WISB356, Utrecht, 9 oktober 2012

Scientific Computing

Gerard Sleijpen
Rob Bisseling
Alessandro Sbrizzi

Universiteit Utrecht
Department of Mathematics

http://www.staff.science.uu.nl/∼sleij101/

WISB356, Utrecht, 9 oktober 2012

Preconditioneren van iteratieve methoden

Gerard Sleijpen

Universiteit Utrecht
Department of Mathematics

http://www.staff.science.uu.nl/∼sleij101/

Program

- Flexible GCR
- Preconditioning
- D-ILU
- Incomplete LU-decomposition
- Why preconditioning?
- Costs
- How to include a preconditioner
- Savings

Flexible GCR

Choose $tol > 0$, $\mathbf{x}$, $k_{\max}$,
Compute $\mathbf{r} = \mathbf{b} - \mathbf{A}\mathbf{x}$
For $k = 0, 1, 2, \ldots, k_{\max}$
  Stop if $\|\mathbf{r}\|_2 \leq tol\|\mathbf{b}\|_2$
  Find an appropriate search vector $\mathbf{u}_k$
  $\mathbf{c}_k = \mathbf{A}\mathbf{u}_k$
  For $j = 0, 1, 2, \ldots, k-1$
    $\beta \leftarrow \mathbf{c}_j^*\mathbf{c}_k/\sigma_j$
    $\mathbf{u}_k = \mathbf{u}_k - \beta\,\mathbf{u}_j$
    $\mathbf{c}_k = \mathbf{c}_k - \beta\,\mathbf{c}_j$
  end for
  $\sigma_k = \mathbf{c}_k^*\mathbf{c}_k$, $\alpha \leftarrow \mathbf{c}_k^*\mathbf{r}/\sigma_k$
  $\mathbf{x} \leftarrow \mathbf{x} + \alpha\,\mathbf{u}_k$
  $\mathbf{r} \leftarrow \mathbf{r} - \alpha\,\mathbf{c}_k$
end for

Preconditioned GCR

Choose $tol > 0$, $\mathbf{x}$, $k_{\max}$,
Compute $\mathbf{r} = \mathbf{b} - \mathbf{A}\mathbf{x}$
For $k = 0, 1, 2, \ldots, k_{\max}$

  Stop if $\|\mathbf{r}\|_2 \le tol \|\mathbf{b}\|_2$
  Solve $\mathbf{M}\mathbf{u}_k = \mathbf{r}_k$ for $\mathbf{u}_k$
  $\mathbf{c}_k = \mathbf{A}\mathbf{u}_k$
  For $j = 0, 1, 2, \ldots, k-1$
    $\beta \leftarrow \mathbf{c}_j^* \mathbf{c}_k / \sigma_j$
    $\mathbf{u}_k = \mathbf{u}_k - \beta\, \mathbf{u}_j$
    $\mathbf{c}_k = \mathbf{c}_k - \beta\, \mathbf{c}_j$
  end for
  $\sigma_k = \mathbf{c}_k^* \mathbf{c}_k$, $\alpha \leftarrow \mathbf{c}_k^* \mathbf{r} / \sigma_k$
  $\mathbf{x} \leftarrow \mathbf{x} + \alpha\, \mathbf{u}_k$
  $\mathbf{r} \leftarrow \mathbf{r} - \alpha\, \mathbf{c}_k$

end for

Preconditioning

An n by n matrix $\mathbf{M}$ is called a **preconditioner** if

- the system $\mathbf{M}\mathbf{u}_k = \mathbf{r}_k$ can efficiently be solved and
- $\mathbf{M}$ approximates $\mathbf{A}$ (to some degree).

Examples.

- **Diagonal preconditioning**. $\mathbf{M} \equiv \mathbf{D}_A \equiv \mathrm{diag}(\mathbf{A})$.
- **Gauss–Seidel**. $\mathbf{M} \equiv \mathbf{L}_A + \mathbf{D}_A$.
- **Symmetric Successive overrelaxation**.

$$\mathbf{M} \equiv (\mathbf{L}_A + \mathbf{D})\mathbf{D}^{-1}(\mathbf{D} + \mathbf{U}_U)$$

with $\mathbf{D} \equiv \frac{1}{\omega}\mathbf{D}_A$ for a **relaxation** parameter ω.

