DELFT UNIVERSITY OF TECHNOLOGY

REPORT 17-05

A HISTORY OF KRYLOV PRODUCT METHODS
- A CASE OF SERENDIPITY -

PETER SONNEVELD

ISSN 1389-6520

Reports of the Delft Institute of Applied Mathematics

Delft 2017

Copyright © 2017 by Delft Institute of Applied Mathematics, Delft, The Netherlands.

No part of the Journal may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior written permission from Delft Institute of Applied Mathematics, Delft University of Technology, The Netherlands.

A history of Krylov Product Methods
- a case of serendipity -

Peter Sonneveld *
p.sonneveld@tudelft.nl

June 2017

Abstract

In about 1976, the author was preparing a renovation of the elementary course on numerical analysis in Delft University. In relation to the problem of solving a single non-linear equation iteratively, he wondered whether the so-called 'secant method' could be generalized to systems of N non-linear equations with N unknowns.

Before starting to read everything on the subject, the author normally tries to think about it unbiased, and so he did this time, and started with (probably) re-inventing the wheel. Would he have seen the book by Ortega and Rheinboldt at that time, he wouldn't have discovered the 'new wheel' IDR, and also CGS and BiCGSTAB probably wouldn't exist today.

Serendipity means something like 'finding the unsought', and the strange history of the so-called 'Krylov Product methods' shows some examples of this phenomenon.

Keywords: Iterative methods, IDR, Krylov-subspace methods, Bi-CGSTAB, CGS, nonsymmetric linear systems.

AMS Subject Classification 65F10, 65F50

1 The beginning.

1.1 Trying to set-up a multidimensional secant method.

The classical secant method is a quasi Newton procedure for solving one non-linear equation with one unknown $f(x) = 0$, in which the derivative $f'(x_n)$ is replaced by the divided difference

$$f'(x_n) \approx f[x_{n-1}, x_n] = \frac{f(x_n) - f(x_{n-1})}{x_n - x_{n-1}}$$

*Delft Institute of Applied Mathematics, Mekelweg 4, 2628 CD Delft, The Netherlands,

The method is of order $(1+\sqrt{5})/2 \approx 1.6$, and effectively more efficient than Newton's method, since the latter requires each step an evaluation of both $f(x)$ and $f'(x)$.
In order to set up a variant for a system $\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{0}$ of N nonlinear equations with N unknowns, assume we have $N+1$ estimates $\boldsymbol{x}_{n-N}, \boldsymbol{x}_{n-N+1}, \ldots, \boldsymbol{x}_n$ of the solution. Let the matrices $\boldsymbol{X}_n$ and $\boldsymbol{F}_n$ be defined as

$$\boldsymbol{X}_n = [\boldsymbol{x}_{n-N}\ , \boldsymbol{x}_{n-N+1}\ \cdots\ \boldsymbol{x}_n], \quad \boldsymbol{F}_n = [\boldsymbol{f}(\boldsymbol{x}_{n-N})\,, \boldsymbol{f}(\boldsymbol{x}_{n-N+1})\,, \cdots, \boldsymbol{f}(\boldsymbol{x}_n)]$$

Then an N-dimensional secant method could be defined by

$$\text{Solve} \qquad \boldsymbol{F}_n \boldsymbol{c} = \boldsymbol{0} \tag{1}$$

$$\text{Calculate} \qquad \boldsymbol{x}_{n+1} = \frac{\boldsymbol{X}_n \boldsymbol{c}}{\sum_j c_j} \tag{2}$$

The new approximation $\boldsymbol{x}_{n+1}$ can be calculated if $\sum c_k \neq 0$, and this is the case if the columns of $\boldsymbol{F}_n$ are not in one $N-1$ dimensional linear manifold.
It can easily be derived that $\boldsymbol{x}_{n+1}$ is the exact solution for $\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{0}$ if this system is linear. Whether this procedure converges in the non-linear case depends not only on the accuracy of the estimates in $\boldsymbol{X}_n$, but also on the 'degree of linear independence' of at least N of the $N+1$ columns of $\boldsymbol{F}_n$. And this is a problem: at increasing n, the vectors $\boldsymbol{f}(\boldsymbol{x}_{n-j})$ become more and more 'dependent'.
So the author experimented with different kinds of strategies for choosing an old column of $\boldsymbol{X}_n$ to be replaced by $\boldsymbol{x}_{n+1}$. It turned out that a lot of columns of the matrices $\boldsymbol{X}_n$ remained unchanged during many steps. To make things easy, he decided to analyse the behaviour of a variant in which only the vectors $\boldsymbol{x}_{n-1}$ and $\boldsymbol{x}_n$ were allowed to be replaced. (Only varying $\boldsymbol{x}_n$ wouldn't be interesting, since this is a simple method of false position, which is only of first order).
So choose

$$\boldsymbol{X}_n = [\boldsymbol{x}_0, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_{N-2},\ \boldsymbol{x}_{n-1}, \boldsymbol{x}_n]$$

We call this 'quasi one-dimensional secant method'.

1.2 Quasi one-dimensional secant method.

An iteration step of this process can be written as

$$\text{Solve} \qquad \boldsymbol{F}_L \widetilde{\boldsymbol{c}} + \alpha_n \boldsymbol{f}_{n-1} + \beta_n \boldsymbol{f}_n = \boldsymbol{0} \tag{3}$$

$$\text{Calculate} \qquad \boldsymbol{x}_{n+1} = \frac{\boldsymbol{X}_L \widetilde{\boldsymbol{c}} + \alpha_n \boldsymbol{x}_{n-1} + \beta_n \boldsymbol{x}_n}{\widetilde{\boldsymbol{e}}^T \widetilde{\boldsymbol{c}} + \alpha_n + \beta_n} \tag{4}$$

where $\boldsymbol{X}_n$ and $\boldsymbol{F}_n$ have been splitted into fixed $N \times (N-1)$ submatrices $\boldsymbol{X}_L$ and $\boldsymbol{F}_L$ respectively, followed by two 'variable' columns $\boldsymbol{x}_{n-1}$ and $\boldsymbol{x}_n$ c.q. $\boldsymbol{f}_{n-1}$ and $\boldsymbol{f}_n$. The vector $\widetilde{\boldsymbol{e}}$ represents the all-ones vector in $\mathbb{R}^{N-1}$.
Some simple experiments indicated superlinear convergence indeed, but not very impressive.

