Solving a large dense linear system by adaptive cross approximation

Katrijn Frederix, Marc Van Barel

Department of Computer Science
K.U.Leuven

September 2007
Outline

Introduction
  Motivation
  Goal

Low rank approximation
  Cross approximation
  Adaptive cross approximation

Solving linear system
  Unitary-weight representation
  Solving linear system with UWR

Numerical results
  Problem formulation
  Results of numerical experiments

Conclusions
Outline

Introduction
  Motivation
  Goal

Low rank approximation

Solving linear system

Numerical results

Conclusions
Solving integral equations results in linear system $Ax = b$. 
Motivation of this research

Solving integral equations results in linear system $Ax = b$.

- Time-consuming evaluation of integrals.
Motivation of this research

Solving integral equations results in linear system $Ax = b$.

- Time-consuming evaluation of integrals.
- Large, dense, full rank and no explicit structure.
Motivation of this research

Solving integral equations results in linear system $Ax = b$.

- Time-consuming evaluation of integrals.
- Large, dense, full rank and no explicit structure.
- Approximate by a rank structured matrix.
Goal of this research

Solve efficiently the linear system without computing all matrix entries of $A$ by use of the adaptive cross approximation.
Outline

Introduction

Low rank approximation
  Cross approximation
  Adaptive cross approximation

Solving linear system

Numerical results

Conclusions
Motivation of cross approximation

- Best low rank approximation: SVD
  - Cannot lead to fast algorithms.
Motivation of cross approximation

- Best low rank approximation: SVD
  - Cannot lead to fast algorithms.
- Cross or skeleton approximation.
  [Goreinov, Tyrtshnikov, Zamarashkin 1997]
  - Less computational effort and uses few entries from the original matrix.
Idea of cross approximation

- Given a matrix $M, R \in \mathbb{C}^{m \times n}$ with $\|M - R\| \leq \tau$ and $\text{rank}(R) \leq r$. 

Katrijn Frederix, Marc Van Barel

Solving system with ACA
Idea of cross approximation

- Given a matrix $M, R \in \mathbb{C}^{m \times n}$ with $\|M - R\| \leq \tau$ and $\text{rank}(R) \leq r$.
- Choose $\hat{m} \subset I = \{1, \ldots, m\}$ and $\hat{n} \subset J = \{1, \ldots, n\}$. 
Idea of cross approximation

- Given a matrix $M, R \in \mathbb{C}^{m \times n}$ with $\|M - R\| \leq \tau$ and $\text{rank}(R) \leq r$.
- Choose $\hat{m} \subset I = \{1, \ldots, m\}$ and $\hat{n} \subset J = \{1, \ldots, n\}$.
- Construct $\tilde{M} = M|_{\hat{m} \times \hat{n}} \cdot S \cdot M|_{\hat{m} \times \hat{n}} \in \text{Rk}(\min\{\#\hat{n}, \#\hat{m}\})$ with $S = (M|_{\hat{m} \times \hat{n}})^{-1}, (M|_{\hat{m} \times \hat{n}})^{-1}$ a submatrix of $M$ of maximal volume (i.e. determinant in modulus).

[Goreinov, Zamarashkin, Tyrtyshnikov 1997]
Idea of cross approximation

- Given a matrix $M, R \in \mathbb{C}^{m \times n}$ with $\|M - R\| \leq \tau$ and $\text{rank}(R) \leq r$.
- Choose $\hat{m} \subset I = \{1, \ldots, m\}$ and $\hat{n} \subset J = \{1, \ldots, n\}$.
- Construct $\tilde{M} = M|_{m \times \hat{n}} \cdot S \cdot M|_{\hat{m} \times n} \in \text{Rk}(\min\{\#\hat{n}, \#\hat{m}\})$ with $S = (M|_{\hat{m} \times \hat{n}})^{-1}, (M|_{\hat{m} \times \hat{n}})^{-1}$ a submatrix of $M$ of maximal volume (i.e. determinant in modulus).

[Goreinov, Zamarashkin, Tyrtysnikov 1997]

- Result: $\|M - \tilde{M}\| \leq \tau(1 + 2\sqrt{r}(\sqrt{m} + \sqrt{n}))$. 
Figure: The matrix $M$ is approximated by a combination of few rows $\hat{m} = \{2, 4, 8\}$ and columns $\hat{n} = \{2, 5, 7\}$ of the matrix.
Determination of cross approximation

Compute successively rank one approximations or skeletons.
Determination of cross approximation

Compute successively rank one approximations or skeletons.

