Iterative Methods for Linear Systems of Equations

Projection methods (3)

ITMAN PhD-course DTU 20-10-08 till 24-10-08

Martin van Gijzen

Overview day 4

- Bi-Lanczos method
- Computation of an approximate solution:
    - Bi-CG
    - QMR
    - CGS and Bi-CGSTAB
    - Convergence
- The induced dimension theorem and IDR(s)

Introduction

Yesterday, we discussed Arnoldi-based methods. A method like GMRES minimises the residual norm, but has to compute and store a new orthogonal basis vector in each iteration. The cost of doing this becomes prohibitive if many iterations have to be performed.

Today, we will discuss another class of iterative methods. They are based on the bi-Lanczos method, and hence closely related to CG. They use short recurrence but the iterates have no optimality property.

We will also discuss a new method: IDR(s). This method is not based on the bi-Lanczos method, but is mathematically related.

TUDelft

The Bi-Lanczos method

The symmetric Lanczos method constructs a matrix Q_k such that $Q_k^T Q_k = I$ and

$$Q_k^T A Q_k = T_k \quad \text{tridiagonal} .$$

The Bi-Lanczos algorithm constructs matrices W_k and V_k such that

$$W_k{}^T A V_k = T_k \quad \text{tridiagonal}$$

and such that

$$W_k{}^T V_k = I$$

Hence W_k and V_k are not orthogonal themselves, but they form a bi-orthogonal pair.

The Bi-Lanczos method (2)

Choose two vectors v_1 and w_1 such that $v_1^T w_1 = 1$

```
$\beta_1 = \delta_1 = 0$    $w_0 = v_0 = 0$                                             initialization
FOR                          $k = 1, \cdots$ DO                                          iteration
                             $\alpha_k = w_k^T A v_k$
                             $\hat{v}_{k+1} = A v_k - \alpha_k v_k - \beta_k v_{k-1}$       new
                             $\hat{w}_{k+1} = A^T w_k - \alpha_k w_k - \delta_k w_{k-1}$    direction $v$
                             $\delta_{k+1} = |\hat{v}_{k+1}^T \hat{w}_{k+1}|$               orthogonal to
                             $\beta_{k+1} = \hat{v}_{k+1}^T \hat{w}_{k+1} / \delta_{k+1}$   previous $w$.
                             $w_{k+1} = \hat{w}_{k+1} / \beta_{k+1}$
                             $v_{k+1} = \hat{v}_{k+1} / \delta_{k+1}$
END FOR
```

The Bi-Lanczos method (3)

Let

$$T_k = \begin{bmatrix} \alpha_1 & \beta_2 & & & 0 \\ \delta_2 & \alpha_2 & \ddots & & \\ & \ddots & \ddots & \ddots & \\ & 0 & \ddots & \ddots & \beta_k \\ & & & \delta_k & \alpha_k \end{bmatrix}$$

and

$$V_k = [v_1 \; v_2 \ldots v_k] \quad W_k = [w_1 \; w_2 \ldots w_k] \; .$$

Then $$AV_k = V_k T_k + \delta_{k+1} v_{k+1} e_k^T$$

and $$A^T W_k = W_k T_k^T + \beta_{k+1} w_{k+1} e_k^T \; .$$

The Bi-Lanczos method (3)

Note that

$$span\{v_1 \cdots v_k\} \text{ is a basis for } K^k(A; v_1)$$

and

$$span\{w_1 \cdots w_k\} \text{ is a basis for } K^k(A^T; w_1)$$

Breakdown of Bi-Lanczos

Bi-Lanczos breaks down if

$$w_{k+1}^T v_{k+1} = 0 \ .$$

This can happen in the following cases:

$v_{k+1} = 0 : \quad \{v_1 \cdots v_k\}$ span an invariant subspace (wrt. A)

$w_{k+1} = 0 : \quad \{w_1 \cdots w_k\}$ span an invariant subspace (wrt. A^T)

$w_{k+1} \neq 0$ and $v_{k+1} \neq 0 :$ serious breakdown

Bi-CG

A CG-type method can now be defined as:

$v_1 = r_0/\|r_0\|$ and $w_1 = \tilde{r}_0/\|\tilde{r}_0\|$ (with e.g. $\tilde{r}_0 = r_0$).

