Tomáš Oberhuber

Faculty of Nuclear Sciences and Physical Engineering
Czech Technical University in Prague

# Iterativní metody

Budeme se zabývat metodami pro řešení úlohy $\mathbb{A}\vec{x} = \vec{b}$ s regulární maticí $\mathbb{A}$. **Chceme najít efektivnější metody, než je GEM.**

- iterativní metody generují posloupnost $\left\{\vec{x}^{(k)}\right\}_{k=1}^{\infty}$, která konverguje k řešení soustavy

$$\lim_{k\to\infty} \vec{x}^{(k)} = \vec{x}^*$$

- prakticky tak (obecně) nikdy nedostaneme přesné řešení, ale můžeme dostat (téměř) libovolně přesnou aproximaci

# Iterativní metody obecně

- obecně platí, že volíme libovolné $\vec{x}^{(0)}$ a dále napočítáváme

$$\vec{x}^{(k+1)} = \mathbb{B}^{(k)}\vec{x}^{(k)} + \vec{c}^{(k)},$$

kde volba $\mathbb{B}^{(k)}$ a $\vec{c}^{(k)}$ záleží na použité metodě

- dále požadujeme, aby pro všechna $k$ platilo

$$\vec{x}^{*} = \mathbb{B}^{(k)}\vec{x}^{*} + \vec{c}^{(k)},$$

kde $\vec{x}^{*}$ je přesné řešení soustavy $\mathbb{A}\vec{x} = \vec{b}$

# Iterativní metody obecně

## Theorem 1

*Iterativní metoda tvaru*

$$\vec{x}^{(k+1)} = \mathbb{B}^{(k)}\vec{x}^{(k)} + \vec{c}^{(k)},$$

*splňující*

$$\vec{x}^{*} = \mathbb{B}^{(k)}\vec{x}^{*} + \vec{c}^{(k)},$$

*konverguje k* $\vec{x}^{*}$ *pro libovolné* $\vec{x}^{(0)}$ *právě když,*

$$\lim_{k\to\infty} \mathbb{B}^{(k)}\mathbb{B}^{(k-1)}\ldots\mathbb{B}^{(0)} = \theta.$$

## Důkaz.

Platí

$$\begin{aligned}
\vec{x}^{(k)} - \vec{x}^{*} &= \mathbb{B}^{(k-1)}\vec{x}^{(k-1)} + \vec{c}^{(k-1)} - \mathbb{B}^{(k-1)}\vec{x}^{*} - \vec{c}^{(k-1)} \\
&= \mathbb{B}^{(k-1)}\vec{x}^{(k-1)} - \mathbb{B}^{(k-1)}\vec{x}^{*} \\
&= \mathbb{B}^{(k-1)}\left(\vec{x}^{(k-1)} - \vec{x}^{*}\right) \\
&= \mathbb{B}^{(k-1)}\mathbb{B}^{(k-2)}\ldots\mathbb{B}^{(0)}\left(\vec{x}^{(0)} - \vec{x}^{*}\right)
\end{aligned}$$

# Citlivost a stabilita iterativních metod

- pokud během výpočtu dojde k nějaké chybě, třeba vinou zaokrouhlení, můžeme se na výsledný vektor dívat jako na nové $\vec{x}^{(0)}$, **které může být libovolné** a metoda bude konvergovat i tak
- chyby ve výpočtech se tedy nekumulují, ale eliminují
- jde o tzv. **samoopravující** vlastnost iterativních metod
- tím pádem již dále nemusíme studovat citlivost iterativních metod na výpočetní chyby

# Iterativní metody obecně

Iterativní metody pro řešení lineárních soustav se dělí na:

- **stacionární** – $\mathbb{B}^{(k)}$ a $\vec{c}^{(k)}$ jsou konstantní tj.

$$\vec{x}^{(k+1)} = \mathbb{B}\vec{x}^{(k)} + \vec{c}$$

- **nestacionární** – $\mathbb{B}^{(k)}$ a $\vec{c}^{(k)}$ jsou různé pro každé $k$

Výhoda stacionárních metod je, že pracujeme jen s jednou maticí. Jsou proto jednodušší na implementaci a také na numerickou analýzu. Dále budeme studovat pouze stacionární metody.

