

# IDR( $s$ )

**A family of simple and fast algorithms  
for solving large nonsymmetric  
systems of linear equations**

CERFACS Sparse Days 2007

Peter Sonneveld en Martin van Gijzen

October 11, 2007

# Outline

- Introduction
- The Induced Dimension Reduction Theorem
- The  $\text{IDR}(s)$  algorithm
- Numerical experiments
- Conclusions

# Product Bi-CG methods

Bi-CG solves nonsymmetric linear systems using (short) CG recursions but needs extra matvec with  $A^H$ .

Idea of Sonneveld: use 'wasted' matvec in a more useful way.

Result: transpose-free methods:

- CGS (Sonneveld, 1989)
- Bi-CGSTAB (Van der Vorst, 1992)
- BiCGSTAB2 (Gutknecht, 1993)
- TFQMR (Freund, 1993)
- BiCGstab( $\ell$ ) (Sleijpen and Fokkema, 1994)
- Many other variants

# Historical remarks

Sonneveld first developed IDR (1980).

Analysis showed that IDR was Bi-CG combined with linear minimal residual steps.

The fact that IDR is transpose free, combined with the relation with Bi-CG led to the development of a now famous algorithm: CGS.

Later Van der Vorst proposed another famous method: Bi-CGSTAB, which is mathematically equivalent with IDR.

As a result of these developments, the basic IDR idea was abandoned for the Bi-CG approach. IDR is now forgotten.

# The IDR idea

The IDR-idea is to generate a sequence of subspaces  $\mathcal{G}_0 \cdots \mathcal{G}_j$  with the following operations:

- Intersect  $\mathcal{G}_{j-1}$  with fixed subspace  $\mathcal{S}$ ,
- Compute  $\mathcal{G}_j = (\mathbf{I} - \omega_j \mathbf{A})(\mathcal{G}_{j-1} \cap \mathcal{S})$ .

# The IDR theorem

**Theorem 1 (IDR)** *Let  $A$  be any matrix in  $\mathbb{C}^{N \times N}$ , let  $v_0$  be any nonzero vector in  $\mathbb{C}^N$ , and let  $\mathcal{G}_0$  be the complete Krylov space  $\mathcal{K}^N(A, v_0)$ . Let  $\mathcal{S}$  denote any (proper) subspace of  $\mathbb{C}^N$  such that  $\mathcal{S}$  and  $\mathcal{G}_0$  do not share a nontrivial invariant subspace of  $A$ , and define the sequence  $\mathcal{G}_j$ ,  $j = 1, 2, \dots$  as*

$$\mathcal{G}_j = (\mathbf{I} - \omega_j \mathbf{A})(\mathcal{G}_{j-1} \cap \mathcal{S})$$

*where the  $\omega_j$ 's are nonzero scalars. Then*

- (i)  $\mathcal{G}_j \subset \mathcal{G}_{j-1}$  for all  $j > 0$ .*
- (ii)  $\mathcal{G}_j = \{\mathbf{0}\}$  for some  $j \leq N$ .*

# Making an IDR algorithm

The IDR theorem can be used to construct solution algorithms.

This is done by constructing residuals  $r_n \in \mathcal{G}_j$ .

According to the IDR theorem ultimately  $r_n \in \mathcal{G}_M = \{\mathbf{0}\}$ .

In order to generate residuals and corresponding solution approximations we first look at the basic recursions.

# Krylov methods: basic recursion (1)

A Krylov-type solver produces iterates  $x_n$ , for which the residuals  $r_n = b - Ax_n$  are in the Krylov spaces

$$\mathcal{K}^n(A, r_0) = r_0 \oplus Ar_0 \oplus A^2r_0 \oplus \cdots \oplus A^n r_0 ,$$

The next residual  $r_{n+1}$  can be generated by

$$r_{n+1} = -\alpha Ar_n + \sum_{j=0}^{\hat{j}} \beta_j r_{n-j} .$$

The parameters  $\alpha, \beta_j$  determine the specific method and must be such that  $x_{n+1}$  can be computed.

# Krylov methods: basic recursion (2)

By using the difference vector

$$\Delta \mathbf{r}_k = \mathbf{r}_{k+1} - \mathbf{r}_k = -\mathbf{A}(\mathbf{x}_{n+1} - \mathbf{x}_n),$$

an explicit way to satisfy this requirement is

$$\mathbf{r}_{n+1} = \mathbf{r}_n - \alpha \mathbf{A} \mathbf{r}_n - \sum_{j=1}^{\hat{j}} \gamma_j \Delta \mathbf{r}_{n-j},$$

which leads to the following update for the  $\mathbf{x}$  estimate:

$$\mathbf{x}_{n+1} = \mathbf{x}_n + \alpha \mathbf{r}_n - \sum_{j=1}^{\hat{j}} \gamma_j \Delta \mathbf{x}_{n-j},$$