- Compute W_k and V_k using Bi-Lanczos
- Compute $x_k = x_0 + V_k y_k$ with y_k such that

$$W_k^T A V_k y_k = W_k^T r_0 \Leftrightarrow T_k y_k = \|r_0\| e_1$$

This algorithm can be cast in a more practical form in the same way as CG. The resulting algorithm is called Bi-CG.

The Bi-CG algorithm

```
$r_0 = b - Ax_0$   Choose $\tilde{r_0}$, $p_0 = r_0, \tilde{p_0} = \tilde{r_0}$   initialization
FOR                $k = 0, 1, \cdots,$ DO
                   $\alpha_k = \frac{\tilde{r}_k^T r_k}{\tilde{p}_k^T A p_k}$
                   $x_{k+1} = x_k + \alpha_k p_k$                update iterate
                   $r_{k+1} = r_k - \alpha_k A p_k$              update residual
                   $\tilde{r}_{k+1} = \tilde{r}_k - \alpha_k A^T \tilde{p}_k$   shadow residual
                   $\beta_k = \frac{\tilde{r}_{k+1}^T r_{k+1}}{\tilde{r}_k^T r_k}$
                   $p_{k+1} = r_{k+1} + \beta_k p_k$             update direction vector
                   $\tilde{p}_{k+1} = \tilde{r}_{k+1} + \beta_k \tilde{p}_k$   shadow direction vector
END FOR
```

Properties of Bi-CG

- The method uses limited memory: only five vectors need to be stored;
- The method is not optimal. However, if A is SPD and with $\tilde{r}_0 = r_0$ we get CG back (but with two times the number of operations per iteration!)
- The method is finite: all the residuals are orthogonal to the shadow residuals $\tilde{r}_i^T r_j = 0$ if $i \neq j$.
- The method is not robust: $\tilde{r}_k^T r_k$ may be zero and $r_k \neq 0$.

’Quasi’ minimising the residuals

In analogy to CG and MINRES (and to FOM and GMRES) we can define a ’quasi’-minimal residual algorithm.

To this end, we first rewrite the Bi-Lanczos relations.

The Bi-Lanczos relations

Define $\underline{T}_k$ as

$$\underline{T}_k = \begin{bmatrix} \alpha_1 & \beta_2 & & & \\ \delta_2 & \alpha_2 & \ddots & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & \beta_k \\ & & & \delta_k & \alpha_k \\ & & & & \delta_{k+1} \end{bmatrix} .$$

We can now write

$$AV_k = V_{k+1}\underline{T}_k$$

Quasi minimal residuals (1)

We would like to solve the problem: find $x_k = x_0 + V_k y_k$ such that $\|r_k\|$ is minimal.

$$r_k = b - Ax_k = r_0 - AV_k y_k = \|r_0\| v_1 - AV_k y_k$$

Hence the problem is to minimise

$$\begin{aligned} \|r_k\| &= \| \|r_0\| v_1 - AV_k y_k \| && (1) \\ &= \| \|r_0\| V_{k+1} e_1 - V_{k+1} \underline{T}_k y_k \| \ . && (2) \end{aligned}$$

If V_{k+1} had orthonormal columns this would be equivalent to: Minimise

$$\|r_k\| = \| \|r_0\| e_1 - \underline{T}_k y_k \|$$

Quasi minimal residuals (2)

However, in general V_k does not have orthogonal columns.

We could ignore this fact and still compute y_k by minimising

$$\| \|r_0\| e_1 - \underline{T}_k y_k \|$$

The resulting algorithm is called QMR. For many problems, the convergence of QMR is smoother than of Bi-CG, but not essentially faster.

Bi-CG wastes time

Bi-CG constructs its approximations as a linear combination of the columns of V_k. The only reason why we construct shadow vectors is to calculate the parameters α_k and β_k.