- Determine pivot index pair \((i^*, j^*)\) with maximal entry in modulus \(|M_{i^*, j^*}|\).
Determination of cross approximation

Compute successively rank one approximations or skeletons.

- Determine pivot index pair \((i^*, j^*)\) with maximal entry in modulus \(|M_{i^*, j^*}|\).
- Set \(\delta = M_{i^*, j^*}\).
Determination of cross approximation

Compute successively rank one approximations or skeletons.

- Determine pivot index pair \((i^*, j^*)\) with maximal entry in modulus \(|M_{i^*, j^*}|\).
- Set \(\delta = M_{i^*, j^*}\).
- Compute entries \(a_i = M_{i, j^*} / \delta \ (i \in I)\) and \(b_j = M_{i^*, j} \ (j \in J)\).
Determination of cross approximation

Compute successively rank one approximations or skeletons.

- Determine pivot index pair \((i^*, j^*)\) with maximal entry in modulus \(|M_{i^*, j^*}|\).
- Set \(\delta = M_{i^*, j^*}\).
- Compute entries \(a_i = M_{i, j^*} / \delta\) \((i \in I)\) and \(b_j = M_{i^*, j}\) \((j \in J)\).

Apply same to remainder \(M - \sum_{k=1}^{l} a^k b^k\).
Determination of cross approximation

Compute successively rank one approximations or skeletons.

- Determine pivot index pair \((i^*, j^*)\) with maximal entry in modulus \(|M_{i^*, j^*}|\).
- Set \(\delta = M_{i^*, j^*}\).
- Compute entries \(a_i = M_{i, j^*}/\delta \ (i \in I)\) and \(b_j = M_{i^*, j} \ (j \in J)\).

Apply same to remainder \(M - \sum_{k=1}^{l} a^k b^k\).

Approximation \(\tilde{M} = \sum_{k=1}^{p} a^k b^k\).
Example of CA with exact rank 3

First skeleton: Determine maximum in modulus element of matrix.

\[
\begin{pmatrix}
6 & 4 & 5 & 4 & 3 & 5 & 3 & 4 \\
8 & 6 & 7 & 5 & 4 & 7 & 4 & 5 \\
10 & 7 & 8 & 7 & 5 & 9 & 5 & 6 \\
6 & 4 & 5 & 4 & 3 & 5 & 3 & 4 \\
8 & 5 & 6 & 6 & 4 & 7 & 4 & 5 \\
10 & 7 & 9 & 6 & 5 & 8 & 5 & 7 \\
6 & 4 & 5 & 4 & 3 & 5 & 3 & 4 \\
10 & 7 & 8 & 7 & 5 & 9 & 5 & 6 \\
\end{pmatrix}
\]

- \( i_1^* = 3 \)
- \( j_1^* = 1 \)
Example of CA with exact rank 3

First skeleton: Compute row $b^1$ and column $a^1$.

\[
\begin{pmatrix}
6 & 4 & 5 & 4 & 3 & 5 & 3 & 4 \\
8 & 6 & 7 & 5 & 4 & 7 & 4 & 5 \\
10 & 7 & 8 & 7 & 5 & 9 & 5 & 6 \\
6 & 4 & 5 & 4 & 3 & 5 & 3 & 4 \\
8 & 5 & 6 & 6 & 4 & 7 & 4 & 5 \\
10 & 7 & 9 & 6 & 5 & 8 & 5 & 7 \\
6 & 4 & 5 & 4 & 3 & 5 & 3 & 4 \\
10 & 7 & 8 & 7 & 5 & 9 & 5 & 6 \\
\end{pmatrix}
\]

- $i_1^* = 3$
- $j_1^* = 1$
- $\delta = 10$
- $\tilde{a}^1 / \delta$
- $b^1$
Example of CA with exact rank 3

Second skeleton: Subtraction of first skeleton.

\[
\begin{pmatrix}
0 & -\frac{1}{5} & \frac{1}{5} & -\frac{1}{5} & 0 & -\frac{2}{5} & 0 & \frac{2}{5} \\
0 & \frac{2}{5} & \frac{3}{5} & -\frac{3}{5} & 0 & -\frac{1}{5} & 0 & \frac{1}{5} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & -\frac{1}{5} & \frac{1}{5} & -\frac{1}{5} & 0 & -\frac{2}{5} & 0 & \frac{2}{5} \\
0 & -\frac{3}{5} & -\frac{2}{5} & -\frac{1}{5} & 0 & -\frac{1}{5} & 0 & \frac{1}{5} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & -\frac{1}{5} & \frac{1}{5} & -\frac{1}{5} & 0 & -\frac{2}{5} & 0 & \frac{2}{5} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{pmatrix}
\]

\[i^*_2 = 2\]

\[j^*_2 = 3\]
Example of CA with exact rank 3

Second skeleton: Compute row $b^2$ and column $a^2$.

