The IDR(S) method for solving nonsymmetric systems
Application to optimization problems

SIOPT
Martin van Gijzen and Peter Sonneveld
May 12, 2008
Outline

• Introduction: Krylov subspace methods
• The Induced Dimension Reduction Theorem
• The IDR(s) algorithm
• Numerical experiments
• Conclusions
Krylov subspace methods

Krylov subspace methods are iterative solution techniques for solving linear systems

$$Ax = b.$$ 

They construct solutions in the Krylov subspace

$$x_n \in K^n(A, r_0) = \text{span} \left( r_0, Ar_0, A^2r_0, \ldots, A^nr_0 \right),$$

(we assume $$x_0 = 0$$).

For symmetric problems CG and MINRES combine an optimal error reduction with short recurrences.

For nonsymmetric problems this is not possible.
Two approaches

GMRES approach:

• Uses Arnoldi’s method to make basis for Krylov subspace
• Uses long recurrences \(\rightarrow\) computational and memory overhead grows with the iteration number
• Gives optimal reduction of residual norm

Bi-CG approach:

• Uses Bi-Lanczos method to make a basis for the Krylov subspace
• Uses short recurrences \(\rightarrow\) computational and memory overhead constant
• Has no optimality property
The IDR-approach

• The IDR-method that we will present is a short-recurrence method.

• It is based on the IDR-theorem, the idea is to compute successive residuals in shrinking subspaces.
  → ultimately the residual will be in the zero-subspace (and hence be zero).

• The approach is different, but there is a theoretical link with Bi-CG(STAB)
  → in particular: IDR(1) and BiCGSTAB are mathematically equivalent.
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 $G_0$ be the complete Krylov space $K^N(A, v_0)$. Let $S$ denote any (proper) subspace of $\mathbb{C}^N$ such that $S$ and $G_0$ do not share a nontrivial invariant subspace of $A$, and define the sequence $G_j$, $j = 1, 2, \ldots$ as

$$G_j = (I - \omega_j A)(G_{j-1} \cap S)$$

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

(i) $G_j \subset G_{j-1}$ for all $j > 0$.

(ii) $G_j = \{0\}$ for some $j \leq N$. 
Making an IDR algorithm

\[ r_{n+1} = (I - \omega_{j+1} A) v_n \]

\[ v_n = r_n - \gamma_1 \Delta r_{n-1} - \gamma_2 \Delta r_{n-2} \]
Making an IDR algorithm (2)

Definition of $S$:
$S$ 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 $v \in S \iff P^H v = 0$.

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 $G_j$ have been computed, the next residual will be in $G_{j+1}$.

Choice of $\omega$
Every $s + 1$st step a new $\omega$ 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
    $d r_n = -dR_n c - \omega t; d x_n = -dX_n c + \omega v$;
    $r_{n+1} = r_n + d r_n; x_{n+1} = x_n + d x_n$;
    $n = n + 1$;
    $dR_n = (d r_{n-1} \cdots d r_{n-s}); dX_n = (d x_{n-1} \cdots d x_{n-s});$
  end for
end while
Some practical remarks

Choice of $P$
For reasons of robustness we choose for the columns of $P$ a set of orthonormalised random vectors.

Choice of $s$
$s$ should not be chosen too large. In practice $s = 4$ is often a good choice.

Finite termination
The algorithm terminates in at most $N + N/s$ iterations (MATVECS) at the exact solution.
# Vector operations per MATVEC

<table>
<thead>
<tr>
<th>Method</th>
<th>DOT</th>
<th>AXPY</th>
<th>Memory Requirements</th>
</tr>
</thead>
<tbody>
<tr>
<td>IDR(1)</td>
<td>2</td>
<td>4</td>
<td>8</td>
</tr>
<tr>
<td>IDR(4)</td>
<td>$4 \frac{2}{5}$</td>
<td>$9 \frac{7}{10}$</td>
<td>17</td>
</tr>
<tr>
<td>Full GMRES</td>
<td>$\frac{n+1}{2}$</td>
<td>$\frac{n+1}{2}$</td>
<td>$n + 2$</td>
</tr>
<tr>
<td>Bi-CGSTAB</td>
<td>2</td>
<td>3</td>
<td>7</td>
</tr>
<tr>
<td>BiCGstab(2)</td>
<td>$2 \frac{1}{4}$</td>
<td>$3 \frac{3}{4}$</td>
<td>9</td>
</tr>
</tbody>
</table>
Numerical experiments