# Stacionární iterativní metody

## Theorem 2

*Metoda daná vztahem*

$$\vec{x}^{(k+1)} = \mathbb{B}\vec{x}^{(k)} + \vec{c}$$

*a splňující*

$$\vec{x}^{*} = \mathbb{B}\vec{x}^{*} + \vec{c},$$

*konverguje k $\vec{x}^{*}$ pro libovolné $\vec{x}^{(0)}$ právě když platí,*

$$\lim_{k\to\infty} \mathbb{B}^{k} = \theta.$$

## Důkaz.

Plyne z věty 1 a

$$\lim_{k\to\infty} \mathbb{B}^{(k-1)} \ldots \mathbb{B}^{(0)} = \lim_{k\to\infty} \mathbb{B}^{k}.$$

# Stacionární iterativní metody

## Theorem 3

*Metoda daná vztahem*

$$\vec{x}^{(k+1)} = \mathbb{B}\vec{x}^{(k)} + \vec{c},$$

*a splňující*

$$\vec{x}^{*} = \mathbb{B}\vec{x}^{*} + \vec{c},$$

*konverguje k* $\vec{x}^{*}$ *pro libovolné* $\vec{x}^{(0)}$ *právě když platí, že* $\rho\left(\mathbb{B}\right) < 1$.

## Důkaz.

Plyne z podmínek pro konvergenci posloupnosti $\mathbb{B}^{k}$. □

# Stacionární iterativní metody

## Theorem 4

*Postačující podmínkou pro to, aby metoda daná vztahem*

$$\vec{x}^{(k+1)} = \mathbb{B}\vec{x}^{(k)} + \vec{c},$$

*a splňující*

$$\vec{x}^{*} = \mathbb{B}\vec{x}^{*} + \vec{c},$$

*konvergovala k $\vec{x}^{*}$ pro libovolné $\vec{x}^{(0)}$ je, že $\|\mathbb{B}\| < 1$ v nějaké maticové normě $\|\cdot\|$.*

## Důkaz.

Plyne z podmínek pro konvergenci posloupnosti $\mathbb{B}^k$. $\square$

# Stacionární iterativní metody

## Theorem 5

**Aposteriorní odhady chyb pro stacionární iterativní metody:** *Pro metodu danou vztahem*

$$\vec{x}^{(k+1)} = \mathbb{B}\vec{x}^{(k)} + \vec{c},$$

*a splňující*

$$\vec{x}^{*} = \mathbb{B}\vec{x}^{*} + \vec{c},$$

*kde* $\vec{x}^*$ *je řešením soustavy lineárních rovnic* $\mathbb{A}\vec{x} = \vec{b}$*, platí následující odhady chyby aproximace řešení:*

- $\left\|\vec{x}^{(k)} - \vec{x}^*\right\| \leq \left\|\mathbb{A}^{-1}\right\| \left\|\mathbb{A}\vec{x}^{(k)} - \vec{b}\right\| = \left\|\mathbb{A}^{-1}\right\| \left\|\vec{r}^{(k)}\right\|,$
- $\left\|\vec{x}^{(k)} - \vec{x}^*\right\| \leq \left\|(\mathbb{I} - \mathbb{B})^{-1}\right\| \left\|\mathbb{B}\right\| \left\|\vec{x}^{(k-1)} - \vec{x}^{(k)}\right\|,$