Another disadvantage of Bi-CG and QMR is that operations with A^T have to be performed, and this may be difficult in matrix-free computations.

Next we look at the following questions:

- Can the computational work of Bi-CG be better exploited?
- Can operations with A^T be avoided?

Bi-CG: a closer look

In the Bi-CG method, the residual vector can be expressed as

$$r_k = \phi_k(A) r_0$$

and the direction vector as

$$p_k = \pi_k(A) r_0$$

with ϕ_k and π_k polynomials of degree k.

The shadow vectors $\tilde{r}_k$ and $\tilde{p}_k$ are defined through the same recurrences as r_k and p_k. Hence

$$\tilde{r}_k = \phi_k(A^T)\tilde{r}_0, \qquad \tilde{p}_k = \pi_k(A^T)\tilde{r}_0$$

Towards a faster algorithm

The scalar α_k in Bi-CG is given by

$$\alpha_k = \frac{(\phi_k(A^T)\tilde{r}_0)^T(\phi_k(A)r_0)}{(\pi_k(A^T)\tilde{r}_0)^T(A\pi_k(A)r_0)} = \frac{\tilde{r}_0^T\phi_k^2(A)r_0}{\tilde{r}_0^T A\pi_k^2(A)r_0}$$

The idea is now to find an algorithm which gives residuals $\hat{r}_k$ that satisfy

$$\hat{r}_k = {\phi_k}^2(A)r_0.$$

The algorithm that is based on this idea is called CGS (by Sonneveld in 1989).

Conjugate Gradients Squared

```
$x_0$ is an initial guess; $r_0 = b - Ax_0$;
$\tilde{r}_0$ arbitrary, such that $r_0^T \tilde{r}_0 \neq 0$ , $\rho_0 = r_0^T \tilde{r}_0)$ ;
$\beta_{-1} = \rho_0$ ; $p_{-1} = q_0 = 0$ ;
FOR $i = 0, 1, 2, ...$ DO
    $u_i = r_i + \beta_{i-1} q_i$ ;
    $p_i = u_i + \beta_{i-1}(q_i + \beta_{i-1} p_{i-1})$ ;
    $\hat{v} = Ap_i$ ;
    $\alpha_i = \frac{\rho_i}{\tilde{r}_0^T \hat{v}}$ ;
    $q_{i+1} = u_i - \alpha_i \hat{v}$ ;
    $s_i = u_i + q_{i+1}$
    $x_{i+1} = x_i + \alpha_i (u_i + q_{i+1})$ ;
    $r_{i+1} = r_i - \alpha_i A(u_i + q_{i+1})$ ;
    $\rho_{i+1} = \tilde{r}_0^T r_{i+1}$ ; $\beta_i = \frac{\rho_{i+1}}{\rho_i}$ ;
END FOR
```

CGS (2)

For many problems CGS converges considerably faster than Bi-CG, sometimes twice as fast.

However, convergence is more erratic. Bi-CG shows peaks in the convergence curves. These peaks are squared by CGS.

The peaks can destroy the accuracy of the solution and also convergence of the method. The search for a more smoothly converging method has led to the development of Bi-CGSTAB by Van der Vorst.

Bi-CGSTAB

Bi-CGSTAB is based on the following observation. Instead of squaring the Bi-CG polynomial, we can construct other iteration methods by which x_k are generated so that

$$r_k = \psi_k(A)\phi_k(A)r_0$$

Bi-CGSTAB takes a polynomial of the form

$$\psi_k(x) = (1 - \omega_1 x)(1 - \omega_2 x)...(1 - \omega_k x)$$

The ω_k are chosen to minimise the current residual in the same way as in the one-step minimal residual method.