- A mixed complementarity optimization problem
- A small KKT-system from nonlinear optimization
- A big KKT-system from nonlinear optimization

Computations performed with MATLAB on IBM Thinkpad.
Mixed complementarity optimization

- Problem 'Major basis' from Qaun Li and Michael Ferris
- Included in Tim Davis matrix collection
- Problem size 160,000
- Nonsymmetric
Solution methods

• Matrix strictly diagonally dominant
  → Jacobi preconditioner

• Iterative methods:
  • GMRES (optimal in terms of MATVECS)
  • Bi-CGSTAB (most popular short recurrence method)
  • IDR(1) and IDR(4)

• Termination criterion: $\|r\|/\|b\| < 10^{-6}$. 
Convergence for major basis
# Computing times

<table>
<thead>
<tr>
<th>Method</th>
<th>MATVECS</th>
<th>Time [s]</th>
</tr>
</thead>
<tbody>
<tr>
<td>GMRES</td>
<td>32</td>
<td>15</td>
</tr>
<tr>
<td>Bi-CGSTAB</td>
<td>42</td>
<td>12</td>
</tr>
<tr>
<td>IDR(1)</td>
<td>41</td>
<td>12</td>
</tr>
<tr>
<td>IDR(4)</td>
<td>38</td>
<td>12</td>
</tr>
<tr>
<td>Direct</td>
<td>-</td>
<td>14</td>
</tr>
</tbody>
</table>

Major basis
Some observations

• Bi-CGSTAB and IDR(1) have same convergence (at even steps)

• Convergence of all methods close to (optimal) GMRES → not much to gain.

• Iterative methods slightly faster than MATLAB’s direct solver

• GMRES slightly slower → bigger vector overhead
A KKT problem

- Problem 'c-49' from Olaf Schenk
- Included in Tim Davis matrix collection
- Problem size 21132 equations
- Symmetric
Preconditioning a KKT-system (1)

System matrix $A$ has block structure

$$A = \begin{pmatrix} F & B^T \\ B & -C \end{pmatrix}$$

with $F$ and $C$ pos. def diagonal matrices.

Block $LU$ decomposition:

$$\begin{pmatrix} F & B^T \\ B & -C \end{pmatrix} = \begin{pmatrix} I & O^T \\ BF^{-1} & I \end{pmatrix} \begin{pmatrix} F & B^T \\ O & -M_S \end{pmatrix}$$

with $M_S = BF^{-1}B^T + C$ the Schur complement.
Preconditioning a KKT-system (2)

Idea (e.g. Elman, Silvester, Wathen): take

\[
P = \begin{pmatrix} F & B^T \\ O & -M_S \end{pmatrix}
\]

as (right) preconditioner:

\[
AP^{-1} = \begin{pmatrix} I & O^T \\ BF^{-1} & I \end{pmatrix}
\]

has only eigenvalue 1: GMRES ready in 2 iterations. **BUT**

- Preconditioner nonsymmetric
- Schur complement too expensive to compute
Preconditioning a KKT-system (3)

An SPD block-diagonal preconditioner:

\[ P = \begin{pmatrix} F & O^T \\ O & M_S \end{pmatrix} \]

Can be used with MINRES (short recurrences)
Preconditioned matrix has three distinct eigenvalues → MINRES needs three iterations.