# Computation of a new residual (1)

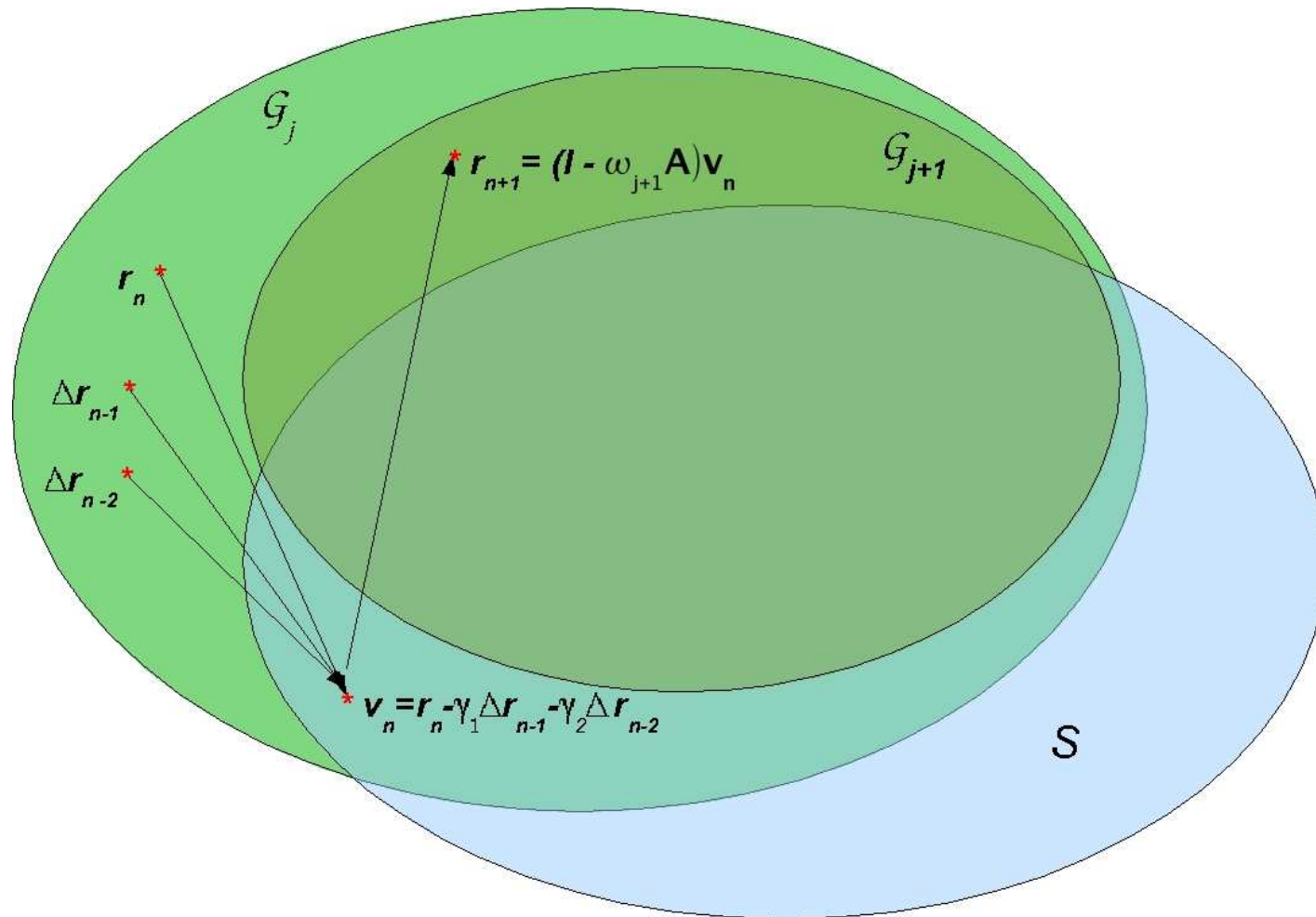
Residuals are computed that are forced to be in the subspaces  $\mathcal{G}_j$ , by application of the IDR-theorem.

The residual  $\mathbf{r}_{n+1}$  is in  $\mathcal{G}_{j+1}$  if

$$\mathbf{r}_{n+1} = (\mathbf{I} - \omega_{j+1}\mathbf{A})\mathbf{v} \quad \text{with } \mathbf{v} \in \mathcal{G}_j \cap \mathcal{S} .$$

The main problem is to find  $\mathbf{v}$ .

# Computation of a new residual (2)



# Computation of a new residual (3)

The vector  $v$  is a combination of the residuals  $r_l$  in  $\mathcal{G}_j$ .

$$v = r_n - \sum_{j=1}^{\hat{j}} \gamma_j \Delta r_{n-j} .$$

Let the space  $\mathcal{S}$  be the left null space of some  $N \times s$  matrix  $P$ :

$$P = (p_1 \ p_2 \ \dots \ p_s), \quad \mathcal{S} = \mathcal{N}(P^H) .$$

Since  $v$  is also in  $\mathcal{S} = \mathcal{N}(P^H)$ , it must satisfy

$$P^H v = 0 .$$

Combining these two yields an  $s \times \hat{j}$  linear system for the coefficients  $\gamma_j$  that (normally) is uniquely solvable if  $\hat{j} = s$ .