*při použití souhlasné maticové normy s danou vektorovou normou.*

## Remark 6

*Zde jsme použili definici rezidua v k-té iteraci*

$$\vec{r}^{(k)} = \mathbb{A}\vec{x}^{(k)} - \vec{b}.$$

# Stacionární iterativní metody

## Remark 7

- *aposteriorní odhady chyb jsou důležité pro zjištění, kdy výpočet metody zastavit a to sice tehdy, je-li* $\left\|\vec{r}^{(k)}\right\|$ *nebo* $\left\|\vec{x}^{(k-1)} - \vec{x}^{(k)}\right\|$ *dostatečně malé*
- *pojem "dostatečně malé" se často vztahuje k normě matice* $\mathbb{A}$ *nebo pravé strany* $\vec{b}$ *rovnice, tj. požadujeme např.*

$$\frac{\left\|\vec{r}^{(k)}\right\|}{\left\|\vec{b}\right\|} < \epsilon$$

# Stacionární iterativní metody

## Theorem 8

**Apriorní odhad chyby pro stacionární iterativní metody:**
*Buď $\|B\| < 1$ v nějaké maticové normě. Pak pro posloupnost $\left\{\vec{x}^{(k)}\right\}_{k=1}^{\infty}$ generovanou předpisem*

$$\vec{x}^{(k+1)} = \mathbb{B}\vec{x}^{(k)} + \vec{c},$$

*a splňující*

$$\vec{x}^{*} = \mathbb{B}\vec{x}^{*} + \vec{c},$$

*platí*

$$\left\|\vec{x}^{(k)} - \vec{x}^{*}\right\| \leq \|\mathbb{B}\|^{k}\left[\left\|\vec{x}^{(0)}\right\| + \frac{\|\vec{c}\|}{1 - \|\mathbb{B}\|}\right],$$

*kde použitá maticová norma je souhlasná s použitou vektorovou normou.*

# Stacionární iterativní metody

- nyní již víme, za jakých podmínek stacionární metoda konverguje k nějakému vektoru $\vec{x}^*$ a známé odhady chyb
- dále se budeme zabývat otázkou, jak volit matici $\mathbb{B}$ a vektor $\vec{c}$, aby vztah

$$\vec{x}^* = \mathbb{B}\vec{x}^* + \vec{c},$$

platil pro řešení $\vec{x}^*$ soustavy lineárních rovnic $\mathbb{A}\vec{x} = \vec{b}$

# Metoda postupných aproximací

- při návrhu nejjednodušší iterativní metody chceme najít nějaký způsob, jak zlepšit určitou aproximaci řešení $\vec{x}^{(k)}$
- potřebujeme tedy odhadnout, jaké chyby se s danou aproximací dopouštíme
- **jediný odhad**, který máme k dispozici, je reziduum

$$\vec{r}^{(k)} = \mathbb{A}\vec{x}^{(k)} - \vec{b}$$

# Metoda postupných aproximací

- zkusme tedy metodu

$$\vec{x}^{(k+1)} = \vec{x}^{(k)} - \vec{r}^{(k)} = \vec{x}^{(k)} - \mathbb{A}\vec{x}^{(k)} + \vec{b} = (\mathbb{I} - \mathbb{A})\,\vec{x}^{(k)} + \vec{b}$$

- vidíme, že metoda splňuje podmínku

$$\vec{x}^{*} = (\mathbb{I} - \mathbb{A})\,\vec{x}^{*} + \vec{b}\left(\Leftrightarrow \mathbb{A}\vec{x}^{*} = \vec{b}\right)$$

- je tedy

$$\begin{aligned} \mathbb{B} &= \mathbb{I} - \mathbb{A}, \\ \vec{c} &= \vec{b} \end{aligned}$$

- tuto metodu nazveme **metodou postupných aproximací**

# Metoda postupných aproximací

```
bool postupneAproximace( const Matrix& A,
                         const Vector& b,
                         Vector& x,
                         double eps )
{
   double normB = norm( b );
   double r = eps + 1.0;
   Vector y;
   while( r > eps )
   {
      /****
       * Postupne aproximace
       */
      r = 0.0;
      for( int i = 0; i < n; i++ )
      {
         double r_i = b[ i ];
         for( int j = 0; j < n; j++ )
               r_i = r_i - A[ i ][ j ] * x[ j ];
         y[ i ] = x[ i ] - r_i;
         r = r + r_i * r_i;
      }
      /****
       * Vypocet rezidua
       */
      r = sqrt( r ) / normB;
   }
}
```

