= R0'*S(:,i);
66 r=r1;
67 rho=norm(r); hist=[hist,rho];
68 i=i+1;
69 j=j+1;
70 end

1 function[x,hist]=idrs1(A,b,x0,R0,s,tol,kmax)
2
3 % Alg 3.2
4 % IDR(s)_1
5
6 x=x0;
7 r=b-(A*x);
8 rho=norm(r); hist=rho;
9 tol=tol*norm(b);
10
11 %Initiation 1: Computing U by the Arnoldi iteration
12 n=size(A,1);
13 U=zeros(n,s);
14
15 if 1
16 H=zeros(s+1,s);
17 U(:,1)=r/norm(r);
18 for m=1:s-1
19     v1=A*U(:,m);
20     hist=[hist,rho];
21     for j=1:m
22         H(j,m)=U(:,j)'\*v1;
23         v1=v1-U(:,j)*H(j,m);
24     end
25     H(m+1,m)=norm(v1);
26     U(:,m+1)=v1/H(m+1,m);
27 end
28 end
29
30 %Initiation 2: Set U=r
31 %U=r;
32
33 %Initiation 3: Set U=R0, S=A*U
34 %Initiation 4: Set U=R0, S=A*U, R0=S

```

```

35 %U=R0;
36 S=A*U;
37 hist=[hist,rho];
38 %R0=S;
39
40 j=s+1;
41 sigma=R0'*S;
42
43 for k=1:kmax
44     if rho<tol, break, end
45
46 gamma=sigma\(R0'*r);
47 v=r-S*gamma;
48 c=A*v;
49
50 if (j> s)
51     omega = (c'*v)/(c'*c);
52     j=0;
53 end
54
55 w=(U*gamma)+(omega*v);
56 x=x+w;
57 r1=v-omega*c;
58
59 if j≠0
60 U(:,j)=w;
61 S(:,j)=r-r1;
62 sigma(:,j)= R0'*S(:,j);
63 end
64
65 r=r1;
66 rho=norm(r); hist=[hist,rho];
67 j=j+1;
68 end

1 function[x,hist]=bicgstab(A,b,x0,R0,tol,kmax)
2 % Alg 4.1
3 % Bi-CGSTAB
4
5 x=x0;
6 r0=R0;
7 r=b-A*x;
8 rho=norm(r);
9 hist=rho;
10 u=r;
11
12 for k=1:kmax
13     if rho<tol, break, end
14
15 Au=A*u;
16 alpha=(r0'*r)/(r0'*Au);
17 v=r-(Au*alpha);
18 x=x+u*alpha;
19 hist=[hist,norm(v)];
20

```

```

21 Av=A*v;
22 omega=(Av'*v)/(Av'*Av);
23 r=v-omega*Av;
24 x=x+(omega*v);
25 rho=norm(r);
26 hist=[hist,rho];
27
28 beta=(r0'*Av)/(r0'*Au);
29 Aw=Av-(Au*beta);
30 w=v-(u*beta);
31 u=w-omega*Aw;
32 end

1 function[x,hist]=idr(A,b,x0,R0,tol,kmax)
2 % Alg 4.2
3 % IDR
4
5 x=x0;
6 r0=R0;
7
8 r=b-A*x;
9 rho=norm(r);
10 hist=rho;
11 u=r;
12
13 for k=1:kmax
14     if rho<tol, break, end
15
16 Au=A*u;
17 alpha=(r0'*r)/(r0'*Au);
18 v=r-(Au*alpha);
19 x=x+u*alpha;
20 hist=[hist,norm(v)];
21
22 Av=A*v;
23
24 r1=r;
25 omega=(Av'*v)/(Av'*Av);
26 r=v-omega*Av;
27 x=x+v*omega;
28 rho=norm(r);
29 hist=[hist,rho];
30
31 s=r1-r;
32 u1=(u*alpha)+(omega*v);
33 beta=(r0'*r)/(r0'*s);
34 v1=r-(s*beta);
35 u=(u1*beta)+(omega*v1);
36 end

```

```

1 function[x,hist]=idrold(A,b,x0,R0,s,tol,kmax)
2 % Alg. 4.3
3 % The original IDR [4]
4
5
6 m=size(A,1);
7 x=x0;
8 n=1;
9 alpha=0;
10 r=(A*x)-b;
11 rho=norm(r); hist=rho;
12 tol=tol*norm(b);
13 dy=zeros(m,s);
14 dg=zeros(m,s);
15
16 for k=0:kmax
17     if rho<tol, rho; break, end
18
19     v=r-alpha*dg;
20     c=A*v;
21
22     if (n==0 || k==0)
23         omega=(c'*v)/(c'*c);
24     end
25
26     dx=(alpha*dy)+(omega*v);
27     dr=(alpha*dg)+(omega*c);
28     x=x-dx;
29     r=r-dr;
30     rho=norm(r);
31     hist=[hist,rho];
32
33     if n==1
34         dg=dr;
35         dy=dx;
36     end
37     alpha=(R0'*r)/(R0'*dg);
38     n=1-n;
39 end

```