Note that $\mathbf{M} = \mathbf{A} + \mathbf{R}$ for

$$\mathbf{R} \equiv (\tfrac{1}{\omega} - 1)\mathbf{D}_A + \mathbf{L}_A \mathbf{D}^{-1}\mathbf{U}_A$$

Preconditioning

$$\mathbf{M} \equiv (\mathbf{L}_A + \mathbf{D})\mathbf{D}^{-1}(\mathbf{D} + \mathbf{U}_A) = \mathbf{A} + \mathbf{R}$$

The system $\mathbf{M}\mathbf{u} = \mathbf{r}$ can be solved in three steps.

- Solve $(\mathbf{L}_A + \mathbf{D})\mathbf{u}' = \mathbf{r}$ for $\mathbf{u}'$.
- Compute $\mathbf{u}'' = \mathbf{D}\mathbf{u}'$.
- Solve $(\mathbf{D} + \mathbf{U}_A)\mathbf{u} = \mathbf{u}''$ for $\mathbf{u}$.

Assignment. Write a function subroutine

$$\mathbf{u} = \texttt{Msolve}(\mathbf{A}, \mathbf{D}, \mathbf{r})$$

that incorporates the above steps. Try to make the routine as efficient as possible also concerning use of memory.

Incorporate `Msolve` in GCR: write a routine PGCR

$$\mathbf{x} = \texttt{PGCR}(\mathbf{A}, \mathbf{b}, \mathbf{x}_0, tol, k_{\max}, \mathbf{D})$$

Preconditioning

Find a diagonal matrix $\mathbf{D}$ such that with

$$\mathbf{M} \equiv (\mathbf{L}_A + \mathbf{D})\mathbf{D}^{-1}(\mathbf{D} + \mathbf{U}_A) = \mathbf{A} + \mathbf{R}$$

the "error"

$$\mathbf{R} \equiv \mathbf{D} - \mathbf{D}_A + \mathbf{L}_A \mathbf{D}^{-1}\mathbf{U}_A$$

is small in some sense.

Examples.

- **Diagonal-Incomplete LU**: $\mathrm{diag}(\mathbf{R}) = \mathbf{0}$.
- **D-Modified ILU**: $\mathbf{R}\mathbf{1} = \mathbf{0}$, with $\mathbf{1} \equiv (1, 1, \ldots, 1)^{\mathrm{T}}$
- **D-Relaxed ILU**: a mix of ILU and MILU

LU-decomposition

$\mathbf{L}$ is strict lower triangular $n \times n$ (i.e., $\mathbf{L}_{ij} = 0$ if $i \le j$).
$\mathbf{I}$ is the $n \times n$ identity. $\ell_j \equiv \mathbf{L}(:, j)$, $\mathbf{e}_j \equiv \mathbf{I}(:, j)$.

LU-decomposition or **Gaussian elimination**:

$\mathbf{U}^{(0)} \equiv \mathbf{A}, \mathbf{U}^{(1)}, \dots, \mathbf{U}^{(n-1)} = \mathbf{U}$ such that

$$\mathbf{U}^{(j)} = (\mathbf{I} - \ell_j \mathbf{e}_j^*)\mathbf{U}^{(j-1)} \qquad (j = 1, \dots, n)$$

and the jth column of $\mathbf{U}^{(j)}$ below the diagonal is zero:
with $p_j \equiv \mathbf{U}^{(j-1)}(j, j)$, $\ell_j(i) = \mathbf{U}^{(j-1)}(i, j)/p_j$ for $i > j$.

Theorem. If the **pivots** $p_j \neq 0$ all j, then

$$\mathbf{A} = \mathbf{L}\mathbf{U}$$

Sparsity pattern of $\mathbf{A}$; $\mathcal{F}_A \equiv \{(i, j) \mid \mathbf{A}(i, j) \neq 0\}$

Fill: $\{(i, j) \notin \mathcal{F}_A \mid \mathbf{L}(i, j) \neq 0$ or $\mathbf{U}^{(k)}(i, j) \neq 0\}$

LU-decomposition

$\mathbf{L}$ is strict lower triangular $n \times n$ (i.e., $\mathbf{L}_{ij} = 0$ if $i \le j$).
$\mathbf{I}$ is the $n \times n$ identity. $\ell_j \equiv \mathbf{L}(:, j)$, $\mathbf{e}_j \equiv \mathbf{I}(:, j)$.

Incomplete LU-decomposition.
Select a **fill pattern** $\mathcal{F} \subset \{(i, j) \mid i, j = 1, \dots, n\}$.
If $\mathbf{B}$ is an $n \times n$ matrix, then $\mathbf{B}'$ is the matrix with entries
$\mathbf{B}'(i, j) = \mathbf{B}(i, j)$ if $(i, j) \in \mathcal{F}$ and $\mathbf{B}'(i, j) = 0$ if $(i, j) \notin \mathcal{F}$.
Put $\Pi(\mathbf{B}) = \mathbf{B}'$.