# Computation of a new residual (4)

Hence with the residual  $\mathbf{r}_n$ , and a matrix  $\Delta\mathbf{R}$  consisting of the last  $s$  residual differences:

$$\Delta\mathbf{R} = (\Delta\mathbf{r}_{n-1} \ \Delta\mathbf{r}_{n-2} \ \dots \ \Delta\mathbf{r}_{n-s})$$

a suitable  $\mathbf{v}$  can be found by

Solve  $s \times s$  system  $(\mathbf{P}^H \Delta\mathbf{R})\mathbf{c} = \mathbf{P}^H \mathbf{r}_n$

Calculate  $\mathbf{v} = \mathbf{r}_n - \Delta\mathbf{R}\mathbf{c}$

# Building $\mathcal{G}_{j+1}$ (1)

Assume  $r_n$  and all columns of  $\Delta R$  are in  $\mathcal{G}_j$ , and let  $r_{n+1}$  be calculated as

$$r_{n+1} = v - \omega_{j+1} A v$$

Then  $r_{n+1} \in \mathcal{G}_{j+1}$ .

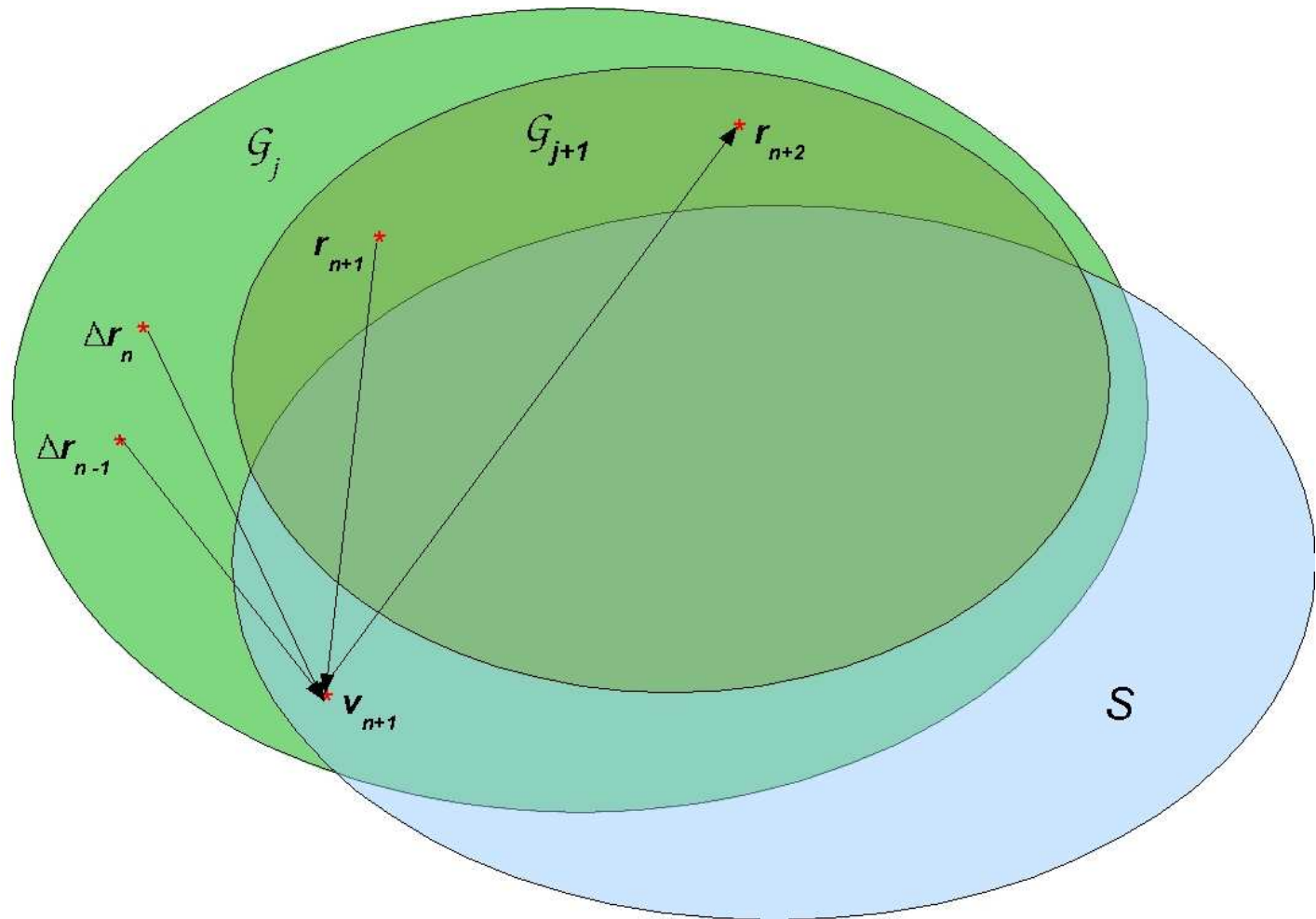
Since  $\mathcal{G}_{j+1} \subset \mathcal{G}_j$  (theorem) we automatically have