Bi-CGSTAB: the algorithm

```
$x_0$ is an initial guess; $r_0 = b - Ax_0$;
$\tilde{r}_0$ arbitrary, such that $r_0^T \tilde{r}_0 \neq 0$ , $\rho_{-1} = \alpha_{-1} = \omega_{-1} = 1$ ;
$v_{-1} = p_{-1} = 0$ ;
FOR $k = 0, 1, 2, \ldots$ DO
    $\rho_k = \tilde{r}_0^T r_k$ ; $\beta_{k-1} = (\rho_k / \rho_{k-1})(\alpha_{k-1} / \omega_{k-1})$ ;
    $p_k = r_k + \beta_{k-1}(p_{k-1} - \omega_{k-1} v_{k-1})$ ;
    $v_k = A p_k$ ;
    $\alpha_k = \rho_k / (\tilde{r}_0^T v_k)$ ;
    $s = r_k - \alpha_k v_k$ ;
    $t = As$ ;
    $\omega_k = (t^T s)/(t^T t)$ ;
    $x_{k+1} = x_k + \alpha_k p_k + \omega_k s$ ;
    $r_{k+1} = s - \omega_k t$ ;
END FOR
```

Convergence of Bi-CG methods

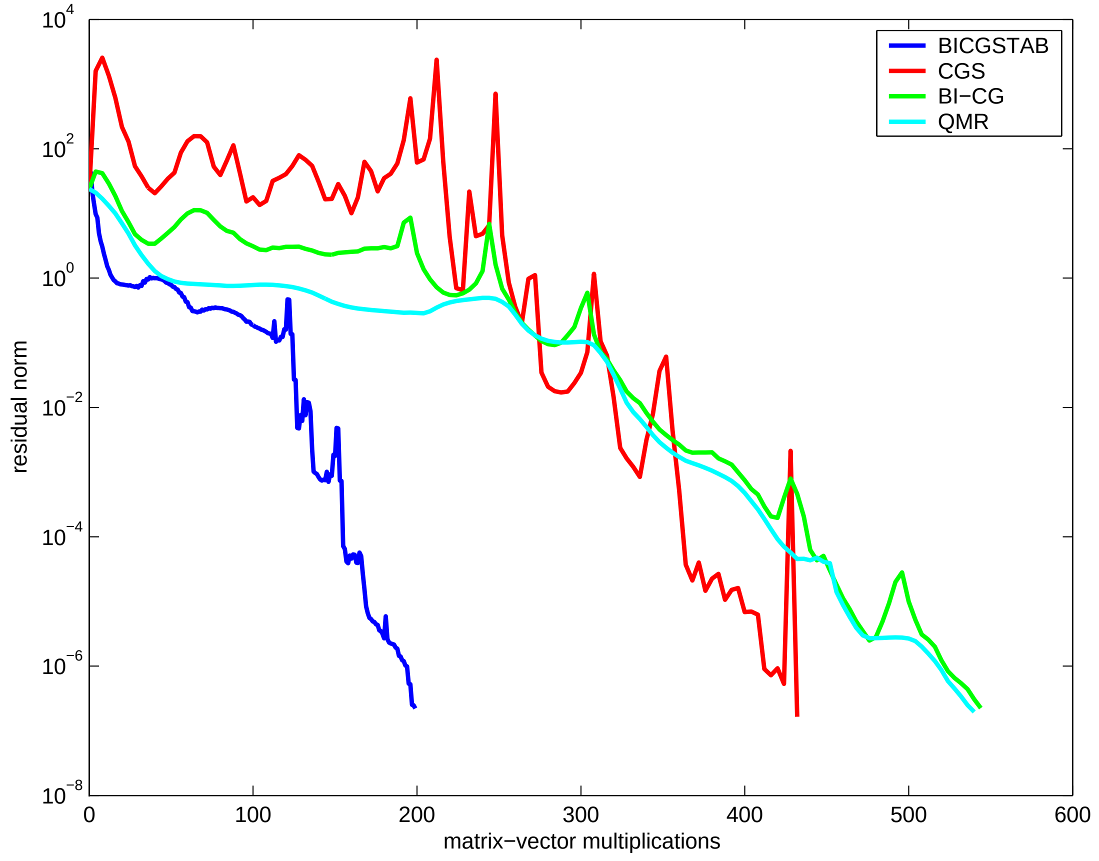
No bounds on the residual norm can be given for the four methods of today: they have no optimality property.