```

1 function[x,hist]=bicg(A,b,x0,n,s,R0,tol,kmax)
2 % Alg. 5.1
3 % The Bi-CG
4
5 x=x0;
6 r=b-A*x0;
7 rho=norm(r);hist=rho;
8 tol=tol*norm(b);
9 k=-1;
10 sigma=eye(s);
11 R=R0;
12
13 U=zeros(n,s);
14 C=zeros(n,s);
15 R=zeros(n,s);

```

```

16
17 for k=1:kmax
18     if rho<tol, break, end
19
20 ss=r;
21 for j=1:s
22     u=ss;
23     ss=A*u;
24     beta=sigma\(R'*ss);
25     u=u-U*beta;
26     U1(:,j)=u;
27     ss=ss-C*beta;
28     C1(:,j)=ss;
29     hist=[hist,rho];
30 end
31 U=U1;
32 C=C1;
33
34 if k==1, R=R0; else S=A'*R;
35 mu=(R'*R)\(R'*S);
36 R=S-(R*mu);
37 end
38
39 sigma=R'*C;
40 alpha=sigma\(R'*r);
41 x=x+(U*alpha);
42 r=r-(C*alpha);
43 rho=norm(r);
44 hist=[hist,rho*ones(1,s)];
45 end

1 function[x,hist]=idrsstab(A,b,x0,n,s,R0,tol,kmax)
2 % Alg. 6.1
3 % The IDR(s) variant of the Bi-CGSTAB
4
5 x=x0;
6 r=b-A*x;
7 rho=norm(r);
8 hist=rho;
9 tol=tol*norm(b);
10
11 U1=zeros(n,0);
12 S1=zeros(n,0);
13 SS=zeros(n,0);
14 S=zeros(n,0);
15 U=zeros(n,0);
16     v=r;
17 for j=1:s
18     J=1:j-1;
19     u=v;
20     v=A*v;
21     h=S(:,J)'+v;
22     hist=[hist,rho];
23     v=v-S(:,J)*h;
24     ns=norm(v);

```

```

25     v=v/ns;
26     S(:,j)=v;
27     u=u-U(:,J)*h;
28     u=u/ns;
29     U(:,j)=u;
30 end
31
32 for k=1:kmax
33     if rho<tol, break, end
34     sigma=R0'*S;
35     alpha=sigma\'(R0'*r);
36     x=x+U*alpha;
37     v=r-S*alpha;
38     rho=norm(v);
39     hist=[hist,rho];
40     ss=A*v;
41     omega=(ss'*v)/(ss'*ss);
42     rho=norm(v);
43     r=v-omega*ss;
44     x=x+omega*v;
45
46 for j=1:s
47     beta=sigma\'(R0'*ss);
48     u=v-(U*beta);
49     v=ss-(S*beta);
50
51     J=1:j-1;
52     h=S1(:,J)'\*v;
53     v=v-S1(:,J)*h;
54     nu=norm(v);
55     v=v/nu;
56     S1(:,j)=v;
57     u=u-U1(:,J)*h;
58     u=u/nu;
59     U1(:,j)=u;
60
61     ss=A*v;
62     SS(:,j)=ss;
63     hist=[hist,rho];
64 end
65
66 U=U1-omega*S1;
67 S=S1-omega*SS;
68 end

```

```

1 % main
2
3 rand('seed',0);
4 tol=1.0e-13;
5 kmax=300;
6
7 n=100;
8 s=8;
9 [R0,t]=qr(rand(n,s),0);
10 A=(diag((1:n)));
11 xe=rand(n,1);
12 x0= zeros(n,1);
13 b=A*xe;
14
15
16 [x,hist2]=idrs2(A,b,x0,R0,s,tol,kmax);
17 [x,hist1]=idrs1(A,b,x0,R0,s,tol,kmax);
18
19
20 %GMRES
21 [x, flag, relres, iter, resvec_gmres] = gmres(A, b, [], tol, n );
22
23 plot(log10(abs(hist1)),'-');hold on
24 plot(log10(abs(hist2)),'+-');hold on
25 plot(log10(abs(resvec_gmres)),' -');
26
27
28 xlabel('# MV')
29 ylabel('log_{10}||r_k||')

```