$$r_{n+1} \in \mathcal{G}_j$$

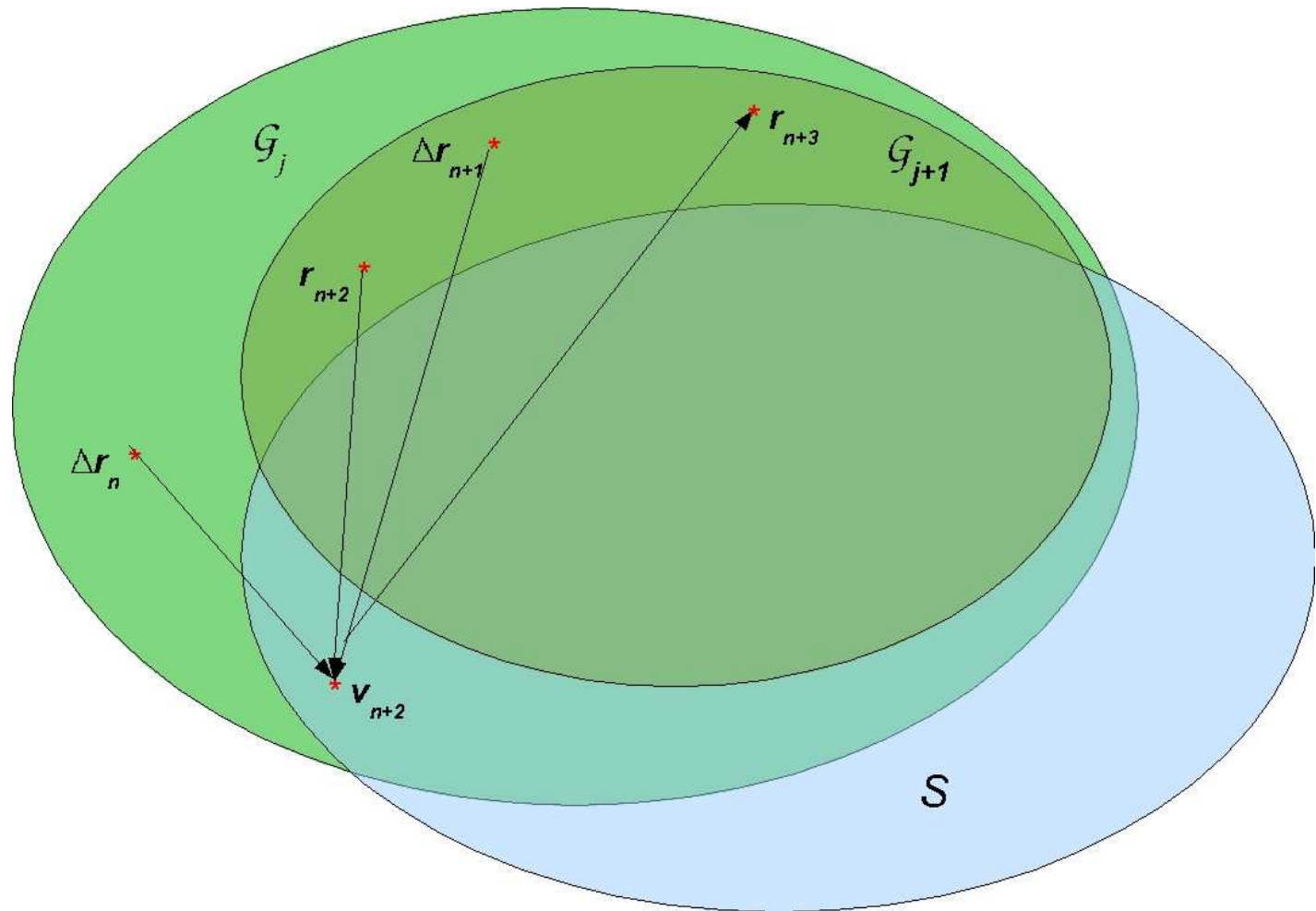
Now the next  $\Delta R$  is made by repeating the calculations.