$\mathbf{U}^{(0)} \equiv \mathbf{A}$, $\mathbf{U}^{(1)}, \dots, \mathbf{U}^{(n-1)} = \mathbf{U}$ such that

$$\widetilde{\mathbf{U}}^{(j)} = (\mathbf{I} - \ell_j \mathbf{e}_j^*)\mathbf{U}^{(j-1)}, \qquad \mathbf{U}^{(j)} = \Pi(\widetilde{\mathbf{U}}^{(j)})$$

and the j column of $\widetilde{\mathbf{U}}^{(j)}$ below the diagonal is zero:
with $p_j \equiv \mathbf{U}^{(j-1)}(j, j)$, $\ell_j(i) = \mathbf{U}^{(j-1)}(i, j)/p_j$ for $i > j$.

Theorem. With ILU and $\mathbf{M} = \mathbf{L}\mathbf{U}$,
we have that $\mathbf{A}(i, j) = \mathbf{M}(i, j)$ for all $(i, j) \in \mathcal{F}$

LU-decomposition

$\mathbf{L}$ is strict lower triangular $n \times n$ (i.e., $\mathbf{L}_{ij} = 0$ if $i \le j$).
$\mathbf{I}$ is the $n \times n$ identity. $\ell_j \equiv \mathbf{L}(:, j)$, $\mathbf{e}_j \equiv \mathbf{I}(:, j)$.

Modified ILU-decomposition. Select a
fill pattern $\mathcal{F} \subset \{(i, j) \mid i, j = 1, \dots, n\}$ with $\{(i, i)\} \subset \mathcal{F}$.
If $\mathbf{B}$ is an $n \times n$ matrix, then $\widetilde{\mathbf{B}}$ is the matrix with entries

$$\widetilde{\mathbf{B}}(i, j) = \mathbf{B}(i, j) \text{ if } (i, j) \in \mathcal{F}, i \neq j$$

$$\widetilde{\mathbf{B}}(i, j) = 0 \text{ if } (i, j) \notin \mathcal{F},$$

$$\widetilde{\mathbf{B}}(i, i) = \mathbf{B}(i, i) + \textstyle\sum_{j,(i,j) \notin \mathcal{F}} \mathbf{B}(i, j)$$

Put $\Pi_M(\mathbf{B}) = \widetilde{\mathbf{B}}$. **Note**. $\mathbf{B}\mathbf{1} = \Pi_M(\mathbf{B})\mathbf{1}$.

$\mathbf{U}^{(0)} \equiv \mathbf{A}$, $\mathbf{U}^{(1)}, \dots, \mathbf{U}^{(n-1)} = \mathbf{U}$ such that

$$\widetilde{\mathbf{U}}^{(j)} = (\mathbf{I} - \ell_j \mathbf{e}_j^*)\mathbf{U}^{(j-1)}, \qquad \mathbf{U}^{(j)} = \Pi_M(\widetilde{\mathbf{U}}^{(j)})$$

and the jth column of $\widetilde{\mathbf{U}}^{(j)}$ below the diagonal is zero:
with $p_j \equiv \mathbf{U}^{(j-1)}(j, j)$, $\ell_j(i) = \mathbf{U}^{(j-1)}(i, j)/p_j$ for $i > j$.

LU-decomposition

$\mathbf{L}$ is strict lower triangular $n \times n$ (i.e., $\mathbf{L}_{ij} = 0$ if $i \le j$).
$\mathbf{I}$ is the $n \times n$ identity. $\ell_j \equiv \mathbf{L}(:, j)$, $\mathbf{e}_j \equiv \mathbf{I}(:, j)$.

Theorem. With MILU-decomposition and $\mathbf{M} \equiv \mathbf{L}\mathbf{U}$,
we have that $\mathbf{M}\mathbf{1} = \mathbf{A}\mathbf{1}$, where $\mathbf{1} \equiv (1, 1, \dots, 1)^{\mathsf{T}}$.

LU-decomposition

$\mathbf{L}$ is strict lower triangular $n \times n$ (i.e., $\mathbf{L}_{ij} = 0$ if $i \leq j$).
$\mathbf{I}$ is the $n \times n$ identity. $\ell_j \equiv \mathbf{L}(:,j)$, $\mathbf{e}_j \equiv \mathbf{I}(:,j)$.