To make a cheap approximation to the Schur complement we take

\[ \tilde{M}_S = \text{diag}(M_S) \]
Solution methods

Block-diagonal preconditioner:

• MINRES (Matlab)
• Bi-CGSTAB
• IDR(1) and IDR(4)

Block-upper triangular preconditioner:

• Bi-CGSTAB and BiCGstab(2)
• IDR(1) and IDR(4)

Tolerance: \( \| b - Ax_n \| < 10^{-6} \| b \| \)
Block-diagonal preconditioner
## Computing times

<table>
<thead>
<tr>
<th>Method</th>
<th>MATVECS</th>
<th>Time [s]</th>
</tr>
</thead>
<tbody>
<tr>
<td>MINRES</td>
<td>859</td>
<td>33</td>
</tr>
<tr>
<td>Bi-CGSTAB</td>
<td>2712</td>
<td>73</td>
</tr>
<tr>
<td>IDR(1)</td>
<td>2664</td>
<td>76</td>
</tr>
<tr>
<td>IDR(4)</td>
<td>963</td>
<td>30</td>
</tr>
<tr>
<td>Direct</td>
<td>-</td>
<td>22</td>
</tr>
</tbody>
</table>

*Block-diagonal preconditioner*
Block-upper triangular preconditioner

![Graph showing convergence rates for different methods]

- Bi-CGSTAB
- BiCGstab(2)
- IDR(1)
- IDR(4)

| Number of MATVECS | $|r|/|b|$ |
|-------------------|--------|
| Bi-CGSTAB         | 10^{-7} |
| BiCGstab(2)       | 10^{-6} |
| IDR(1)            | 10^{-5} |
| IDR(4)            | 10^{-4} |
## Computing times

<table>
<thead>
<tr>
<th>Method</th>
<th>MATVECS</th>
<th>Time [s]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bi-CGSTAB</td>
<td>1078</td>
<td>32</td>
</tr>
<tr>
<td>BiCGStab(2)</td>
<td>828</td>
<td>25</td>
</tr>
<tr>
<td>IDR(1)</td>
<td>886</td>
<td>27</td>
</tr>
<tr>
<td>IDR(4)</td>
<td>467</td>
<td>16</td>
</tr>
<tr>
<td>Direct</td>
<td>-</td>
<td>22</td>
</tr>
</tbody>
</table>

Block-upper triangular preconditioner
Some observations

- Without preconditioning none of the methods converge.
- IDR(4) with block-diagonal preconditioner closely follows optimal MINRES convergence.
- IDR(4) with block-upper triangular preconditioner is clearly the fastest method.
- IDR(4) with block-upper triangular preconditioner also faster than Matlab’s direct solver.
A BIG KKT problem

- Problem ’c-big’ from Olaf Schenk
- Included in Tim Davis matrix collection
- Problem size 345241 equations
- Symmetric
Block-upper triangular preconditioner

![Graph showing the convergence of Bi-CGSTAB, BiCGstab(2), IDR(1), and IDR(4) methods.

- **X-axis:** Number of MATVECS
- **Y-axis:** $|r|/|b|$

The graph compares the convergence of different methods, illustrating how they perform over the number of MATVECS.
## Computing times

<table>
<thead>
<tr>
<th>Method</th>
<th>MATVECS</th>
<th>Time [s]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bi-CGSTAB</td>
<td>2902</td>
<td>1664</td>
</tr>
<tr>
<td>BiCGStab(2)</td>
<td>2296</td>
<td>1407</td>
</tr>
<tr>
<td>IDR(1)</td>
<td>3413</td>
<td>2015</td>
</tr>
<tr>
<td>IDR(4)</td>
<td>1710</td>
<td>1077</td>
</tr>
<tr>
<td>Direct</td>
<td>-</td>
<td>Out of memory</td>
</tr>
</tbody>
</table>

Block-upper triangular preconditioner
Conclusions

• IDR($s$) is quite promising and outperforms Bi-CGSTAB for relevant (nonsymmetric) optimisation problems.

• Even for symmetric KKT-problems, IDR($s$) may be competitive with a method like MINRES if a nonsymmetric preconditioner is used.

More information:
http://ta.twi.tudelft.nl/nw/users/gijzen/software.html

- Report: IDR($s$): a family of simple and fast algorithms for solving large nonsymmetric linear systems, submitted

- Matlab code
Acknowledgement

Part of this research has been funded by the Dutch BSIK/BRICKS project.