$$
\begin{pmatrix}
0 & \frac{-1}{5} & \frac{1}{5} & \frac{-1}{5} & 0 & \frac{-2}{5} & 0 & \frac{2}{5} \\
0 & \frac{2}{5} & \frac{3}{5} & \frac{-3}{5} & 0 & \frac{-2}{5} & 0 & \frac{2}{10} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & \frac{-1}{5} & \frac{1}{5} & \frac{-1}{5} & 0 & \frac{-2}{5} & 0 & \frac{2}{5} \\
0 & \frac{-3}{5} & \frac{-2}{5} & \frac{-1}{5} & 0 & \frac{-1}{5} & 0 & \frac{1}{5} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & \frac{-1}{5} & \frac{1}{5} & \frac{-1}{5} & 0 & \frac{-2}{5} & 0 & \frac{2}{5} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{pmatrix}
$$

- $i_2^* = 2$
- $j_2^* = 3$
- $\delta = 6/10$
- $\tilde{a}^2 / \delta$
- $b^2$
Example of CA with exact rank 3

Third skeleton: Subtraction of second skeleton.

\[
\begin{pmatrix}
0 & -\frac{1}{3} & 0 & 0 & 0 & -\frac{1}{3} & 0 & \frac{1}{3} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & -\frac{1}{3} & 0 & 0 & 0 & -\frac{1}{3} & 0 & \frac{1}{3} \\
0 & -\frac{1}{3} & 0 & 0 & 0 & -\frac{1}{3} & 0 & \frac{1}{3} \\
0 & -\frac{2}{3} & 0 & 0 & 0 & -\frac{2}{3} & 0 & \frac{1}{3} \\
0 & -\frac{1}{3} & 0 & 0 & 0 & -\frac{1}{3} & 0 & \frac{1}{3} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{pmatrix}
\]

\[i_2^* = 6\]
\[j_2^* = 2\]
Example of CA with exact rank 3

Third skeleton: Compute row $b^3$ and column $a^3$.

$$
\begin{pmatrix}
0 & -\frac{1}{3} & 0 & 0 & 0 & -\frac{1}{3} & 0 & \frac{1}{3} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & -\frac{1}{3} & 0 & 0 & 0 & -\frac{1}{3} & 0 & \frac{1}{3} \\
0 & -\frac{1}{3} & 0 & 0 & 0 & -\frac{1}{3} & 0 & \frac{1}{3} \\
0 & -\frac{2}{3} & 0 & 0 & 0 & -\frac{2}{3} & 0 & \frac{2}{3} \\
0 & -\frac{1}{3} & 0 & 0 & 0 & -\frac{1}{3} & 0 & \frac{1}{3} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{pmatrix}
$$

- $i_3^* = 6$
- $j_3^* = 2$
- $\delta = 2/3$
- $\tilde{a}^3 / \delta$
- $b^3$
Example of CA with exact rank 3

Third skeleton: Subtraction of third skeleton results in zero matrix.
Example of CA with exact rank 3

Third skeleton: Subtraction of third skeleton results in zero matrix.

> Low rank approximation is found $\tilde{M} = \sum_{k=1}^{3} a^k b^k$. 
Determination of cross approximation

Disadvantages of method:

- All elements of $M$ have to be known.
Determination of cross approximation

Disadvantages of method:
- All elements of $M$ have to be known.
- Matrix $M$ has to be updated.
Determination of cross approximation

Disadvantages of method:

- All elements of $M$ have to be known.
- Matrix $M$ has to be updated.
- Rank of $M$ has to be known in advance.
Adaptive cross approximation (ACA)

Other approach:
- Small part of rows and columns are considered.
Adaptive cross approximation (ACA)

Other approach:

- Small part of rows and columns are considered.
- Only necessary rows and columns are updated.
Adaptive cross approximation (ACA)

Other approach:

- Small part of rows and columns are considered.
- Only necessary rows and columns are updated.
- Rank is determined according to a stopping criterion.
Adaptive cross approximation (ACA)

Other approach:

- Small part of rows and columns are considered.
- Only necessary rows and columns are updated.
- Rank is determined according to a stopping criterion.