# Metoda postupných aproximací

- stacionární metody pro řešení soustav lineárních rovnic jsou implementovány v programu `stationary-solver`
  - `--input-file` – vstupní soubor s maticí
  - `--method` – použitá metoda
    - `richardson`, `jacobi`, `gauss-seidel` a `sor`
  - `--relaxation r` – relaxační parametr
  - `--matrix-format dense/ellpack` – formát pro uložení matice soustavy
    - pro řídké matice je určen formát `ellpack`
  - `--initial-value v` – nastavení vektoru $\vec{x}^{(0)}$
  - `--max-iterations n` – maximální možný počet provedených iterací
  - `--convergence-residue r` – hodnota kritéria pro zastavení výpočtu (jde o proměnnou `eps` v kódu)
  - `--verbose n`
    - nastavuje úroveň vypisovaných informací v průběhu výpočtu

# Metoda postupných aproximací

- jak uvidíme později, metoda postupných aproximací je vlastně Richardsonova metoda s relaxačním parametrem rovným 1
- na tuto hodnotu je tento parametr nastaven automaticky, pokud neuvedeme jinak

## Example 9

```
stationary-solver --method richardson \
                  --input-file data/matrix-pd.mtx
```

# Metoda postupných aproximací

## Theorem 10

*Metoda postupných aproximací pro rovnici $\mathbb{A}\vec{x} = \vec{b}$ (s regulární maticí $\mathbb{A}$) ve tvaru*

$$\vec{x}^{(k+1)} = (\mathbb{I} - \mathbb{A})\,\vec{x}^{(k)} + \vec{b},$$

*konverguje pro každé $\vec{x}^{(0)}$ k řešení zadané rovnice právě když*

$$\rho\,(\mathbb{I} - \mathbb{A}) < 1.$$

*Je-li*

$$\|\mathbb{I} - \mathbb{A}\| < 1,$$

*pro nějakou maticovou normu $\|\cdot\|$, pak tato metoda také konverguje k řešení soustavy pro libovolné $\vec{x}^{(0)}$.*

# Metoda postupných aproximací

## Theorem 11

*Buď $p(x)$ polynom proměnné $x$. Buď $\mathbb{A} \in C^{n,n}$, $\lambda \in \sigma(\mathbb{A})$. Pak $p(\lambda) \in \sigma(p(\mathbb{A}))$.*

## Důkaz.

Lze ukázat snadno s využitím Jordanova tvaru matice. □

# Metoda postupných aproximací

## Theorem 12

*Buď $\mathbb{A}$ hermitovská a PD. Pak metoda postupných aproximací konverguje právě když platí*

$$2\mathbb{I} > \mathbb{A} > 0.$$

## Důkaz.

Je-li $\mathbb{A}$ hermitovská a má-li být $\rho(\mathbb{I} - \mathbb{A}) < 1$, pak musí být $\sigma(\mathbb{I} - \mathbb{A}) \subset (-1, 1)$ a podle předchozí věty

$$\sigma(\mathbb{A}) \in (0, 2) \Leftrightarrow 2\mathbb{I} > A > 0.$$

$\square$

## Remark 13

*Takových matic ale moc není.*

# Předpodmíněná metoda postupných aproximací

- abychom zlepšili konvergenci metody postupných aproximací, použijeme tzv. **předpodmínění**
- soustavu $\mathbb{A}\vec{x} = \vec{b}$ vynásobíme vhodnou regulární maticí $\mathbb{H}$
- dostaneme soustavu

$$\mathbb{H}\mathbb{A}\vec{x} = \mathbb{H}\vec{b}$$

- řešení se tím nezmění

# Předpodmíněná metoda postupných aproximací

- metoda postupných aproximací má nyní mít tvar

$$
\begin{aligned}
\vec{x}^{(k+1)} &= \vec{x}^{(k)} - \vec{r}^{(k)} \\
&= \vec{x}^{(k)} - \mathbb{H}\mathbb{A}\vec{x}^{(k)} + \mathbb{H}\vec{b} \\
&= \vec{x}^{(k)} - \mathbb{H}\left(\mathbb{A}\vec{x}^{(k)} - \vec{b}\right) \\
&= \left(\mathbb{I} - \mathbb{H}\mathbb{A}\right)\vec{x}^{(k)} + \mathbb{H}\vec{b}
\end{aligned}
$$