In order to monitor the limiting behaviour of the process, a linear system $\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{A}\boldsymbol{x} = \boldsymbol{0}$ was chosen, with an imitated non-linearity via $\boldsymbol{F}_L = \boldsymbol{A}\boldsymbol{X}_L + \boldsymbol{E}$, with $\boldsymbol{E} \neq \boldsymbol{O}$. For the size of the system $N = 11$ was chosen. In the following tables some results are shown.

n	$\|\boldsymbol{f}_n\|$	n	$\|\boldsymbol{f}_n\|$	n	$\|\boldsymbol{f}_n\|$	n	$\|\boldsymbol{f}_n\|$
0	2.9481e+00	7	1.0877e+00	14	8.5673e-02	21	9.9293e-08
1	3.2656e+00	8	1.8072e+00	15	3.5577e-03	22	2.5199e-15
2	2.1400e+00	9	4.7870e-01	16	1.9406e-03	23	2.3340e-16
3	2.6949e+00	10	2.4484e-01	17	1.0996e-03	24	9.0617e-16
4	2.3372e+00	11	1.8856e-01	18	3.1076e-05	25	8.9141e-17
5	1.0998e+01	12	1.3929e-01	19	5.8016e-06	26	7.0092e-18
6	1.1215e+00	13	2.1758e-01	20	2.8276e-07		

Idealized secant method, N=11

If we inspect the table, we see that at entry 22, $\|\boldsymbol{f}_n\|$ drops to machine precision. If we choose a different but not too large value for N, we also observe such a drop at entry $2N$. At the time that these experiments were done by the author in 1976, this phenomenon was more difficult to recognize, because 8 digits arithmetic was used at the time. So he considers himself lucky to have observed this norm-dropping property anyway.

1.3 An analysis of the $2N$-reduction behaviour.

A careful analysis of the process (3), (4) leads to the following result. The residuals $\boldsymbol{f}_n$ are related to each other by

$$\boldsymbol{f}_{n+1} = \rho_n \boldsymbol{B}\left(\boldsymbol{f}_n + \gamma(\boldsymbol{f}_n - \boldsymbol{f}_{n-1})\right), \tag{5}$$

$$\text{where} \quad \gamma \text{ satisfies } \boldsymbol{p}^T\left(\boldsymbol{f}_n + \gamma(\boldsymbol{f}_n - \boldsymbol{f}_{n-1})\right) = 0 \tag{6}$$

Here $\boldsymbol{B}$ is a fixed matrix depending of $\boldsymbol{F}_L$ and $\boldsymbol{E}$. ρ_n are $O(1)$ scale factors. Carrying out the process (5) and (6) with random matrix $\boldsymbol{B}$, and all ρ_n chosen unity, shows a similar norm dropping at $n = 2N$. Obviously this behaviour is structural. **Why does this happen?**

Apart from $\boldsymbol{f}_0$ and $\boldsymbol{f}_1$, all vectors $\boldsymbol{f}_n$ are in the $\boldsymbol{B}$ image of $\boldsymbol{p}^\perp$. Especially $\boldsymbol{f}_2$ and $\boldsymbol{f}_3$ are in that subspace. But: a special combination is made of these vectors, that is perpendicular to $\boldsymbol{p}$. So $\boldsymbol{f}_4$ is in the $\boldsymbol{B}^2$ image of $\boldsymbol{p}^\perp$ as well, and so will all $\boldsymbol{f}_n$ with $n > 4$.
Going on like this, after $2k$ steps, we have $\boldsymbol{f}_n \in \boldsymbol{B}^k(\boldsymbol{p}^\perp)$, for all $n \geq 2k$. So in particular

$$\boldsymbol{f}_{2k} \in \bigcap_{j=1}^{k} \boldsymbol{B}^j(\boldsymbol{p}^\perp)$$

The intersection of subsets in the righthand side is probably 'shrinking', as can be shown by the following formalization.
Let $\mathcal{G}_0 = \text{span}(\{\boldsymbol{f}_0, \boldsymbol{B}\boldsymbol{f}_0, \ldots, \boldsymbol{B}^j\boldsymbol{f}_0, \ldots\})$, so $\mathcal{G}_0$ is the Krylov space corresponding to $\boldsymbol{B}$ and $\boldsymbol{f}_0$. Let $\boldsymbol{S}$ be a proper subspace of $\mathbb{R}^N$, and let the sequence of spaces $\{\mathcal{G}_k\}$ be generated by $\mathcal{G}_k = \boldsymbol{B}(\mathcal{G}_{k-1} \bigcap \boldsymbol{S})$ for $k = 1, 2, 3, \ldots$. Then it can easily be shown that $\mathcal{G}_k \subset \mathcal{G}_{k-1}$. This process continues until $\mathcal{G}_{M+1} = \mathcal{G}_M$ for some M. In that case $\mathcal{G}_n = \mathcal{G}_M$ for all $n > M$.
Now $\mathcal{G}_{M+1} = \mathcal{G}_M$ implies $\mathcal{G}_M$ is an invariant subspace for $\boldsymbol{B}$, and $\mathcal{G}_M \subset \boldsymbol{S}$. Therefore, if $\dim(\mathcal{G}_M) > 0$, $\boldsymbol{B}$ has an eigenvector in $\boldsymbol{S}$. In our experiment, $\boldsymbol{S} = \boldsymbol{p}^\perp$, so $\boldsymbol{B}$ has one eigenvector perpendicular to $\boldsymbol{p}$. Since a Krylov space contains at most one eigen direction for each distinct eigenvalue, this is a non-generic case. Hence, for 'almost every choice for $\boldsymbol{p}$ we have $\mathcal{G}_M = \{\boldsymbol{0}\}$ for some $M \leq N$.