In this way we find  $s + 1$  residuals in  $\mathcal{G}_{j+1}$

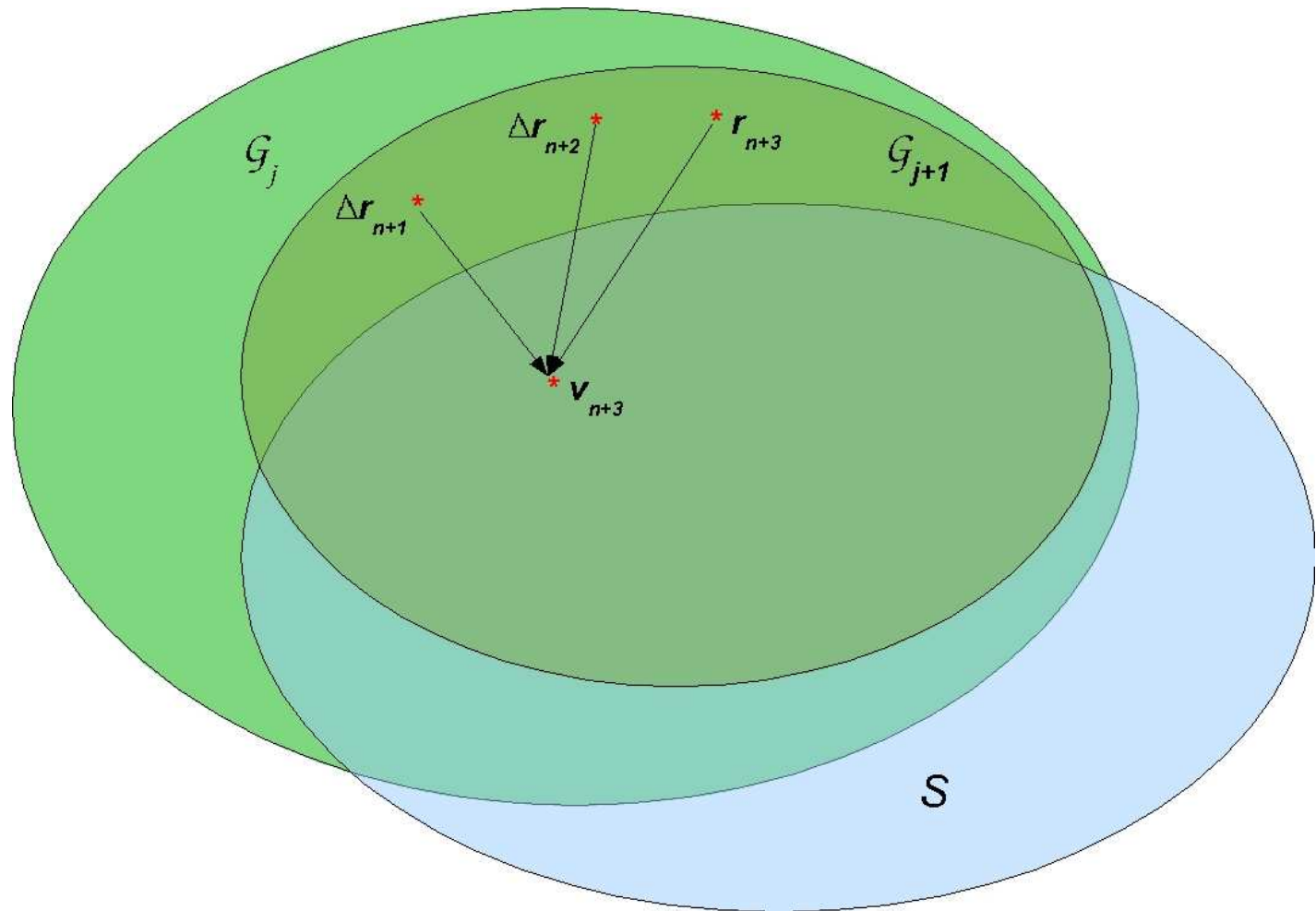
# Building $\mathcal{G}_{j+1}$ (2)



# Building $\mathcal{G}_{j+1}$ (3)



# Building $\mathcal{G}_{j+1}$ (4)



# A few details

1. The first  $s + 1$  residuals, starting with  $r_0$  can be constructed by any Krylov-based iteration, such as a local minimum residual method.
2. In our actual implementation, all steps are identical. However, in calculating the first residual in  $\mathcal{G}_{j+1}$ , a new value  $\omega_{j+1}$  may be chosen. We choose  $\omega_{j+1}$  such that  $\|\mathbf{v} - \omega_{j+1}\mathbf{A}\mathbf{v}\|$  is minimal.