Some general observations can be made:

- All four methods that we discussed today are finite. They all show superlinear convergence.
- The rate of convergence of Bi-CG and QMR is basically the same, but convergence of QMR is smoother.
- Convergence of Bi-CGSTAB and CGS is most of the time faster than of the other two methods. But convergence of CGS can be erratic which has a negative effect on the accuracy.

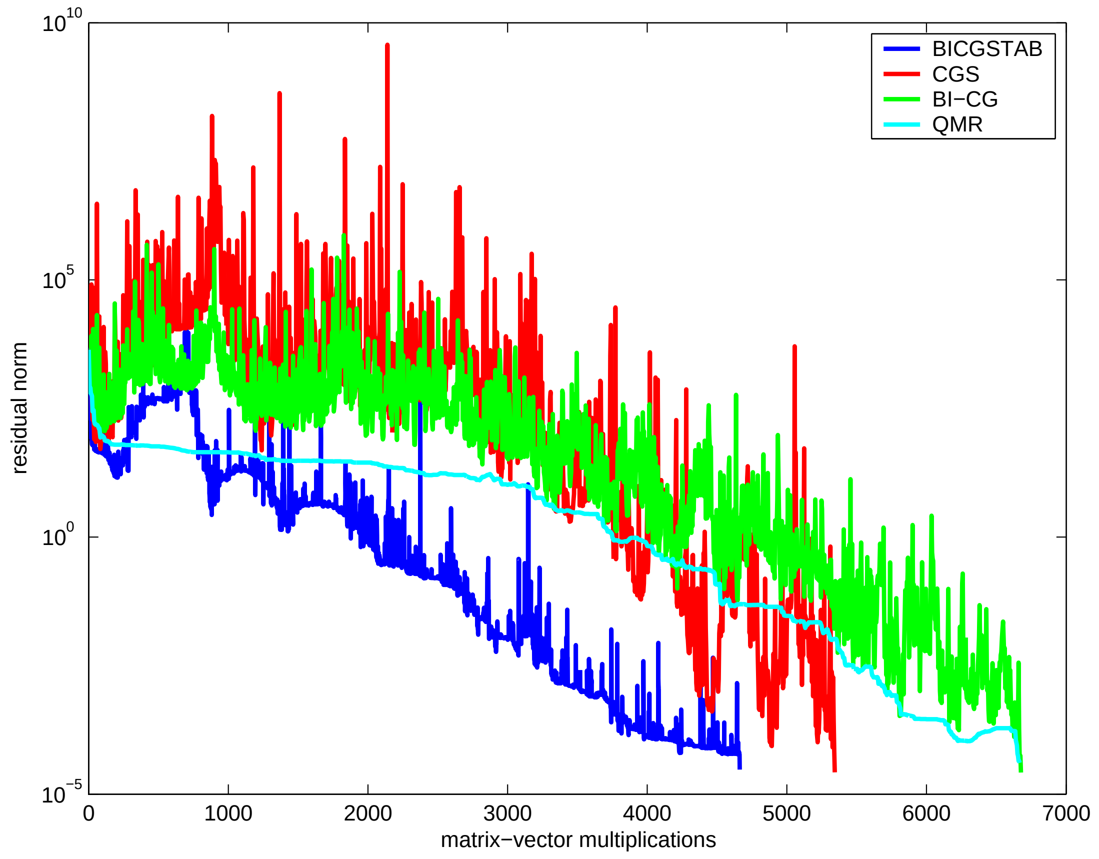
Convergence: first example

The figure below shows typical convergence curves for Bi-CGSTAB, CGS, QMR and Bi-CG.



Convergence: second example

The figure below shows another example of the convergence of Bi-CGSTAB, CGS, QMR and Bi-CG.