This property is called the *IDR principle* (*I*nduced *D*imension *R*eduction)

2 IDR-based finite solvers.

The process (5) and (6), with $\rho_n = 1$ for all n, is an implementation of the IDR principle: it produces vectors $\boldsymbol{f}_n$ that are forced to be in the space $\mathcal{G}_j$ for $n \geq 2j$. It could be used as a finite solver for a linear system $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ if we could interprete $\boldsymbol{f}_n$ as the residual $\boldsymbol{r}_n = \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}_n$ for a sequence of approximations $\boldsymbol{x}_n$.
Now residual differences satisfy $\boldsymbol{r}_{n+1} - \boldsymbol{r}_n = -\boldsymbol{A}(\boldsymbol{x}_{N+1} - \boldsymbol{x}_n)$, and using this we can write the IDR-recursion as

$$\boldsymbol{r}_{n+1} - \boldsymbol{r}_n = -\boldsymbol{A}(\boldsymbol{x}_{n+1} - \boldsymbol{x}_n) = (\boldsymbol{B} - \boldsymbol{I})\boldsymbol{r}_n - \gamma_n \boldsymbol{B}\boldsymbol{A}(\boldsymbol{x}_n - \boldsymbol{x}_{n-1})$$

Now choosing $\boldsymbol{B} = \boldsymbol{I} - \boldsymbol{A}$, and observing that $\boldsymbol{A}$ and $\boldsymbol{B}$ commute consequently, we can 'divide $\boldsymbol{A}$ out':

$$\boldsymbol{x}_{n+1} - \boldsymbol{x}_n = \boldsymbol{r}_n + \gamma_n \boldsymbol{B}(\boldsymbol{x}_n - \boldsymbol{x}_{n-1})$$

An extra multiplication with $\boldsymbol{B}$ seems to be necessary, but of course we can use $\boldsymbol{B}(\boldsymbol{x}_n - \boldsymbol{x}_{n-1}) = \boldsymbol{x}_n - \boldsymbol{x}_{n-1} + \boldsymbol{r}_n - \boldsymbol{r}_{n-1}$.
Write the system $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ as $\boldsymbol{x} = \boldsymbol{B}\boldsymbol{x} + \boldsymbol{b}$, then the above primitive algorithm can be regarded as an 'accelerator' for a 'classical' iteration process $\boldsymbol{x}_{n+1} = \boldsymbol{B}\boldsymbol{x}_n + \boldsymbol{b}$. The very first operational IDR-algorithm, in 1976 was based on the Gauss-Seidel iteration procedure, thus using the splitting $\boldsymbol{A} = \boldsymbol{L} + \boldsymbol{D} + \boldsymbol{U}$, where $\boldsymbol{L}$, $\boldsymbol{U}$, and $\boldsymbol{D}$ are strictly lower triangular, strictly upper triangular and diagonal matrices respectively. We then have $\boldsymbol{B} = -(\boldsymbol{L} + \boldsymbol{D})^{-1}\boldsymbol{U}$, and solve the system $\boldsymbol{x} = \boldsymbol{B}\boldsymbol{x} + (\boldsymbol{L} + \boldsymbol{D})^{-1}\boldsymbol{b}$. Of course the matrix $\boldsymbol{B}$ is not required explicitly: solving a triangular system is enough. Since 2008 this procedure is called *Accelerated Gauss Seidel* (AGS), and has actually been a research object. The acceleration effect is shown in figure 1.
If tested on a discretized Poisson equation on 25×25 grid, (529 equations), it required about 160 steps (matvecs) to obtain a residual reduction of 10^{-14},

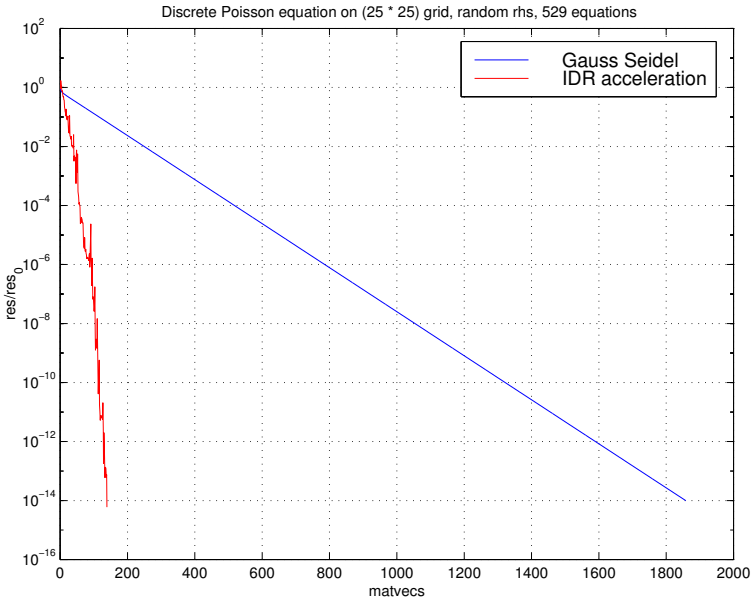

Figure 1: First IDR method (Accelerated Gauss-Seidel)

whereas plain Gauss Seidel requires about 1850 steps. But also $160 \ll 2 * 529$, and the method turns out to be much more successful than might be expected from the theoretical '$2N$ steps'.

The variant of IDR that was presented in [10], the proceedings of the Paderborn conference in 1979[1] is the first that doesn't require a suitable splitting for $\boldsymbol{A}$. It uses a variable splitting $\boldsymbol{B}_k = \boldsymbol{I} - \omega_k \boldsymbol{A}$, where the ω value could be changed each even-numbered iteration step, in a suitable way. For instance in such way that the smallest residual norm is created for that step. The algorithm has the same finite termination property, because in the analysis of the IDR-phenomenon, the matrix $\boldsymbol{B}$ may be replaced each dimension reduction step by (another) linear combination of the type $\alpha \boldsymbol{I} + \beta \boldsymbol{B}$. This method worked well in some mildly unsymmetric cases, especially if a good preconditioner was used. Later, after more elaborate tests, the method sometimes suffered from numerical instability.

3 The polynomial connection.

Like Gauss Seidel, AGS and IDR are Krylov subspace methods as well, and this suggests a convergence analysis by means of polynomials. The residuals

[1] This rather obscure paper is present on the world wide web, as `www.springerlink.com/index/r26754052u272286.pdf`

produced by the primitive IDR method are in a Krylov space based on $\boldsymbol{A}$ and $\boldsymbol{r}_0$, and therefore

$$\boldsymbol{r}_n = \Phi_n(\boldsymbol{A})\boldsymbol{r}_0$$

for all $n \geq 0$, where Φ_n denotes a polynomial of degree n, satisfying $\Phi_n(0) = 1$. Let polynomials Ω_k be defined by

$$\Omega_0(t) = 1,\ \Omega_k(t) = (1 - \omega_k t)\Omega_{k-1}(t),\ k = 1, 2, \ldots$$