Remark: Matrix $S$ is not of maximal volume.
Example of ACA with exact rank 3

First skeleton:

- Pivot row index (arbitrary): $i^*_1 = 1$. 
Example of ACA with exact rank 3

First skeleton:

- Pivot row index (arbitrary): $i_1^* = 1$.
- Compute row: $b^1 = M(1,:) = \begin{pmatrix} 6 & 4 & 5 & 4 & 3 & 5 & 3 & 4 \end{pmatrix}$. 
Example of ACA with exact rank 3

First skeleton:

- Pivot row index (arbitrary): $i_1^* = 1$.
- Compute row: $b^1 = M(1,:) = (6 \ \ 4 \ \ 5 \ \ 4 \ \ 3 \ \ 5 \ \ 3 \ \ 4)$.  
- Determine maximum of $b^1$: $j_1^* = 1$ and set $\delta = 6$. 

Katrijn Frederix, Marc Van Barel

Solving system with ACA
Example of ACA with exact rank 3

First skeleton:

- Pivot row index (arbitrary): \( i^*_1 = 1 \).
- Compute row: \( b^1 = M(1, :) = (6 \ 4 \ 5 \ 4 \ 3 \ 5 \ 3 \ 4) \).
- Determine maximum of \( b^1 \): \( j^*_1 = 1 \) and set \( \delta = 6 \).
- Compute column:
  \[
  a^1 = M(:, 1)/\delta = (1 \ 4/3 \ 5/3 \ 1 \ 4/3 \ 5/3 \ 1 \ 5/3)^T.
  \]
Example of ACA with exact rank 3

First skeleton:

- **Pivot row index (arbitrary):** $i_1^* = 1$.
- **Compute row:** $b^1 = M(1,:) = \begin{pmatrix} 6 & 4 & 5 & 4 & 3 & 5 & 3 & 4 \end{pmatrix}$.
- **Determine maximum of $b^1$:** $j_1^* = 1$ and set $\delta = 6$.
- **Compute column:**
  
  $$a^1 = M(:,1)/\delta = \begin{pmatrix} 1 & 4/3 & 5/3 & 1 & 4/3 & 5/3 & 1 & 5/3 \end{pmatrix}^T$$

- **Stopping criterion:** arbitrary set of matrix entries (outside skeleton). Subtract skeleton and compare with original. Stopping criterion not fulfilled.
Example of ACA with exact rank 3

Second skeleton:

- Determine maximum of $a^1: i_2^* = 3$. 

Katrijn Frederix, Marc Van Barel

Solving system with ACA
Example of ACA with exact rank 3

Second skeleton:

- Determine maximum of $a^1: i_2^* = 3$.
- Compute row:

$$b^2 = M(3,:) - a^1_3 b^1 = \begin{pmatrix} 0 & \frac{1}{3} & -\frac{1}{3} & \frac{1}{3} & 0 & \frac{2}{3} & 0 & -\frac{2}{3} \end{pmatrix}.$$
Example of ACA with exact rank 3

Second skeleton:

- Determine maximum of $a^1$: $i_2^* = 3$.
- Compute row:
  
  $b^2 = M(3, :) - a_3^1 b^1 = \begin{pmatrix} 0 & \frac{1}{3} & -\frac{1}{3} & \frac{1}{3} & 0 & \frac{2}{3} & 0 & -\frac{2}{3} \end{pmatrix}$.
- Determine maximum of $b^2$: $j_2^* = 6$ and set $\delta = \frac{2}{3}$.
Example of ACA with exact rank 3

Second skeleton:

- Determine maximum of $a^1$: $i_2^* = 3$.
- Compute row:
  \[ b^2 = M(3,:) - a^1 b^1 = \begin{pmatrix} 0 & \frac{1}{3} & -\frac{1}{3} & \frac{1}{3} & 0 & \frac{2}{3} & 0 & -\frac{2}{3} \end{pmatrix}. \]
- Determine maximum of $b^2$: $j^*_2 = 6$ and set $\delta = \frac{2}{3}$.
- Compute column:
  \[ a^2 = (M(:,6) - a^1 b^1_6) / \delta = \begin{pmatrix} 0 & \frac{1}{2} & 1 & 0 & \frac{1}{2} & -\frac{1}{2} & 0 & 1 \end{pmatrix}^T. \]
Example of ACA with exact rank 3

Second skeleton:

► Determine maximum of $a^1$: $i_2^* = 3$.

► Compute row:

$$b^2 = M(3,:)-a_3^1 b^1 = \begin{pmatrix} 0 & \frac{1}{3} & -\frac{1}{3} & \frac{1}{3} & 0 & \frac{2}{3} & 0 & -\frac{2}{3} \end{pmatrix}.$$  

► Determine maximum of $b^2$: $j_2^* = 6$ and set $\delta = \frac{2}{3}$.