Relaxed ILU-decomposition. Select an $\omega \in [0,1]$ and a **fill pattern** $\mathcal{F} \subset \{(i,j) \mid i,j = 1,\ldots,n\}$ with $\{(i,i)\} \subset \mathcal{F}$.
If $\mathbf{B}$ is an $n \times n$ matrix, then $\widetilde{\mathbf{B}}$ is the matrix with entries

$$\widetilde{\mathbf{B}}(i,j) = \mathbf{B}(i,j) \text{ if } (i,j) \in \mathcal{F}, i \neq j$$

$$\widetilde{\mathbf{B}}(i,j) = 0 \text{ if } (i,j) \notin \mathcal{F},$$

$$\widetilde{\mathbf{B}}(i,i) = \mathbf{B}(i,i) + \omega \textstyle\sum_{j,(i,j)\notin\mathcal{F}} \mathbf{B}(i,j)$$

Put $\Pi_\omega(\mathbf{B}) = \widetilde{\mathbf{B}}$.

$\mathbf{U}^{(0)} \equiv \mathbf{A}$, $\mathbf{U}^{(1)}, \ldots, \mathbf{U}^{(n-1)} = \mathbf{U}$ such that

$$\widetilde{\mathbf{U}}^{(j)} = (\mathbf{I} - \ell_j \mathbf{e}_j^*)\mathbf{U}^{(j-1)}, \qquad \mathbf{U}^{(j)} = \Pi_\omega(\widetilde{\mathbf{U}}^{(j)})$$

and the jth column of $\widetilde{\mathbf{U}}^{(j)}$ below the diagonal is zero:
with $p_j \equiv \mathbf{U}^{(j-1)}(j,j)$, $\ell_j(i) = \mathbf{U}^{(j-1)}(i,j)/p_j$ for $i > j$.

LU-decomposition

$\mathbf{L}$ is strict lower triangular $n \times n$ (i.e., $\mathbf{L}_{ij} = 0$ if $i \leq j$).
$\mathbf{I}$ is the $n \times n$ identity. $\ell_j \equiv \mathbf{L}(:,j)$, $\mathbf{e}_j \equiv \mathbf{I}(:,j)$.

Remark. RILU(0)=ILU, RILU(1)=MILU.

Diagonal ILU decomposition

Write $\mathbf{A} = \mathbf{L}_A + \mathbf{D}_A + \mathbf{U}_A$ with
$\mathbf{L}_A$ the strict lower triangular part of $\mathbf{A}$
($\mathbf{L}_A(i,j) = \mathbf{A}(i,j)$ if $i > j$, $\mathbf{L}_A(i,j) = 0$ if $i \leq j$)
$\mathbf{D}_A = \mathrm{diag}(\mathbf{A})$ (in Matlab: `D_A=diag(diag(A));`)
$\mathbf{U}_A$ the strict upper triangular part of $\mathbf{A}$.

For an $n \times n$ diagonal matrix $\mathbf{D}$ consider

$$\mathbf{M} \equiv (\mathbf{L}_A + \mathbf{D})\mathbf{D}^{-1}(\mathbf{D} + \mathbf{U}_A)$$

D-MILU: $\mathbf{D}$ is such that $\mathbf{M1} = \mathbf{A1}$.

Theorem. If $\mathbf{A}$ is the matrix from a 5-point stencil (2-d advection diffusion) or from a 7-point stencil (3-d advection diffusion), then
D-ILU=ILU(0) and D-MILU=MILU(0)

Other ILU-decompositions

For a fill pattern $\mathcal{F} \subset \{(i,j) \mid i,j = 1,\ldots,n\}$, define

$$\mathcal{F}^+ \equiv \{(i,j) \mid (i,k),(k,j) \in \mathcal{F} \text{ for some } k < i,\ k < j\}$$

Terminology. With $\mathcal{F}_A(0) \equiv \mathcal{F}_A \equiv \{(i,j) \mid \mathbf{A}(i,j) \neq 0\}$,

$$\mathcal{F}_A(\ell) \equiv \mathcal{F}_A(\ell-1)^+ \text{ for } \ell = 1,2,\ldots$$

$\mathcal{F}_A(\ell)$ is **fill of level ℓ**.

Note that to determine $\mathcal{F}_A(\ell)$ no specific values for the entries of $\mathbf{A}$ are required.

ILU(ℓ), that is, ILU for $\mathbf{A}$ with fill pattern $\mathcal{F}_A(\ell)$,
is called **ILU of level ℓ**.