Then inspection of the precise recurrence relations in the IDR-algorithm shows a more specific property

$$\boldsymbol{r}_{2k} = \Omega_k(\boldsymbol{A})\varphi_k(\boldsymbol{A})\boldsymbol{r}_0, \quad \boldsymbol{r}_{2k+1} = \Omega_k(\boldsymbol{A})\psi_{k+1}(\boldsymbol{A})\boldsymbol{r}_0$$

where φ_k and ψ_k are polynomials of degree k. Now the following property is not extremely difficult to prove:

$$\boldsymbol{p}^T\Omega_j(\boldsymbol{A})\varphi_k(\boldsymbol{A})\boldsymbol{r}_0 = (\Omega_j(\boldsymbol{A}^T)\boldsymbol{p})^T\varphi_k(\boldsymbol{A})\boldsymbol{r}_0 = 0,\ j = 0, 1, 2, \ldots, k-1 \qquad (7)$$

In the case that $\boldsymbol{A}$ is symmetric and positive definite, and $\boldsymbol{p} = \boldsymbol{r}_0$, the polynomials φ_k can be identified with (scaled) Lanczos polynomials.
There is another solver, for symmetric positive definite systems, where the residuals are generated by Lanczos polynomials, namely the Conjugate Gradients algorithm. This CG algorithm can be considered as a computational variant of the Lanczos eigenvalue process [3], but dedicated to solving linear systems instead. This is also a finite termination algorithm *that most of the time converges quickly as an iterative procedure*.
So IDR appears to be related to CG, but with the difference that the latter only applies for symmetric positive definite linear systems.
For CG, there exist a beautiful convergence analysis, based on properties of the CG-polynomials. It was the author's hope that the 'polynomial connection' between IDR and CG could be helpful for a convergence analysis of IDR.

3.1 CG for polynomials.

In the classical CG method, the residuals $\boldsymbol{r}_n$ and search directions $\boldsymbol{p}_n$ can be written as

$$\boldsymbol{r}_n = \varphi_n(\boldsymbol{A})\boldsymbol{r}_0, \quad \boldsymbol{p}_n = \psi_n(\boldsymbol{A})\boldsymbol{r}_0$$

The inner products $\boldsymbol{r}_n^T\boldsymbol{r}_n$ and $\boldsymbol{p}_n^T\boldsymbol{A}\boldsymbol{p}_n$, required for the calculation of the coefficients in the algorithm, can be expressed as *polynomial inner products:*

$$\langle \chi_1, \chi_2 \rangle = [\chi_1(\boldsymbol{A})\boldsymbol{r}_0]^T[\chi_2(\boldsymbol{A})\boldsymbol{r}_0]$$

For real symmetric matrices $\boldsymbol{A}$, this bilinear form is an innerproduct on the space of polynomials of maximal degree $d-1$, where d is the dimension of the full Krylov space.
The CG algorithm can now be interpreted as an algorithm for the construction of the set orthogonal polynomials corresponding to the $\langle .\,, .\rangle$ innerproduct:

CG algorithm for orthogonal polynomials:

$\varphi_0 = 1,\ \psi_{-1} = 0,\ \rho_{-1} = 1$
for $n = 0, 1, 2, \ldots$
$\rho_n = \langle \varphi_n, \varphi_n \rangle,\ \beta_n = \rho_n / \rho_{n-1}$
$\psi_n(t) = \varphi_n(t) + \beta_n \psi_{n-1}(t),$
$\sigma_n = \langle \psi_n, t\psi_n \rangle,\ \alpha_n = \rho_n / \sigma_n$
$\varphi_{n+1}(t) = \varphi_n(t) - \alpha_n t \psi_n(t)$
$n = n + 1$
end

An interesting feature of this polynomial algorithm is the possibility of using an alternative innerproduct definition for the polynomials. One could define

$$\langle \chi_1 \, , \chi_2 \rangle \ = \ \boldsymbol{u}^T \chi_1(\boldsymbol{A}) \chi_2(\boldsymbol{A}) \boldsymbol{v} \tag{8}$$

Although this bilinear form isn't necessarily positive (semi-)definite, the algorithm still produces polynomials that are 'orthogonal' with respect to this bilinear form. Consequently, the vectors $\boldsymbol{u}_n = \varphi_n(\boldsymbol{A}^T)\boldsymbol{u}$ and $\boldsymbol{v}_n = \varphi_n(\boldsymbol{A})\boldsymbol{v}$, form a bi-orthogonal system, just as is the case of the Lanczos algorithm for the eigenvalue problem [2] as described in [3]. The actual calculation of the vectors $\boldsymbol{u}_n$ and $\boldsymbol{v}_n$ however would require 'shadow' operations like $\boldsymbol{A}^T\boldsymbol{u}_n$ together with $\boldsymbol{A}\boldsymbol{v}_n$.

3.2 Squaring the polynomials.

In the IDR process, the Lanczos polynomials are 'hidden' in expressions like $\Phi_{2n}(t) = \Omega_n(t)\varphi_n(t)$. Furthermore, no shadow operations with $\boldsymbol{A}^T$ are required to generate the coefficients in the algorithm. This can be understood by a remarkable property of polynomial innerproducts:

$$\langle \chi_1, \chi_2 \rangle \equiv \langle 1, \chi_1 \chi_2 \rangle$$

where $\chi_1\chi_2$ denotes the ordinary product of two polynomials. The quantities ρ_n and σ_n in the polynomial algorithm can therefore be obtained in an alternative way.

$$\rho_n = \langle 1, \varphi_n^2 \rangle,\ \sigma_n = \langle 1, t\psi_n^2 \rangle = \langle t, \psi_n^2 \rangle$$

In order to do this, the polynomials φ_n^2 and ψ_n^2 must be known explicitly. An algorithm to calculate these polynomials requires one cross-product polynomial $\varphi_n\psi_{n-1}$.

[2] Similar to the original Lanczos method which was meant for general matrices, CG can be reformulated by carrying out a left 'shadow process' as well. This I called bi-CG, not knowing at the time that Fletcher already had defined that method, and named it just so.