► Compute column:

$$a^2 = (M(:,6)-a^1 b_6^1)/\delta = \begin{pmatrix} 0 & \frac{1}{2} & 1 & 0 & \frac{1}{2} & -\frac{1}{2} & 0 & 1 \end{pmatrix}^T.$$  

► Stopping criterion: Update previous check points and compare with original. Stopping criterion not fulfilled.
Example of ACA with exact rank 3

Third skeleton:

- Determine maximum of $a^2$: $i_3^* = 8$. 

Katrijn Frederix, Marc Van Barel

Solving system with ACA
Example of ACA with exact rank 3

Third skeleton:
- Determine maximum of $a^2$: $i^*_3 = 8$.
- Compute row:

$$b^3 = M(8,:)-a_8^1 b^1 - a_8^2 b^2 = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \end{pmatrix}.$$
Example of ACA with exact rank 3

Third skeleton:

- Determine maximum of $a^2$: $i^*_3 = 8$.
- Compute row: $b^3 = M(8,:) - a^1_8 b^1 - a^2_8 b^2$
- $\delta = 0$: choose other row index.
Example of ACA with exact rank 3

Third skeleton:

- Determine maximum of $a^2$: $i_3^* = 8$.
- Compute row: $b^3 = M(8,:) - a_8^1b^1 - a_8^2b^2$
- $\delta = 0$: choose other row index.
- Row index: $i_3^* = 2$. 
Example of ACA with exact rank 3

Third skeleton:

- Determine maximum of $a^2$: $i_3^* = 8$.
- Compute row: $b^3 = M(8,:) - a_8^1b^1 - a_8^2b^2$
- $\delta = 0$: choose other row index.
- Row index: $i_3^* = 2$.
- Compute row:
  \[
  b^3 = M(2,:) - a_2^1b^1 - a_2^2b^2 = \begin{pmatrix}
  0 & \frac{1}{2} & \frac{1}{2} & -\frac{1}{2} & 0 & 0 & 0 & 0
  \end{pmatrix}.
  \]
Example of ACA with exact rank 3

Third skeleton:

- Determine maximum of $a^2$: $i_3^* = 8$.
- Compute row: $b^3 = M(8, :) - a_8^1 b_1 - a_8^2 b_2$
- $\delta = 0$: choose other row index.
- Row index: $i_3^* = 2$.
- Compute row: $b^3 = M(2, :) - a_2^1 b_1 - a_2^2 b_2$
- Determine maximum of $b^3$: $j_2^* = 2$ and set $\delta = \frac{1}{2}$.
Example of ACA with exact rank 3

Third skeleton:

- Determine maximum of $a^2$: $i^*_3 = 8$.
- Compute row: $b^3 = M(8, :) - a^1_8 b^1 - a^2_8 b^2$
- $\delta = 0$: choose other row index.
- Row index: $i^*_3 = 2$.
- Compute row: $b^3 = M(2, :) - a^1_2 b^1 - a^2_2 b^2$
- Determine maximum of $b^3$: $j^*_2 = 2$ and set $\delta = \frac{1}{2}$.
- Compute column: $a^3 = (M(:, 2) - a^1 b^1_2 - a^2 b^2_2)/\delta$
  $$= \begin{pmatrix} 0 & 1 & 0 & 0 & -1 & 1 & 0 & 0 \end{pmatrix}^T.$$
Example of ACA with exact rank 3

Third skeleton:

- Determine maximum of $a^2$: $i^*_3 = 8$.
- Compute row: $b^3 = M(8, :) - a^1_8b^1 - a^2_8b^2$
- $\delta = 0$: choose other row index.
- Row index: $i^*_3 = 2$.
- Compute row: $b^3 = M(2, :) - a^1_2b^1 - a^2_2b^2$
- Determine maximum of $b^3$: $j^*_2 = 2$ and set $\delta = \frac{1}{2}$.
- Compute column: $a^3 = (M(:, 2) - a^1b^1 - a^2b^2) / \delta$
- Stopping criterion: Update previous check points and compare with original. Stopping criterion fulfilled.
Stopping criterion of ACA

- Arbitrary sets of $t$ matrix entries $R$, outside skeletons.
Stopping criterion of ACA

- Arbitrary sets of $t$ matrix entries $R$, outside skeletons.
- Entries are updated every iteration $R_{i_l,j_l}^P = R_{i_l,j_l}^P - d_{i_l}^P b_{j_l}^P$, $l = 1, \ldots, t$. 