BiCGSTAB2 and BiCGstab(ℓ)

For some problems, in particular for real problems with large complex eigenvalues, Bi-CGSTAB does not work well. The reason is that for such problems it is not possible to construct a residual minimising polynomial of the form

$$\psi_k(x) = (1 - \omega_1 x)(1 - \omega_2 x)...(1 - \omega_k x) \ .$$

Such a polynomial has real roots, and can therefore not approximate large eigenvalues.

To overcome this problem Gutknecht proposed BiCGSTAB2, which uses a polynomial with quadratic factors. Sleijpen and Fokkema proposed BiCGstab(ℓ), which uses polynomial factors of degree ℓ.

Intermezzo

Of the Bi-CG methods Bi-CGSTAB is by far the most widely used.

Recently, the new method IDR(s) has been proposed by Sonneveld and van Gijzen.

IDR(1) and Bi-CGSTAB are mathematically equivalent. But convergence of IDR(s) is often much faster than of Bi-CGSTAB for $s > 1$.

Although Bi-CGSTAB and IDR(1) are mathematically equivalent, IDR(s) is based on a completely different approach.

The IDR approach for solving $Ax = b$

Generate residuals $r_n = b - Ax_n$ that are in subspaces $\mathcal{G}_j$ of decreasing dimension.

These nested subspaces are related by

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

where

- $\mathcal{S}$ is a fixed proper subspace of $\mathbb{C}^N$,
- and the $\omega_j \in \mathbb{C}$'s are non-zero scalars.

Ultimately $r_n \in \{\mathbf{0}\}$ (IDR theorem).