So define

$$\Phi_n = \varphi_n^2, \quad \Theta_n = \varphi_n \psi_{n-1}, \quad \Psi_n = \psi_n^2$$

then these polynomials can be produced by simply squaring and multiplying the recursion steps of the polynomial CG-algorithm. This way we get:

$$\begin{aligned} \Psi_n(t) &= \Phi_n(t) + 2\beta_n \Theta_n(t) + \beta_n^2 \Psi_{n-1}(t) \\ \Theta_{n+1}(t) &= \Phi_n(t) + \beta_n \Theta_n(t) - \alpha_n t \Psi_n(t) \\ \Phi_{n+1}(t) &= \Phi_n(t) - 2\alpha_n t[\Phi_n(t) + \beta_n \Theta_n(t)] + \alpha_n^2 t^2 \Psi_n(t) \end{aligned}$$

In these recursion formulas, the α_n and β_n satisfy

$$\beta_n = \frac{\rho_n}{\rho_{n-1}}, \quad \alpha_n = \frac{\rho_n}{\sigma_n} \tag{9}$$

where ρ_n and σ_n can be calculated by

$$\rho_n = \langle 1, \Phi_n \rangle, \quad \sigma_n = \langle t, \Psi_n \rangle \tag{10}$$

3.3 Back to the vectors: CGS.

The standard BiCG residuals satisfy $\boldsymbol{r}_n = \varphi_n(\boldsymbol{A})\boldsymbol{r}_0$. If things are converging, the matrix polynomial $\varphi_n(\boldsymbol{A})$ may be regarded as a contracting operator, at least if it acts on $\boldsymbol{r}_0$. But then we could take advantage from applying this operator twice: $\varphi_n(\boldsymbol{A})\boldsymbol{r}_n = \Phi_n(\boldsymbol{A})\boldsymbol{r}_0$. Therefore it may be a good idea to translate the squared polynomial algorithm to vectors again.
Let the polynomial innerproduct be defined by (8), with $\boldsymbol{v} = \boldsymbol{r}_0$, and $\boldsymbol{u} = \widetilde{\boldsymbol{r}}_0$, where $\widetilde{\boldsymbol{r}}_0$ is a more or less arbitrary 'shadow-residual'. Define $\widehat{\boldsymbol{r}}_n = \Phi_n(\boldsymbol{A})\boldsymbol{r}_0$, $\widehat{\boldsymbol{p}}_n = \Psi_n(\boldsymbol{A})\boldsymbol{r}_0$, and $\widehat{\boldsymbol{q}}_n = \Theta_n(\boldsymbol{A})\boldsymbol{r}_0$. Now substitute $\boldsymbol{A}$ for t in the 'squared polynomial algorithm', and apply the obtained operators to $\boldsymbol{r}_0$, then we get

$$\begin{aligned} \widehat{\boldsymbol{p}}_n &= \widehat{\boldsymbol{r}}_n + 2\beta_n \widehat{\boldsymbol{q}}_n + \beta_n^2 \widehat{\boldsymbol{p}}_{n-1} \\ \widehat{\boldsymbol{q}}_{n+1} &= [\widehat{\boldsymbol{r}}_n + \beta_n \widehat{\boldsymbol{q}}_n] - [\alpha_n \boldsymbol{A} \widehat{\boldsymbol{p}}_n] \\ \widehat{\boldsymbol{r}}_{n+1} &= \widehat{\boldsymbol{r}}_n - \alpha_n \boldsymbol{A} \{2[\widehat{\boldsymbol{r}}_n + \beta_n \widehat{\boldsymbol{q}}_n] - [\alpha_n \boldsymbol{A} \widehat{\boldsymbol{p}}_n]\} \end{aligned}$$

in which α_n and β_n satisfy (9), where ρ_n and σ_n are calculated by

$$\rho_n = \langle 1, \Phi_n \rangle = \widetilde{\boldsymbol{r}}_0^T \widehat{\boldsymbol{r}}_n, \quad \sigma_n = \langle t, \Psi_n \rangle = \widetilde{\boldsymbol{r}}_0^T \boldsymbol{A} \widehat{\boldsymbol{p}}_n$$

The expressions in square brackets are to be stored in temporary vectors, to reduce the number of matrix vector operations to 2, the minimum.
It is easily seen that $\widehat{\boldsymbol{r}}_{n+1} - \widehat{\boldsymbol{r}}_n$ is 'divisible by $\boldsymbol{A}$', so an update line for the solution is easily made. If extended with a proper initialization, this procedure is (a prototype of) the *Conjugate Gradients Squared algorithm* [5]

The fact that in the CGS algorithm no $\boldsymbol{A}^T$ operations are required, is an advantage that is generally regarded as important. For that matter: CGS shares this property with IDR. Nevertheless, because of a connection with the main stream — generalized CG variants for non-symmetric linear systems —, and also for esthetical reasons, CGS took over the control of the author's mind completely, and gradually the IDR idea 'fell asleep'.

3.4 (Bi-)CGStab against squaring the misery.

The second expected advantage of CGS over BiCG is the double use of the contractors $\varphi_n(A)$. For problems where BiCG converges, CGS converges considerable faster indeed, see figure 2. However, if BiCG has a difficult job, for

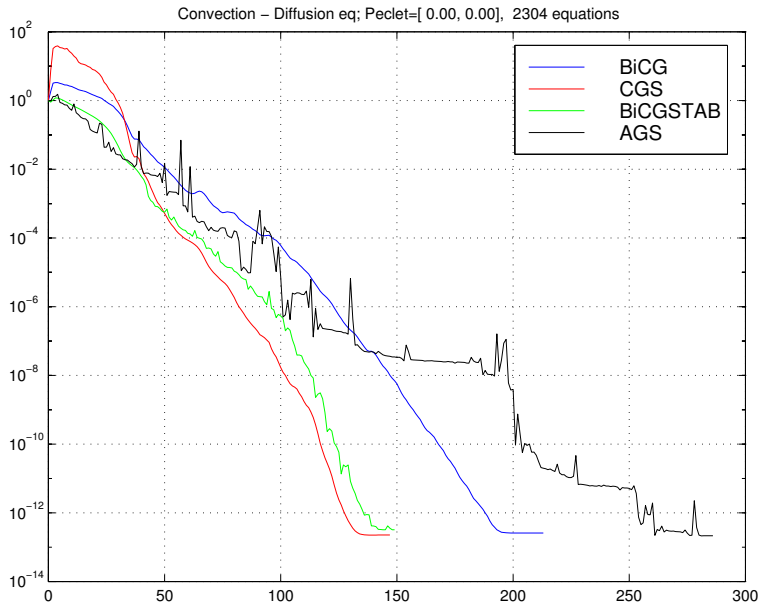

Figure 2: The advantage of squaring

CGS the job is even more difficult. Not only contraction is 'squared', but also expansion! (figure 3)
To overcome this disadvantage Henk van der Vorst suggested to replace one of the polynomials φ_n in the square φ_n^2 by a polynomial that could be used for damping out unwished behaviour of CGS. This lead to the development of BiCGSTAB [9], a method meant as a stable variant of CGS, although many people consider it as a stabilization of BiCG.
In fact BiCGSTAB was mathematically equivalent to IDR, only the recurrence formulae differ completely, since they stem from the α's and β's in the (Bi)-CG method. Both the author and Van der Vorst were well aware of that. But apparently the BiCGSTAB algorithm had a significantly better numerical stability, for which reason the IDR-interpretation was buried![3]

Of course also BiCGSTAB sometimes suffered from instability problems. This led to the development of modifications and generalizations of BiCGSTAB.