Katrijn Frederix, Marc Van Barel
Stopping criterion of ACA

- Arbitrary sets of $t$ matrix entries $R$, outside skeletons.
- Entries are updated every iteration $R^P_{i_l,j_l} = R^P_{i_l,j_l} - a^P_{i_l}b^P_{j_l}$, $l = 1, \ldots, t$.
- If for all matrix entries it holds that
  \[ \frac{|M_{i_l,j_l} - R^P_{i_l,j_l}|}{\max |M_{i,j}|} \leq \tau \]
  with $l = 1, \ldots, t$ and $\tau$ a given accuracy, the algorithm stops.
Choice of the number of matrix entries used in stopping criterion

Matrix entries can be divided in three disjunct subsets:

- Coming from skeleton.
- Fulfill stopping condition.
- Do not fulfill stopping condition.
Choice of the number of matrix entries used in stopping criterion

Matrix entries can be divided in three disjunct subsets:

- Coming from skeleton.
- Fulfill stopping condition.
- Do not fulfill stopping condition.

The probability that all the matrix entries fulfill the condition is given by

\[ P_t \approx p^t \]

with \( p \) the probability that a matrix entry is in the second set.
Choice of the number of matrix entries used in stopping criterion

\[ n = 100 \]

- Number of entries considerably large but not too large.
Choice of the number of matrix entries used in stopping criterion

\[ \frac{\| M - \tilde{M} \|}{\| M \|}, \ n = 200 \]
Choice of the number of matrix entries used in stopping criterion

Results:
- No increase of accuracy of approximation
- No increase of number of iterations
Choice of the number of matrix entries used in stopping criterion

Results:

- No increase of accuracy of approximation
- No increase of number of iterations

Value in numerical experiments: One percent of total matrix entries is used for stopping criterion.
Outline

Introduction

Low rank approximation

Solving linear system
  Unitary-weight representation
  Solving linear system with UWR

Numerical results

Conclusions
Unitary-weight representation (UWR)

[Delvaux, Van Barel 2006]

- Rank structured matrix: every block beginning in the bottom left corner is of low rank.
Unitary-weight representation (UWR)

[Delvaux, Van Barel 2006]

- Rank structured matrix: every block beginning in the bottom left corner is of low rank.
- Compact representation of the rank structure.
Unitary-weight representation

- Rank structure with three blocks.
Unitary-weight representation

- Unitary transformation on bottom block.
- Created zeros in two bottom rows.
- Top row contains compressed information about whole block.
- Block of weights is stored.

Rk 1

Rk 2

0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0

Katrijn Frederix, Marc Van Barel
Solving system with ACA
Unitary-weight representation

- Unitary transformation on second block.
- Created zeros in two bottom rows.
- Top rows contain compressed information about whole block.
- Block of weights is stored.

$$
\begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
\end{bmatrix}
$$
Unitary-weight representation

- Schematic picture of the unitary-weight representation for this rank structure.
- Three unitary transformations.
- Three weight blocks.
Unitary-weight representation with ACA

Rank structure with three blocks.
Unitary-weight representation with ACA

Consider bottom block.
Unitary-weight representation with ACA

Adaptive cross approximation on bottom block.
Unitary-weight representation with ACA

QR-factorization of $A_3$.
Unitary-weight representation with ACA

Save weight block and $Q_3$. 

$Q_3 \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}$
Unitary-weight representation with ACA

Consider second block.
Unitary-weight representation with ACA

Adaptive cross approximation on second block.
Consider blocks: $\bar{A}_2 = \begin{bmatrix} A_2 & 0 \\ 0 & R_3 \end{bmatrix}$ and $\bar{B}_2 = \begin{bmatrix} B_2 \\ B_3 |_{1,\ldots,j_2} \end{bmatrix}$.
Unitary-weight representation with ACA

- Consider blocks: $\bar{A}_2 = \begin{bmatrix} A_2 & \mathbf{0} \\ \mathbf{0} & R_3 \end{bmatrix}$ and $\bar{B}_2 = \begin{bmatrix} B_2 \\ B_3 |_{1, \ldots, j_2} \end{bmatrix}$.

- Apply truncated SVD to $\bar{B}_2 = U_2 \Sigma_2 V_2^*$. 
Unitary-weight representation with ACA

- Consider blocks: $\bar{A}_2 = \begin{bmatrix} A_2 & 0 \\ 0 & R_3 \end{bmatrix}$ and $\bar{B}_2 = \begin{bmatrix} B_2 \\ B_3 |_{1, \ldots, j_2} \end{bmatrix}$.

- Apply truncated SVD to $\bar{B}_2 = U_2 \Sigma_2 V_2^*$. 

- Apply $QR$-factorization to $\bar{A}_2 U_2 \Sigma_2$. 