ILU(0)=ILU.

Other ILU-decompositions

Select an $\varepsilon > 0$ (**the drop tolerance**).

If $\mathbf{B}$ is an $n \times n$ matrix, then $\widetilde{\mathbf{B}}$ is the matrix with entries

$$\widetilde{\mathbf{B}}(i,j) = \mathbf{B}(i,j) \quad \text{if} \quad |\mathbf{B}(i,j)| > \varepsilon$$
$$\widetilde{\mathbf{B}}(i,j) = 0 \quad \text{if} \quad |\mathbf{B}(i,j)| \leq \varepsilon$$

Put $\quad \Pi_\varepsilon(\mathbf{B}) \equiv \widetilde{\mathbf{B}}$.

Using Π_ε in each step of the Gaussian elimination process leads to ILU(ε), **ILU with drop tolerance**

Advanced ILU.

- Drop tolerance and level strategies can be combined.
- The value of the drop tolerance can be selected to depend on the level, on the size of the matrix entries,
- ...

Why preconditioning?

Example. $\begin{cases} -\frac{\partial}{\partial x}\frac{\partial}{\partial x}\phi = 0 \quad \text{on} \quad D \equiv [0,1] \\ \phi(0) = 1, \quad \phi(1) = 0 \end{cases}$

Discretization: symmetric finite differences: $h = \frac{1}{n+1}$

$$\mathbf{A} = \frac{1}{h^2}\begin{bmatrix} 2 & -1 & 0 & \dots & 0 \\ -1 & 2 & -1 & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & \\ \vdots & \ddots & -1 & 2 & -1 \\ & & 0 & -1 & 2 \end{bmatrix}, \quad \mathbf{b} = \frac{1}{h^2}\begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix}.$$

With $\mathbf{x}_0 = \mathbf{0}$, we have $\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0 = \mathbf{b} = \tau\mathbf{e}_1$

GCR and LMR: $\mathbf{r}_{k-1}, \mathbf{x}_k \in \text{span}(\mathbf{e}_1, \dots, \mathbf{e}_k)$ for $k = 1, \dots, n$.

Interpretation. It takes a Krylov subspace method at least n (=grid size) steps to carry the information in $\mathbf{r}_0$ over the whole grid.

Does preconditioning harm?

$$\mathbf{M} = (\mathbf{L}_A + \mathbf{D})\mathbf{D}^{-1}(\mathbf{D} + \mathbf{U}_A).$$

Costs. The costs of computing $\mathbf{D}$ are negligible

An $\mathbf{M}$-solve costs $11n$ flop.

- The extra costs in k-steps for including $\mathbf{M}$ solves in each step of GCR are $11kn$.
- Reduction costs in GCR when reducing the number if steps from $k+m$ to k is (with no precond. for 2-d) is
$$\geq (19n + 6kn)m \text{ flop}.$$

Conclusion. The total costs in GCR already reduces by including $\mathbf{M}$-solves if this leads to a reduction in the number of required steps by 2 steps (if $m \geq 2$ then $(6kn)m \geq 11kn$).

Does preconditioning harm?

$$\mathbf{M} = (\mathbf{L}_A + \mathbf{D})\mathbf{D}^{-1}(\mathbf{D} + \mathbf{U}_A).$$

Costs. The costs of computing $\mathbf{D}$ are negligible

An $\mathbf{M}$-solve costs $11n$ flop.

- The extra costs in k-steps for including $\mathbf{M}$ solves in each step of LMR are $11kn$.
- Reduction costs in LMR when reducing the number if steps from $k+m$ to k is (with no precond. for 2-d) is
$$\geq 19nm \text{ flop}.$$

Conclusion. The total costs in LMR reduces by including $\mathbf{M}$-solves if this leads to a reduction in the number of required steps by 40%.

Convergence ILU preconditioning

Groundwaterflow: $\lambda(\mathbf{A}) \in [\lambda_1, \lambda_n] \subset (0, \infty)$, $\mathcal{C} \equiv \frac{\lambda_n}{\lambda_1}$

$1/\mathcal{C} \sim \max(h_x^2, h_y^2, \ldots)$.

Convergence.
$$\rho_k \equiv \frac{\|\mathbf{r}_k\|}{\|\mathbf{r}_0\|} \leq \exp(-2k/\mu)$$

Without preconditioning

LMR: $\mu = \mathcal{C}$, GCR: $\mu = \sqrt{\mathcal{C}}$

With D-MILU preconditioning

LMR: $\mu = \sqrt{\mathcal{C}}$, GCR: $\mu = \mathcal{C}^{\frac{1}{4}}$.