# Basic IDR( $s$ ) algorithm.

**while**  $\|\mathbf{r}_n\| > TOL$  **or**  $n < MAXIT$  **do**

**for**  $k = 0$  **to**  $s$  **do**

Solve  $\mathbf{c}$  from  $\mathbf{P}^H d\mathbf{R}_n \mathbf{c} = \mathbf{P}^H \mathbf{r}_n$

$\mathbf{v} = \mathbf{r}_n - d\mathbf{R}_n \mathbf{c}; \mathbf{t} = \mathbf{A}\mathbf{v};$

**if**  $k = 0$  **then**

$$\omega = (\mathbf{t}^H \mathbf{v}) / (\mathbf{t}^H \mathbf{t});$$

**end if**

$d\mathbf{r}_n = -d\mathbf{R}_n \mathbf{c} - \omega \mathbf{t}; d\mathbf{x}_n = -d\mathbf{X}_n \mathbf{c} + \omega \mathbf{v};$

$\mathbf{r}_{n+1} = \mathbf{r}_n + d\mathbf{r}_n; \mathbf{x}_{n+1} = \mathbf{x}_n + d\mathbf{x}_n;$

$n = n + 1;$

$d\mathbf{R}_n = (d\mathbf{r}_{n-1} \cdots d\mathbf{r}_{n-s}); d\mathbf{X}_n = (d\mathbf{x}_{n-1} \cdots d\mathbf{x}_{n-s});$

**end for**

**end while**

# Vector operations per MATVEC

Method	DOT	AXPY	Memory Requirements
IDR(1)	2	4	8
IDR(2)	$2\frac{2}{3}$	$5\frac{5}{6}$	11
IDR(4)	$4\frac{2}{5}$	$9\frac{7}{10}$	17
IDR(6)	$6\frac{2}{7}$	$13\frac{9}{14}$	23
Full GMRES	$\frac{n+1}{2}$	$\frac{n+1}{2}$	$n + 2$
BiCGSTAB	2	3	7

# Relation with other methods

Although the approach is different,  $IDR(s)$  is closely related to some Bi-CGSTAB methods:

- $IDR(1)$  and Bi-CGSTAB yield the same residuals at the even steps.
- $ML(k)BiCGSTAB$  (Yeung and Chan, 1999) seems closely related to  $IDR(s)$ , BUT
  - $IDR(s)$  is MUCH simpler (both conceptually and its implementation)
  - Other, more natural extensions are possible, e.g. to avoid breakdown.

# Performance of IDR( $s$ )

The IDR theorem states that

- it is possible to generate a sequence of nested subspace  $\mathcal{G}_j$  of shrinking dimension,
- but does not say how fast the dimension shrinks

It can be proven that the dimension reduction is (normally)  $s$ ,

So  $\dim(\mathcal{G}_{j+1}) = \dim(\mathcal{G}_j) - s$ .

IDR( $s$ ) requires at at most  $N + \frac{N}{s}$  matrix-vector multiplications to compute the exact solution.

# Numerical experiments

We will present two typical numerical examples

- A 2D Ocean Circulation Problem
- A 3D Helmholtz Problem

# A 2D Ocean Circulation Problem

We compare IDR( $s$ ) with Full GMRES, restarted GMRES and Bi-CGSTAB.

This ocean example is representative for a wide class of CFD problems.

We will compare:

- Rate of convergence
- Stagnation level (of the true residual norm)

# Stommel's model for ocean circulation

Balance between bottom friction, wind stress and Coriolis force.

$$-r \Delta \psi - \beta \frac{\partial \psi}{\partial x} = (\nabla \times \mathbf{F})_z$$

plus circulation condition around islands  $k$

$$\oint_{\Gamma_k} r \frac{\partial \psi}{\partial n} ds = - \oint_{\Gamma_k} \mathbf{F} \cdot \mathbf{s} ds.$$

- $\psi$ : streamfunction
- $r$ : bottom friction parameter
- $\beta$ : Coriolis parameter
- $\mathbf{F}$ : Wind stress