- a samozřejmě platí podmínka

$$
\left(\mathbb{I} - \mathbb{H}\mathbb{A}\right)\vec{x}^{*} + \mathbb{H}\vec{b} = \vec{x}^{*} - \mathbb{H}\mathbb{A}\vec{x}^{*} + \mathbb{H}\vec{b} = \vec{x}^{*}
$$

# Předpodmíněná metoda postupných aproximací

- ze vztahu $\vec{x}^{(k+1)} = (\mathbb{I} - \mathbb{H}\mathbb{A})\,\vec{x}^{(k)} + \mathbb{H}\vec{b}$ vidíme, že

$$
\begin{aligned}
\mathbb{B} &= \mathbb{I} - \mathbb{H}\mathbb{A}, \\
\vec{c} &= \mathbb{H}\vec{b}
\end{aligned}
$$

- platí tedy následující věty

# Předpodmíněná metoda postupných aproximací

## Theorem 14

*Předpodmíněná metoda postupných aproximací pro rovnici $\mathbb{A}\vec{x} = \vec{b}$ (s regulární maticí $\mathbb{A}$) s předpodmíněním $\mathbb{H}$ ve tvaru*

$$\vec{x}^{(k+1)} = \left(\mathbb{I} - \mathbb{H}\mathbb{A}\right)\vec{x}^{(k)} + \mathbb{H}\vec{b},$$

*konverguje pro každé $\vec{x}^{(0)}$ k řešení zadané rovnice právě když*

$$\rho\left(\mathbb{I} - \mathbb{H}\mathbb{A}\right) < 1.$$

*Je-li*

$$\|\mathbb{I} - \mathbb{H}\mathbb{A}\| < 1,$$

*pro nějakou maticovou normu $\|\cdot\|$, pak tato metoda také konverguje k řešení soustavy pro libovolné $\vec{x}^{(0)}$.*

# Předpodmíněná metoda postupných aproximací

## Theorem 15

*Buď $\mathbb{A}$ hermitovská a PD. Pak předpodmíněná metoda postupných aproximací konverguje, pokud platí*

$$\mathbb{W} + \mathbb{W}^* > \mathbb{A} > 0,$$

*kde $\mathbb{W} = \mathbb{H}^{-1}$. Konvergence je navíc monotónní vzhledem k vektorové normě $\|\cdot\|_A$.*

## Remark 16

*Pro metodu postupných aproximací bez předpodmínění, je $\mathbb{W} = \mathbb{I}$ a dostáváme již dokázané kritérium*

$$2\mathbb{I} > \mathbb{A} > 0.$$

*Tato věta nám ale neříká nutnou podmínku!!!*

# Předpodmíněná metoda postupných aproximací

- nyní vyvstává otázka, jak volit matici $\mathbb{H}$
- ze vztahu $\vec{x}^{(k+1)} = \vec{x}^{(k)} - \mathbb{H}\mathbb{A}\vec{x}^{(k)} + \mathbb{H}\vec{b}$ dostáváme, že při volbě $\mathbb{H} = \mathbb{A}^{-1}$ je

$$
\begin{aligned}
\vec{x}^{(k+1)} &= \vec{x}^{(k)} - \mathbb{H}\mathbb{A}\vec{x}^{(k)} + \mathbb{H}\vec{b} \\
&= \vec{x}^{(k)} - \mathbb{A}^{-1}\mathbb{A}\vec{x}^{(k)} + \mathbb{A}^{-1}\vec{b} \\
&= \vec{x}^{(k)} - \vec{x}^{(k)} + \vec{x}^{*} \\
&= \vec{x}^{*}
\end{aligned}
$$