Consider blocks: \( \bar{A}_2 = \begin{bmatrix} A_2 & 0 \\ 0 & R_3 \end{bmatrix} \) and \( \bar{B}_2 = \begin{bmatrix} B_2 \\ B_3 |_{1, \ldots, j_2} \end{bmatrix} \).

- Apply truncated SVD to \( \bar{B}_2 = U_2 \Sigma_2 V_2^* \).
- Apply QR-factorization to \( \bar{A}_2 U_2 \Sigma_2 \).
- Save weight matrix \( R_2 V_2^* \).
Unitary-weight representation with ACA

- Compress rows $B_2$ and $B_3|1, \ldots, j_2$ with SVD.
- $QR$-factorization of column block.
Unitary-weight representation with ACA

- Save weight block and $Q_2$.
- Consider first block.
Unitary-weight representation with ACA

Adaptive cross approximation on first block.
Compress rows $B_1$ and $B_2|1, \ldots, j_1$ with SVD.

$QR$-factorization of column block.

Save weight block and $Q_1$. 

Katrijn Frederix, Marc Van Barel
Solving linear system with UWR

[Delvaux, Van Barel 2006]

Compute QR-factorization of a rank structured matrix using unitary-weight representation.
Solving linear system with UWR

[Delvaux, Van Barel 2006]

- Compute QR-factorization of a rank structured matrix using unitary-weight representation.
- QR-factorization is used as a preprocessing step for solving the linear system.
Computation of QR-factorization of rank structured matrix using UWR

- Rank structured matrix with three blocks.

Katrijn Frederix, Marc Van Barel

Solving system with ACA
Computation of QR-factorization of rank structured matrix using UWR

Preparative phase:
- Apply precomputed unitary transformations.
Computation of QR-factorization of rank structured matrix using UWR

Preparative phase:
- No mixture real size and weight elements.
- Regression of column representation.
Computation of QR-factorization of rank structured matrix using UWR

Preparative phase:
- All precomputed unitary transformations are applied.
Computation of QR-factorization of rank structured matrix using UWR

Residual phase:

- No mixture real size and weight elements.
- Spreading out of column representation.
Computation of QR-factorization of rank structured matrix using UWR

Residual phase:
- Make first block column upper triangular.
- Apply same unitary transformation to other part.
Computation of QR-factorization of rank structured matrix using UWR

Residual phase:
- Spreading out column representation.
- Make second block upper triangular.
Computation of QR-factorization of rank structured matrix using UWR

Residual phase:
- Unitary transformations at left: $Q$-factor.
- Upper triangular matrix together with UWR: $R$-factor.
QR-factorization as preprocessing step

System

\[ Ax = b \]

is written as

\[ Rx = Q^H b. \]
QR-factorization as preprocessing step

Bottom part solved by backward substitution.
QR-factorization as preprocessing step

- Apply unitary operations to working copy of solution (auxiliary vector).
- Solve next block with backward substitution.
QR-factorization as preprocessing step

- Apply unitary operations to working auxiliary vector.
- Solve next block with backward substitution.
- Solution of system is obtained.
Outline

Introduction

Low rank approximation

Solving linear system

Numerical results

Problem formulation

Results of numerical experiments

Conclusions
Problem formulation

- Scattering of a time-harmonic wave by a 2D circular obstacle: \( \triangle u + k^2 u = 0 \) Helmholtz equation.
Problem formulation

- Scattering of a time-harmonic wave by a 2D circular obstacle: \( \triangle u + k^2 u = 0 \) Helmholtz equation.
- Apply Galerkin discretization (\( N \) basis functions): \( Ax = b \).
Problem formulation

- Scattering of a time-harmonic wave by a 2D circular obstacle: \( \triangle u + k^2 u = 0 \) Helmholtz equation.
- Apply Galerkin discretization (\( N \) basis functions): \( Ax = b \).
- Two parameters of the problem: \( k \) the wavenumber and \( N \) basis functions (size of \( A \)).
Problem formulation

- Scattering of a time-harmonic wave by a 2D circular obstacle: \( \Delta u + k^2 u = 0 \) Helmholtz equation.
- Apply Galerkin discretization (\( N \) basis functions): \( Ax = b \).
- Two parameters of the problem: \( k \) the wavenumber and \( N \) basis functions (size of \( A \)).
- Problem: hard to solve linear system as \( k \) increases.
Choice of $k$ and $N$

$k = 4, N = 1024$

$k = 64, N = 1024$

Values in numerical experiments: $k = 4, 16, 64$ and $N = 256, 1024$. 
Choice of rank structure

Two parameters:

- Size of the blocks.
- Distance to diagonal.
Choice of rank structure

Two parameters:

- Size of the blocks.
- Distance to diagonal.