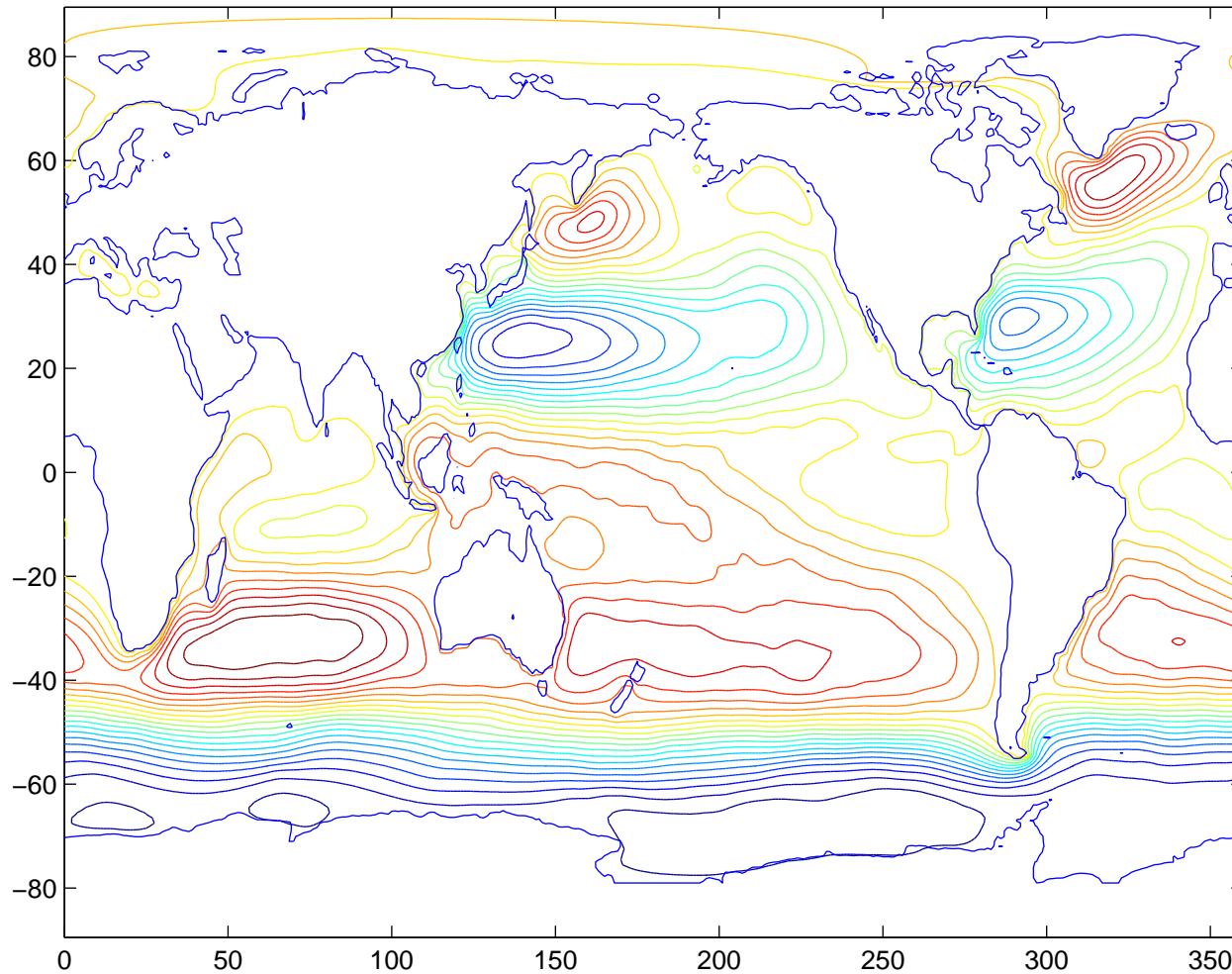
# Discretization of the ocean problem

- Discretization with linear finite elements
- Results in nonsymmetric system of 42248
- Eigenvalues are (almost) real

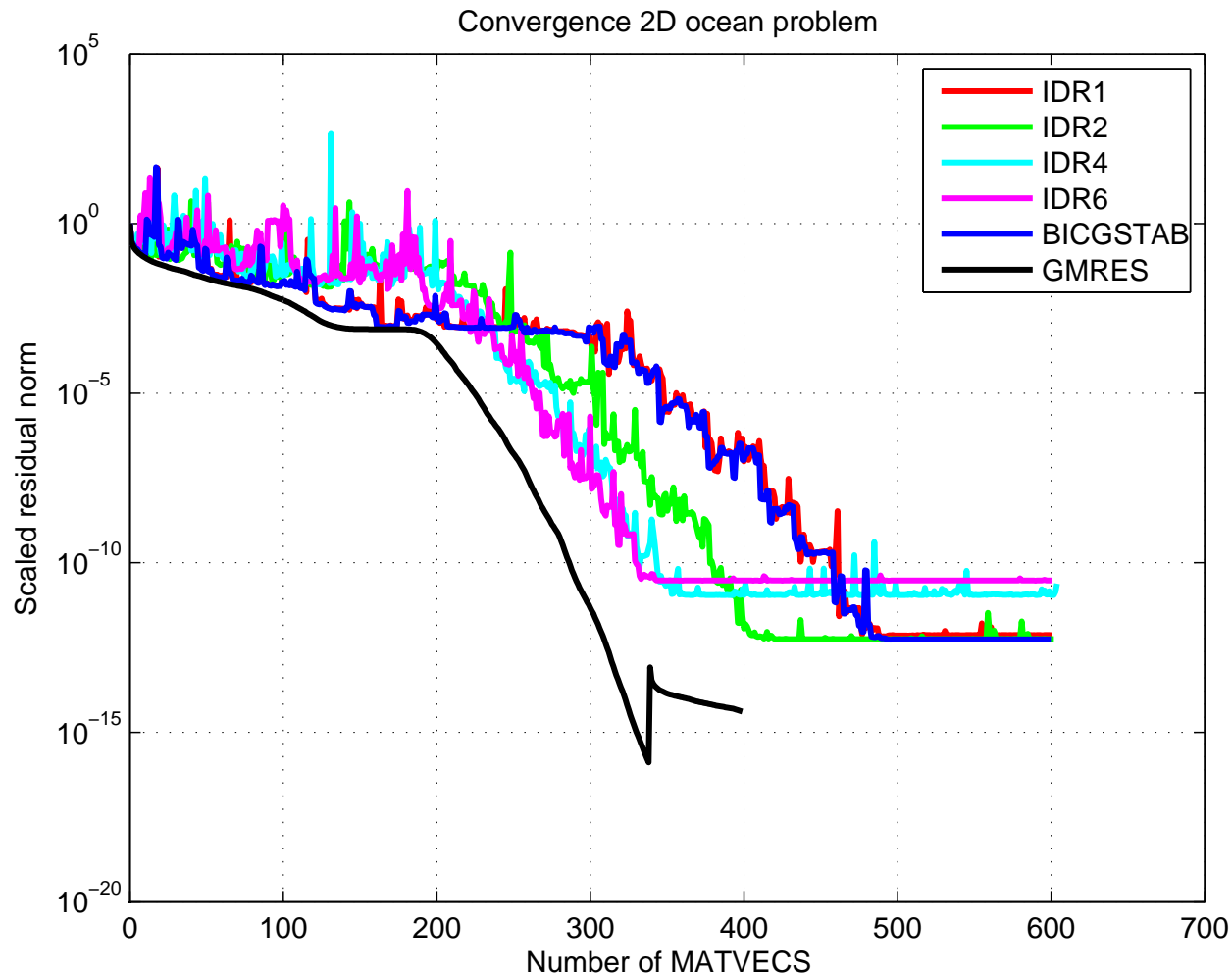
Solution parameters:

- ILU(0) preconditioning
- $P$ :  $s - 1$  random vectors plus  $r_0$  (for comparison with Bi-CGSTAB)

# Solution of the ocean problem



# Convergence for the ocean problem



# Some observations

- Required number of MATVECS decreases if  $s$  is increased. IDR(4) and IDR(6) are close to the optimal convergence curve of full GMRES.
- Convergence curves of IDR(1) and Bi-CGSTAB coincide.
- Stagnation levels of IDR( $s$ ) comparable with Bi-CGSTAB.

# Required number of MATVECS

Method	Number of MATVECS	Vectors
Full GMRES	265	268
GMRES(20)	> 10000	23
GMRES(50)	4671	53
Bi-CGSTAB	411	7
IDR(1)	420	8
IDR(2)	339	11
IDR(4)	315	17
IDR(6)	307	23

Tolerance:  $\|b - Ax_n\| < 10^{-8} \|b\|$

# A 3D Helmholtz Problem

Example models sound propagation in a room of  $4 \times 4 \times 4m^3$ .

A harmonic sound source gives acoustic pressure field

$$p(\mathbf{x}, t) = \hat{p}(\mathbf{x})e^{2\pi ift}.$$

The pressure function  $\hat{p}$  can be determined from

$$\frac{-(2\pi f)^2}{c^2}\hat{p} - \Delta\hat{p} = \delta(\mathbf{x} - \mathbf{x}_s) \quad \text{in } \Omega.$$

in which

- $c$ : the sound speed (340 m/s)
- $\delta(\mathbf{x} - \mathbf{x}_s)$ : the harmonic point source, in the center of the room.

# Boundary conditions

Five of the walls are reflecting, modeled by

$$\frac{\partial \hat{p}}{\partial n} = 0 ,$$

and the remaining wall is sound absorbing,

$$\frac{\partial \hat{p}}{\partial n} = -\frac{2\pi i f}{c} \hat{p} \text{ on } \Gamma_3.$$

# Discretization

Discretization with FEM yields linear system

$$[-(2\pi f)^2 \mathbf{M} + 2\pi i f \mathbf{C} + \mathbf{K}] \mathbf{p} = \mathbf{b}$$

- Frequency  $f = 100 \text{ Hz}$ .
- System matrix complex, symmetric (but not Hermitian) and indefinite: **difficult for iterative methods**
- gridsize  $h = 8 \text{ cm}$ : 132651 equations

Solution parameters:

- ILU(0) preconditioning
- $P$ : Initial residual plus  $s - 1$  real random vectors
- Only comparison with BiCGstab( $\ell$ )

# Results Helmholtz Problem

Method	Number of MATVECS	Elapsed time [s]
IDR(1)	1500	3322
IDR(2)	598	1329
IDR(4)	353	783
IDR(6)	310	698
BiCGstab(1)	1828	3712
BiCGstab(2)	1008	2045
BiCGstab(4)	656	1362
BiCGstab(8)	608	1337

# Conclusions

- The IDR-theorem offers a new approach for the development of iterative solution algorithms
- The  $\text{IDR}(s)$  algorithm presented here is quite promising and seems to outperform state-of-the-art Bi-CG-type methods for important classes of problems.

More information:

<http://ta.twi.tudelft.nl/nw/users/gijzen/software.html>

- Report:  $\text{IDR}(s)$ : a family of simple and fast algorithms for solving large nonsymmetric linear systems, submitted
- Matlab code, (includes preconditioning and deflation)