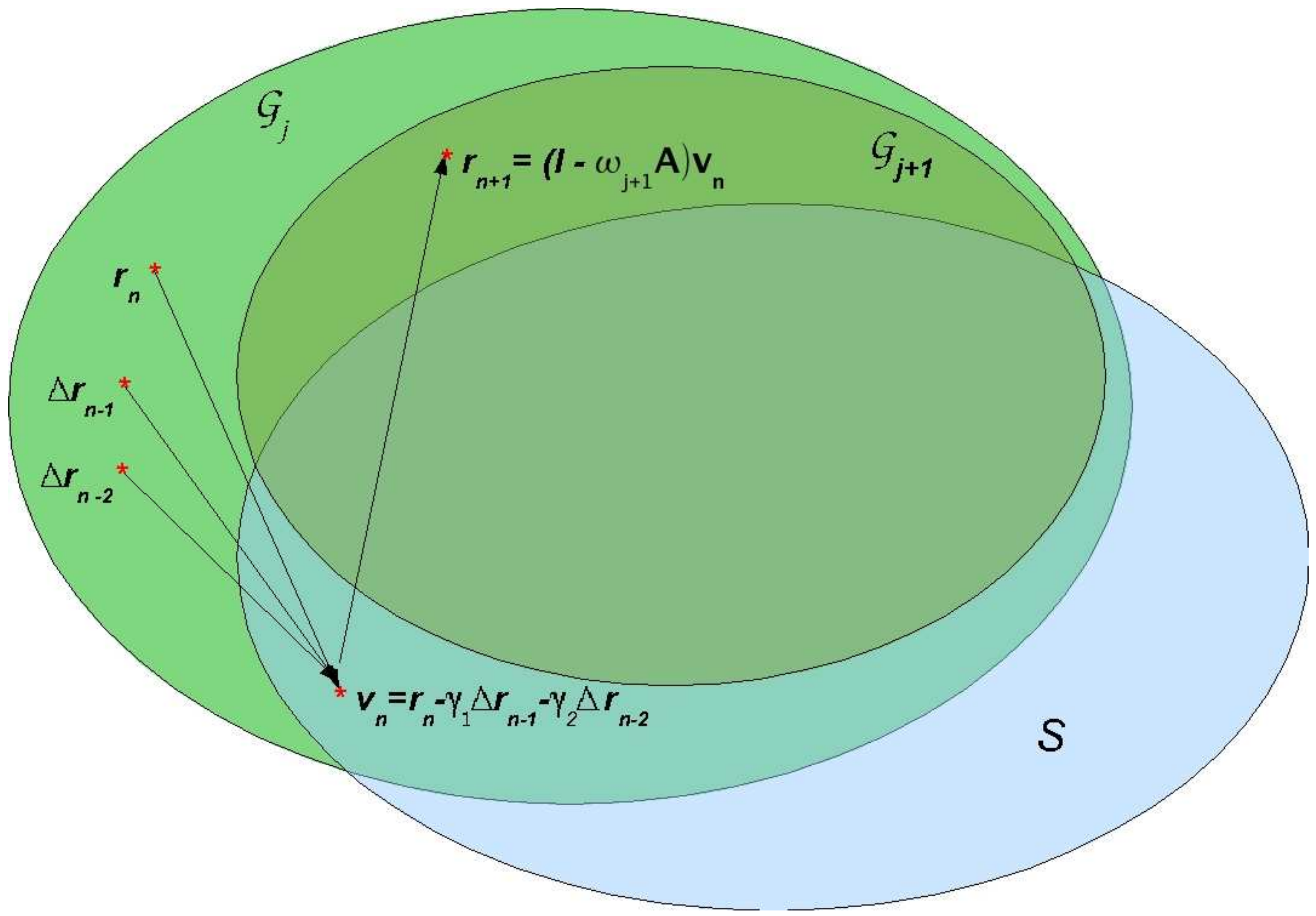
The IDR theorem

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 $K^N(A; v_0)$. 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 $\mathcal{G}_j$, $j = 1, 2, \ldots$ as*

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

where the ω_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



$\mathcal{G}_j$
$\mathcal{G}_{j+1}$
$\mathbf{r}_{n+1} = (\mathbf{I} - \omega_{j+1}\mathbf{A})\mathbf{v}_n$
$\mathbf{r}_n$
$\Delta\mathbf{r}_{n-1}$
$\Delta\mathbf{r}_{n-2}$
$\mathbf{v}_n = \mathbf{r}_n - \gamma_1\Delta\mathbf{r}_{n-1} - \gamma_2\Delta\mathbf{r}_{n-2}$
S

Making an IDR algorithm (2)

Definition of $\mathcal{S}$:

$\mathcal{S}$ can be defined as $span(p_1 \ldots p_s)^{\perp}$. Let P be the matrix with $p_1 \ldots p_s$ as its columns. Then $v \in \mathcal{S} \Leftrightarrow P^H v = \mathbf{0}$.

Residual difference vectors:

We compute residual difference vectors $\Delta r_n = r_{n+1} - r_n = -A\Delta x_n$ to update the solution vector x_{n+1} with the residual r_{n+1}.

Making an IDR algorithm (3)

Intermediate residuals

Intermediate residuals r_n can be generated by repeating the algorithm. Once $s+1$ residuals in $\mathcal{G}_j$ have been computed, the next residual will be in $\mathcal{G}_{j+1}$.