[3]This, after all, appears to be wrong. It wasn't the theoretical basis (IDR versus BiCG) that caused IDR's instability, but a not so lucky implementation of the old IDR algorithm.

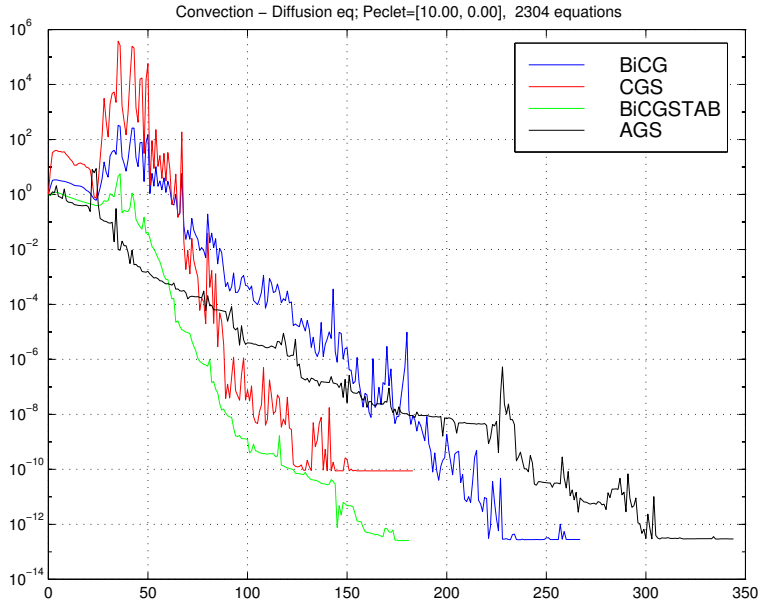

Figure 3: The price of squaring

Very clever generalizations have been developed by Martin Gutknecht, [2], Gerard Sleijpen, [4] and others.

The author had other things on his mind, and went on with a not specifically Krylov-subspace-related life.

4 Waking up

In the summer of 2006, just when the author was retiring from Delft University, he got an email by Jens Peter Zemke, from Hamburg–Harburg University, in which he asked to tell what happened to the (old) IDR method. It took a couple of weeks to reconstruct the way things went 30 years ago. Due to moving to another building, and later moving several times to different floors, some old work couldn't be found anymore. So the author started an archeological expedition in his mind, and replayed the old experiments that led to the discovery of IDR. Then he saw, that in the IDR-theorem as published in [10], page 550, the crucial recurrence relation between the spaces $\mathcal{G}_j$ was described as

$$\mathcal{G}_0 = \mathbb{R}^N, \quad \mathcal{G}_{j+1} = (\boldsymbol{I} - \omega_j \boldsymbol{A})[\mathcal{G}_j \bigcap \mathcal{S}] \tag{11}$$

where $\mathcal{S}$ denoted *any* proper subspace of $\mathbb{R}^N$, and not merely $\boldsymbol{p}^\perp$.

And then he wondered **Why didn't I try something with a subspace $\mathcal{S}$ that is significantly smaller than $p^{\perp}$**, or equivalently, use more vectors p? Some possible reasons:

1. It costs more computer time and memory, and these were expensive those days.
2. The author's intuition made him expect nothing positive from making a simple method complicated: each iteration step would require the solution of a full system of linear equations.

My thoughts today about these considerations:

1. 'More work' might produce more profit, f.i. a larger dimension reduction per step. The memory problem is wiped away by Moore's law during 25 years.
2. Intuition may never be the only argument for rejecting an idea. Besides that, today we have Matlab, so it is easy to do an experiment with more vectors p_k.

The experiment required one afternoon Matlab programming, and in figure 4 the result is shown for a one dimensional diffusion equation on a 60 points grid: That experiment was the start of IDR(s), [7], a useful extension of the 'sparse Krylov subspace toolbox'.