Indication: Look at singular values $> \tau$ for different submatrices.
Choice of rank structure

\[ k = 4, \ N = 256 \]

\[ k = 4, \ N = 1024 \]
Choice of rank structure

$k = 64, N = 256$

$k = 64, N = 1024$
Choice of rank structure $\tau = 10^{-6}$

$N = 256$

$N = 1024$
Choice of rank structure $\tau = 10^{-10}$
### Results for accuracy $\tau = 10^{-6}$

<table>
<thead>
<tr>
<th>$k$</th>
<th>$N$</th>
<th>$\frac{|x - \tilde{x}|}{|x|}$</th>
<th>$\frac{|A\tilde{x} - b|}{|b|}$</th>
<th>rank struc</th>
<th>total</th>
<th>rank</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>256</td>
<td>$9.665 \times 10^{-6}$</td>
<td>$3.225 \times 10^{-7}$</td>
<td>48%</td>
<td>57%</td>
<td>16</td>
</tr>
<tr>
<td>4</td>
<td>1024</td>
<td>$4.758 \times 10^{-5}$</td>
<td>$4.495 \times 10^{-7}$</td>
<td>23%</td>
<td>30%</td>
<td>21</td>
</tr>
<tr>
<td>16</td>
<td>256</td>
<td>$1.983 \times 10^{-5}$</td>
<td>$2.099 \times 10^{-6}$</td>
<td>46%</td>
<td>65%</td>
<td>20</td>
</tr>
<tr>
<td>16</td>
<td>1024</td>
<td>$6.545 \times 10^{-5}$</td>
<td>$2.371 \times 10^{-6}$</td>
<td>28%</td>
<td>34%</td>
<td>28</td>
</tr>
<tr>
<td>64</td>
<td>256</td>
<td>$1.512 \times 10^{-5}$</td>
<td>$3.990 \times 10^{-6}$</td>
<td>70%</td>
<td>90%</td>
<td>24</td>
</tr>
<tr>
<td>64</td>
<td>1024</td>
<td>$7.099 \times 10^{-5}$</td>
<td>$9.192 \times 10^{-6}$</td>
<td>30%</td>
<td>57%</td>
<td>40</td>
</tr>
</tbody>
</table>
### Results for accuracy $\tau = 10^{-10}$

<table>
<thead>
<tr>
<th>$k$</th>
<th>$N$</th>
<th>$\left| x - \tilde{x} \right|_{|x|}$</th>
<th>$\left| A\tilde{x} - b \right|_{|b|}$</th>
<th>rank struc</th>
<th>total</th>
<th>rank</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>256</td>
<td>$2.280 \times 10^{-9}$</td>
<td>$3.997 \times 10^{-11}$</td>
<td>46%</td>
<td>64%</td>
<td>20</td>
</tr>
<tr>
<td>4</td>
<td>1024</td>
<td>$7.052 \times 10^{-9}$</td>
<td>$5.187 \times 10^{-11}$</td>
<td>30%</td>
<td>41%</td>
<td>32</td>
</tr>
<tr>
<td>16</td>
<td>256</td>
<td>$3.294 \times 10^{-9}$</td>
<td>$2.017 \times 10^{-10}$</td>
<td>59%</td>
<td>73%</td>
<td>28</td>
</tr>
<tr>
<td>16</td>
<td>1024</td>
<td>$1.268 \times 10^{-8}$</td>
<td>$3.468 \times 10^{-10}$</td>
<td>25%</td>
<td>38%</td>
<td>37</td>
</tr>
<tr>
<td>64</td>
<td>256</td>
<td>$1.855 \times 10^{-9}$</td>
<td>$4.401 \times 10^{-10}$</td>
<td>75%</td>
<td>91%</td>
<td>40</td>
</tr>
<tr>
<td>64</td>
<td>1024</td>
<td>$1.274 \times 10^{-8}$</td>
<td>$1.037 \times 10^{-9}$</td>
<td>37%</td>
<td>62%</td>
<td>58</td>
</tr>
</tbody>
</table>
Outline

Introduction

Low rank approximation

Solving linear system

Numerical results

Conclusions
Conclusions

- Efficient algorithm to solve linear system without having to compute all entries of $A$. 
Conclusions

- Efficient algorithm to solve linear system without having to compute all entries of $A$.
- Accurate solution.
Conclusions

- Efficient algorithm to solve linear system without having to compute all entries of $A$.
- Accurate solution.
- Reduction of the number of computed elements.
Thank you for your attention.
References