Choice of ω

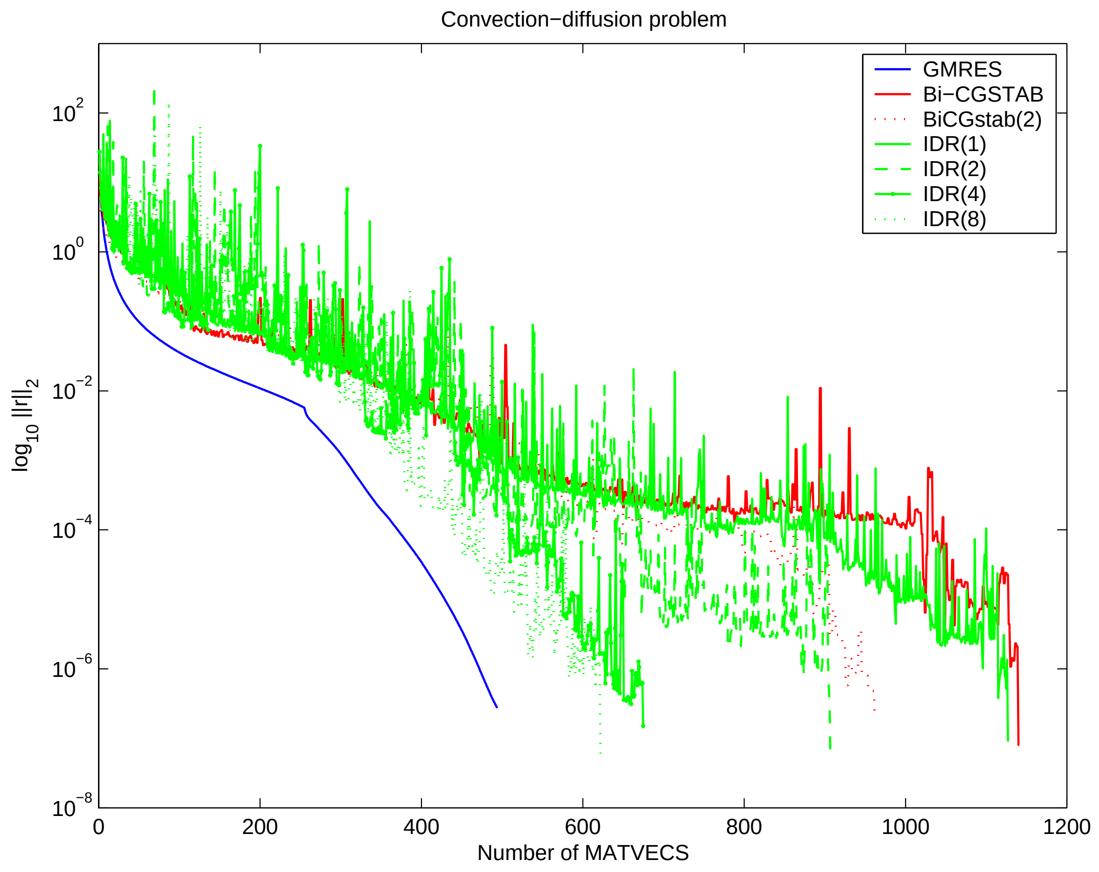
Every $s+1$st step a new ω may be chosen. We choose it such that the next residual is minimized in norm.

Prototype IDR(s) algorithm.

```
while $\|r_n\| > TOL$ or $n < MAXIT$ do
    for $k = 0$ to $s$ do
        Solve $c$ from $P^H dR_n c = P^H r_n$
        $v = r_n - dR_n c$; $t = Av$;
        if $k = 0$ then
            $\omega = (t^H v)/(t^H t)$;
        end if
        $dr_n = -dR_n c - \omega t$; $dx_n = -dX_n c + \omega v$;
        $r_{n+1} = r_n + dr_n$; $x_{n+1} = x_n + dx_n$;
        $n = n + 1$;
        $dR_n = (dr_{n-1} \cdots dr_{n-s})$; $dX_n = (dx_{n-1} \cdots dx_{n-s})$;
    end for
end while
```

Convergence of IDR(s)

The figure below shows typical convergence of Bi-CGSTAB, GMRES (optimal) and IDR(s).



Concluding remarks (1)

Today we discussed short recurrence methods for solving nonsymmetric problems.

Of these methods Bi-CGSTAB is by far the most popular.

A logical question to ask is: are there better methods than Bi-CGSTAB possible that use short recurrences?
The answer is: sure! For example IDR(s). However, neither Bi-CGSTAB nor IDR(s) reduce an error in an optimal way.

Another question is: Is it possible to derive an optimal method for general matrices that uses short recurrences?
The answer is: unfortunately not. This has been proven by Faber and Manteufel in the 90's.

Concluding remarks

When should we use IDR(s) and when GMRES?

It is difficult to give a general answer.

Rules of thumb are:

- Are matrix-vector multiplications (and/or preconditioning operations) expensive?
  GMRES is your best bet, at least if the number of iterations is limited.
- Are matrix-vector multiplications inexpensive and you expect that many iterations are needed (for example because you use a simple preconditioner)?
  Use IDR(s) (with $s = 4$ as a good default).