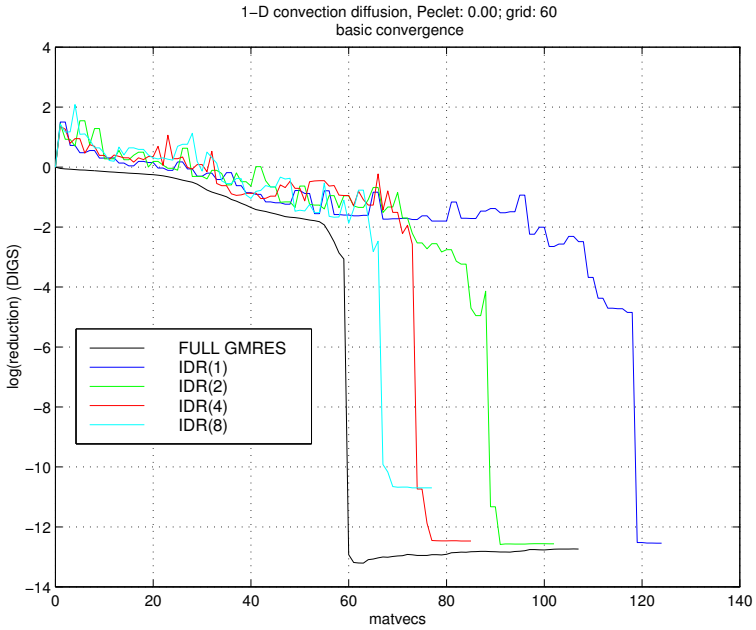

Figure 4: First IDR(s) experiment

4.1 The (simple) principle of IDR(s)

In the IDR-theorem, as published in [10], the spaces $\mathcal{G}_j$, as defined in (11) are proved to have decreasing dimension. Choosing $\mathcal{S} = \mathcal{N}(\boldsymbol{P}^T)$, where $\boldsymbol{P}$ is a $N \times s$ matrix, it can be proved that 'almost always' $\dim(\mathcal{G}_{j+1}) = \dim(\mathcal{G}_j) - s$. Construction of a sequence of vectors that are in $\mathcal{G}_j$ with increasing j is not really difficult.
Suppose we have $s+1$ vectors $\boldsymbol{g}_0, \boldsymbol{g}_1, \ldots, \boldsymbol{g}_s$, in $\mathcal{G}_j$, and let $\boldsymbol{G} = [\boldsymbol{g}_0\, \boldsymbol{g}_1\, \cdots\, \boldsymbol{g}_s]$, then if $\boldsymbol{c}$ is a vector satisfying $(\boldsymbol{P}^T\boldsymbol{G})\boldsymbol{c} = \boldsymbol{0}$, the vector $\boldsymbol{h} = \boldsymbol{G}_j\boldsymbol{c}$ belongs to $\mathcal{G}_j \bigcap \mathcal{S}$. Then the vector $\widetilde{\boldsymbol{g}} = (1 - \omega_j \boldsymbol{A})\boldsymbol{h}$ is in $\mathcal{G}_{j+1}$. It is also in $\mathcal{G}_j$ because of the nest property $\mathcal{G}_{j+1} \subset \mathcal{G}_j$. So we may repeat the procedure with one of the $\boldsymbol{g}$ vectors replaced by $\widetilde{\boldsymbol{g}}$.
After $s+1$ of these steps, all the vectors $\widetilde{\boldsymbol{g}}$ are in $\mathcal{G}_{j+1}$, and we can start with building (vectors in) $\mathcal{G}_{j+2}$....
The construction of iterates $\boldsymbol{x}_n$ for which $\boldsymbol{r}_n = \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}_n$ requires a careful scaling of the appropriate vectors

In IDR(s), we obtain a dimension reduction of s, at the cost of $(s+1)N$ matrix vector operations. So in exact arithmetic, after about $N(s+1)/s$ matvecs, we have obtained the exact solution of $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$. In practice however, for a quite broad category of sparse systems, a fairly good approximation is obtained after much less steps of the process, as is shown in figure 5. The increasing loss of

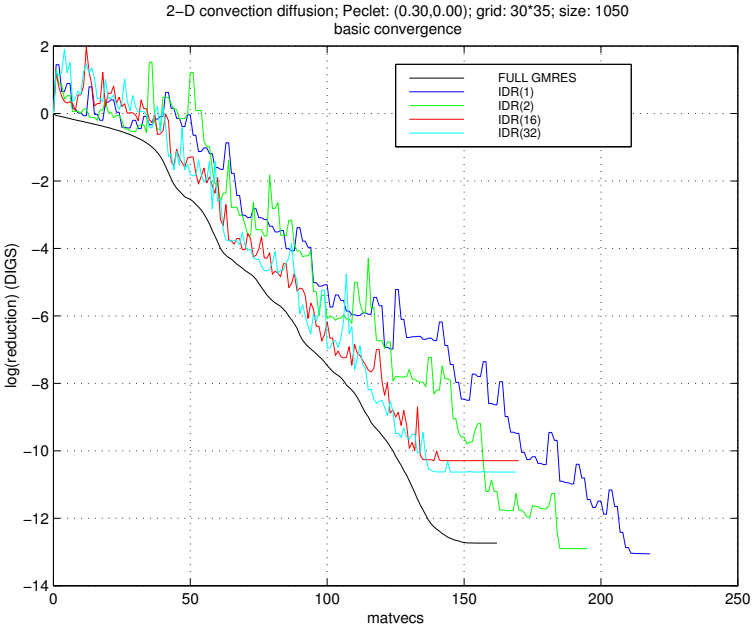

Figure 5: IDR(s) iteration for different s

digits at increasing s are only present in the first IDR(s) implementation. In a later implementation in [8], the important causes of numerical instability have been eliminated.

4.2 A convergence analysis at last

If the convergence behaviour of IDR(s) is observed for different values of s, then it can be seen that the convergence plots are shifting to the left at increasing s. The rightmost curve (the slowest) is $IDR(1)$, which is comparable to BiCGSTAB). At the higher values of s, the convergence plot is close to the curve of GMRES. This last method is the fastest, theoretically, but due to an increasing number of vector updates and inner products, GMRES becomes very expensive at high iteration numbers.
So experiments suggest that IDR(s) could be 'sold' as *poor man's GMRES*.
A convergence analysis based on contracting properties related to the spectrum of $\boldsymbol{A}$ seems to be not possible, we could try to link the convergence of IDR(s) to GMRES, which converges according to a different principle. So we try to piggyback on GMRES.
In all practical experiments, the 'shadow vectors' (the columns of $\boldsymbol{P}$), were chosen randomly (Sonneveld, [6]). Recent analysis of the convergence behaviour with respect to this choice, has produced an interesting theoretical result for not too small values of s.
As in all Krylov subspace methods, the residuals satisfy $\boldsymbol{r}_n = \Phi_n(\boldsymbol{A})\boldsymbol{r}_0$. If we concentrate on the first residual in each space $\mathcal{G}_j$, we can write

$$\Phi_{j(s+1)} = \Omega_j \cdot \Psi_{js}$$

where $\Omega_j(\boldsymbol{A})$ is a damping factor, and Ψ_{js} a kind of *Lanczos factor*. Now the *reduced residuals* $\widetilde{\boldsymbol{r}}_{js} = \Psi_{js}(\boldsymbol{A})\boldsymbol{r}_0$, can be associated with a Galerkin procedure, with a Krylov space as search space, and with almost completely random test vectors. The residual $\widetilde{\boldsymbol{r}}_k$ of such Galerkin approximation can be related to the GMRES residual $\widehat{\boldsymbol{r}}_k$ by

$$\widetilde{\boldsymbol{r}}_k - \widehat{\boldsymbol{r}}_k = \mathcal{P}\widehat{\boldsymbol{r}}_k$$

where $\mathcal{P}$ is the oblique projection matrix, with $\mathcal{R}(\mathcal{P}) = \text{span}\{\boldsymbol{r}_0, \boldsymbol{A}\boldsymbol{r}_0, \ldots, \boldsymbol{A}^k\boldsymbol{r}_0\}$ as search space, and $\mathcal{R}(\mathcal{P}^T)$ as test space.
Now if the test space is spanned by stochastically independent standard normally distributed vectors, the quantity $\xi = \|\boldsymbol{r} - \widehat{\boldsymbol{r}}\| / \|\widehat{\boldsymbol{r}}\|$ is a stochastic variable *with the same distribution as*

$$z = \|\boldsymbol{B}^{-1}\boldsymbol{u}\|$$

in which $\boldsymbol{B}$ is a $k \times k$ random matrix, and $\boldsymbol{u}$ a random vector in $\mathbb{R}^k$. Random means here: *all entries are stochastically independent standard normally distributed stochastic variables*.