- metoda by tak konvergovala po jedné iteraci
- získat $\mathbb{A}^{-1}$ ale umíme jen pomocí GEM, čemuž jsme se chtěli vyhnout
- umění je najít matici $\mathbb{H}$ tak, aby co nejlépe aproximovala $\mathbb{A}^{-1}$ ale byla snadno dosažitelná

# Richardsonovy iterace

- tato metoda je dána předpisem

$$\vec{x}^{(k+1)} = \vec{x}^{(k)} - \theta(\mathbb{A}\vec{x}^{(k)} - \vec{b}),$$

  pro $\theta \in \mathbb{R}$
- parametru $\theta$ se říká **relaxační parametr**
- snadno vidíme, že

$$\begin{aligned} \mathbb{H} &= \theta\mathbb{I} \\ \mathbb{B} &= \mathbb{I} - \theta\mathbb{A} \\ \vec{c} &= \theta\vec{b} \end{aligned}$$

# Richardsonovy iterace

```
bool RichardsonovyIterace( const Matrix& A,
                           const Vector& b,
                           Vector& x,
                           double theta,
                           double eps )
{
   double normB = norm( b );
   double r = eps + 1.0;
   Vector y;
   while( r > eps )
   {
      /****
       * Richardsonovy iterace
       */
      r = 0.0;
      for( int i = 0; i < n; i++ )
      {
         double r_i = b[ i ];
         for( int j = 0; j < n; j++ )
               r_i = r_i - A[ i ][ j ] * x[ j ];
         y[ i ] = x[ i ] - theta * r_i;
         r = r + r_i * r_i;
      }
      /****
       * Vypocet rezidua
       */
      r = sqrt( r ) / normB;
   }
}
```

# Richardsonovy iterace

## Example 17

```
stationary-solver --method richardson \
                  --relaxation 0.5 \
                  --input-file data/matrix-pd.mtx
stationary-solver --method richardson \
                  --relaxation 1.0 \
                  --input-file data/matrix-pd.mtx
stationary-solver --method richardson \
                  --relaxation 1.5 \
                  --input-file data/matrix-pd.mtx
stationary-solver --method richardson \
                  --relaxation 2.0 \
                  --input-file data/matrix-pd.mtx
```

# Richardsonovy iterace

## Theorem 18

*Je-li $\mathbb{A}$ hermitovská a PD, pak metoda Richardsonových iterací konverguje, právě když*

$$\frac{2}{\theta}\mathbb{I} > \mathbb{A} > 0.$$

*Konvergence je navíc monotónní vzhledem k vektorové normě $\|\cdot\|_A$.*

## Důkaz.

Důkaz postačující podmínky plyne z věty 15, kde $\mathbb{W} = \frac{1}{\theta}\mathbb{I}$. Že jde o nutnou podmínku lze ukázata podobně jako u metody postupných aproximací. □

## Remark 19

*Pro hermitovskou a PD matici $\mathbb{A}$ metoda konverguje pro*

$$\theta < \frac{2}{\rho(\mathbb{A})}.$$

# Jacobiho metoda

- navrhl ji Carl Gustav Jakob Jacobi (1845)
- u Jacobiho metody napočítáváme $i$-tou složku vektoru $\vec{x}^{(k+1)}$ z vektoru $\vec{x}^{(k)}$ tak, aby byla splněna $i$-tá rovnice soustavy tj.

$$\begin{aligned} x_i^{(k+1)} &= \frac{1}{a_{ii}} \Big( b_i - a_{i1} x_1^{(k)} - a_{i2} x_2^{(k)} - \dots \\ &\quad - a_{i,i-1} x_{i-1}^{(k)} - a_{i,i+1} x_{i+1}^{(k)} - \dots - a_{in} x_n^{(k)} \Big), \end{aligned}$$

pro $i = 1, \dots, n$.