Example. $\mathcal{C} = 2\,10^4$. GCR:

without precond. $\rho_k \leq 10^{-3}$ for $k = 490$,

with D-MILU $\rho_k \leq 10^{-3}$ for $k = 42$.

Including a preconditioner

- Modify the GCR algorithm (**implicit** preconditioning): replace "$\mathbf{u}_k = \mathbf{r}_k$" by "`Solve` $\mathbf{M}\mathbf{u}_k = \mathbf{r}_k$ `for` $\mathbf{u}_k$".
- Modify the problem to $\mathbf{A}\mathbf{M}^{-1}\tilde{\mathbf{x}} = \mathbf{b}$ (**explicit right** preconditioning): replace $\mathbf{A}$ by $\mathbf{A}\mathbf{M}^{-1}$; $\mathbf{x} = \mathbf{M}^{-1}\tilde{\mathbf{x}}$.
- Modify the problem to $\mathbf{M}^{-1}\mathbf{A}\mathbf{x} = \tilde{\mathbf{b}} \equiv \mathbf{M}^{-1}\mathbf{b}$ (**explicit left** preconditioning): replace $\mathbf{A}$ by $\mathbf{M}^{-1}\mathbf{A}$ and $\mathbf{b}$ by $\tilde{\mathbf{b}}$.

Assignment. Write a routine that perform explicit left preconditioning.

Compare the performance of this routine with the one of GCR with implicit preconditioning (use the same preconditioner and the same $\mathbf{x}_0$). Make sure that you obtain residuals of comparable quality.

Restarted GCR

```
Choose tol > 0, x, k_max, ℓ ∈ ℕ
Compute r = b − Ax
For k = 0 : k_max
    Stop if ||r||_2 ≤ tol ||b||_2
    u_k = r
    c_k = A u_k
    For j = ℓ⌊k/ℓ⌋ : k − 1
        β ← c_j* c_k / σ_j
        u_k = u_k − β u_j
        c_k = c_k − β c_j
    end for
    σ_k = c_k* c_k,   α ← c_k* r / σ_k
    x ← x + α u_k
    r ← r − α c_k
end for
```

Truncated GCR

```
Choose tol > 0, x, k_max, ℓ ∈ ℕ
Compute r = b − Ax
For k = 0 : k_max
    Stop if ||r||_2 ≤ tol ||b||_2
    u_k = r
    c_k = A u_k
    For j = max(k − ℓ, 0) : k − 1
        β ← c_j* c_k / σ_j
        u_k = u_k − β u_j
        c_k = c_k − β c_j
    end for
    σ_k = c_k* c_k,   α ← c_k* r / σ_k
    x ← x + α u_k
    r ← r − α c_k
end for
```

Conjugate Residuals

```
Choose tol > 0, x, k_max
Compute r = b − Ax
For k = 0 : k_max
    Stop if ||r||_2 ≤ tol||b||_2
    u_k = r
    c_k = Au_k
    if k > 0
        β ← c*_{k−1} c_k / σ_{k−1}
        u_k ← u_k − β u_{k−1}
        c_k ← c_k − β c_{k−1}
    end if
    σ_k = c*_k c_k,   α ← c*_k r / σ_k
    x ← x + α u_k
    r ← r − α c_k
end for
```

Theorem. If $\mathbf{A}^* = \mathbf{A}$ then GCR = CR.

that is, in exact arithmetic CR and GCR have the same residual $\mathbf{r}_k$ and the same approximate solution $\mathbf{x}_k$ (when started withe the same initial guess $\mathbf{x}_0$).

Assignment. Program CR. Check that CG = GCR in case the matrix is symmetric (and real).

Include also preconditioning in CR. What is the reduction in number of steps that is required by including preconditioning in order to have a more efficient CR algorithm?

Conjugate Residuals

```
Choose tol > 0, x, k_max
u_1 = c_1 = 0, σ = 1
Compute r = b − Ax
For k = 0 : k_max
    Stop if ||r||_2 ≤ tol||b||_2
    u_0 ← u_1, u_1 ← r
    c_0 ← c_1, c_1 ← Au_k
    β ← c*_0 c_1 / σ
    u_1 ← u_1 − β u_0
    c_1 ← c_1 − β c_0
    σ ← c*_1 c_1,   α ← c*_1 r / σ
    x ← x + α u_1
    r ← r − α c_1
end for
```