In [6] the probability distribution for z is derived:

$$f_k(x) = C\frac{x^{k-1}}{(1+x^2)^{(k+1)/2}}, \quad \text{with } C = \frac{2}{\sqrt{\pi}}\frac{\Gamma(\frac{k+1}{2})}{\Gamma(\frac{k}{2})}.$$

The distributions for different k are such that the normalized stochastic variables $\widetilde{z} = \dfrac{z}{k}$ have asymptotically the same distribution for large k. See figures 6 and 7. After working out, we find for the expected number of digits that

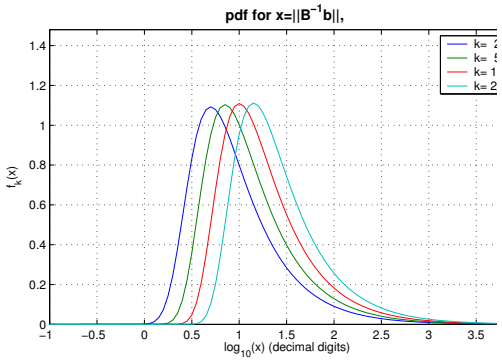

Figure 6: Probability density

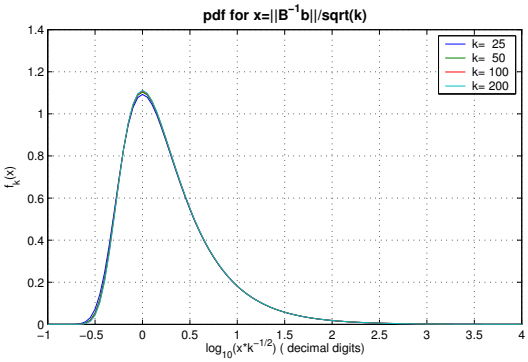

Figure 7: Normalized density

$\widetilde{\boldsymbol{r}}_k = \Psi_k(\boldsymbol{A})\boldsymbol{r}_0$ is behind the GMRES residual $\widehat{\boldsymbol{r}}_k$:

$$\mathcal{E}\left(\log_{10}(\|\widetilde{\boldsymbol{r}}_k - \widehat{\boldsymbol{r}}_k\|) - \log_{10}(\|\widehat{\boldsymbol{r}}_k\|)\right) \approx c + \log_{10}(\sqrt{k}) \tag{12}$$

with c close to 0.3. This means that at iteration steps 100 resp. 1000, we may expect to be about 1.3 resp. 1.8 decimal digits behind the GMRES procedure.
If the factors $\Omega_j(\boldsymbol{A})$ in the IDR(s) residuals are contracting, (convection diffusion equations), the convergence will be (a bit) better than shown in (12). But if $\boldsymbol{A}$ has eigenvalues in both half planes of the complex plane (Helmholz equation), *IDR(s) still converges*, albeit considerably slower because of the expanding character of $\Omega_j(\boldsymbol{A})$. These results agree with our IDR(s) experiments.

4.3 Final remarks

At least two serendipity moments can be marked in the development just described. The first was recognizing the $2N$ norm drop in the experimental secant method. Another, less obvious, was the discovery of IDR(s) when searching the history of the completely abandoned method IDR. Finally, there was the recognition of a possibly fruitful relation with GMRES in the convergence pictures of IDR(s) that triggered the convergence analysis.
The author thinks that serendipity is an important part of scientific research, and at least it is an extremely satisfying part.

According to Peter Wynn [1], 'numerical analysis is much of an experimental science', and in the IDR-CGS-IDR(s) development, the experimental part was the main source of serendipity.

So the numerical mathematician should never hesitate to do numerical experiments, nor hesitate to look not only his/her results, but also the non-results. There may be something in it!

References

[1] CLAUDE BREZINSKI AND LUC WUYTACK, *Numerical analysis in the twentieth century*, Numerical analysis : historical developments in the 20th century, pp. 1-40, Elsevier, Amsterdam. 2001

[2] M.H. GUTKNECHT, *Variants of BICGSTAB for Matrices with Complex Spectrum*, SIAM J. Sci. Comp., 14(5), pp. 1020-1033, 1993.

[3] CORNELIUS LANCZOS, *An Iteration Method for the Solution of the Eigenvalues Problem of Linear Differential and Integral Operators*, Journal of Research of the National Bureau of Standards, 45, pp. 255-282, 1950.

[4] G.L.G. SLEIJPEN AND D.R. FOKKEMA, *BiCGstab(ℓ) for linear equations involving matrices with complex spectrum*, ETNA, 1, pp. 11-32, 1994.

[5] P. SONNEVELD, *CGS: a fast Lanczos-type solver for nonsymmetric linear systems*, SIAM J. Sci. and Statist. Comput., 10, pp. 36-52, 1989.

[6] PETER SONNEVELD, *On the Convergence Behavior of IDR(s) and Related Methods*, SIAM J. Sci. Comput., 34:5, pp. A2576-A2598, 2012.

[7] PETER SONNEVELD AND MARTIN B. VAN GIJZEN, *IDR(s): a family of simple and fast algorithms for solving large nonsymmetric systems of linear equations* , SIAM J. Sci. and Statist. Comput. , 31:2, pp. 1035-1062, 2008.

[8] MARTIN B. VAN GIJZEN AND PETER SONNEVELD, *Algorithm 913: An Elegant IDR(s) Variant that Efficiently Exploits Bi-orthogonality Properties*, ACM Transactions on Mathematical Software, 38, pp. 5:1-5:19, 2011.

[9] H.A. VAN DER VORST, *Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems*, SIAM J. Sci. Comp., 13, pp. 631-644, 1992. .

[10] P. WESSELING AND P. SONNEVELD, *Numerical Experiments with a Multiple Grid- and a Preconditioned Lanczos Type Method*, Lecture Notes in Mathematics 771, Springer-Verlag, Berlin, Heidelberg, New York, pp. 543-562, 1980.