# Jacobiho metoda

```
bool JacobihoMetoda( const Matrix& A,
                     const Vector& b,
                     Vector& x,
                     double eps )
{
   double normB = norm( b );
   double r = eps + 1.0;
   Vector y;
   while( r > eps )
   {
      /****
       * Jacobiho iterace
       */
      for( int i = 0; i < n; i++ )
      {
         double s = 0.0;
         for( int j = 0; j < n; j++ )
            if( j != i )
               s = s + A[ i ][ j ] * x[ j ];
         y[ i ] = ( b[ i ] - s ) / A[ i ][ i ];
      }
      /****
       * Vypocet rezidua
       */
      r = 0.0;
      for( int i = 0; i < n; i++ )
      {
         double Ax_i = 0.0;
         for( int j = 0; j < n; j++ )
            Ax_i = Ax_i + A[ i ][ j ] * y[ j ];
         r = r + ( Ax_i - b_i ) * ( Ax_i - b_i );
         x[ i ] = y[ i ];
      }
      r = sqrt( r ) / normB;
   }
}
```

# Jacobiho metoda

## Example 20

```
stationary-solver --method jacobi \
                  --input-file data/matrix-pd.mtx
```

# Jacobiho metoda - numerická analýza

- matici $\mathbb{A}$ přepíšeme ve tvaru

$$\mathbb{A} = \mathbb{D} - \mathbb{L} - \mathbb{R},$$

kde

- $\mathbb{D}$ je diagonála matice $\mathbb{A}$
- $\mathbb{L}$ jsou záporně vzaté prvky matice $\mathbb{A}$ **pod** diagonálou
- $\mathbb{R}$ jsou záporně vzaté prvky matice $\mathbb{A}$ **nad** diagonálou

# Jacobiho metoda - numerická analýza

- pro numerickou analýzu lze tuto metodu maticově zapsat takto

$$
\begin{aligned}
\vec{x}^{(k+1)} &= \mathbb{D}^{-1}\left[\vec{b} + (\mathbb{D} - \mathbb{A})\,\vec{x}^{(k)}\right] \\
&= \mathbb{D}^{-1}(\mathbb{L} + \mathbb{R})\,\vec{x}^{(k)} + \mathbb{D}^{-1}\vec{b} \\
&= \left(\mathbb{I} - \mathbb{D}^{-1}\mathbb{A}\right)\vec{x}^{(k)} + \mathbb{D}^{-1}\vec{b}
\end{aligned}
$$

# Jacobiho metoda - numerická analýza

- vidíme tedy, že

$$
\begin{aligned}
\mathbb{B} &= \mathbb{I} - \mathbb{D}^{-1}\mathbb{A} \\
\vec{c} &= \mathbb{D}^{-1}\vec{b} \\
\mathbb{H} &= \mathbb{D}^{-1}
\end{aligned}
$$

# Jacobiho metoda - numerická analýza

## Theorem 21

*Nutnou a postačující podmínkou pro to, aby Jacobiho metoda konvergovala pro libovolné $\vec{x}^{(0)}$ je*

$$\rho\left(\mathbb{D}^{-1}(\mathbb{L}+\mathbb{R})\right) < 1.$$

*Postačující podmínkou je*

$$\left\|\mathbb{D}^{-1}(\mathbb{L}+\mathbb{R})\right\| < 1,$$

*v libovolné maticové normě* $\|\cdot\|$.

# Jacobiho metoda - numerická analýza

## Definition 22

**Maticí s převládající diagonálou** nazveme takovou matici, pro kterou platí

$$\sum_{j=1,\ldots,n; j\neq i} \left| a_{ij} \right| < \left| a_{ii} \right| ,$$

pro všechna $i = 1, \ldots n$.

# Jacobiho metoda - numerická analýza

## Theorem 23

*Má-li matice $\mathbb{A}$ převládající diagonálu, pak Jacobiho metoda konverguje k řešení soustavy rovnic $\mathbb{A}\vec{x} = \vec{b}$ pro libovolnou volbu $\vec{x}^{(0)}$.*

## Theorem 24

*Je-li matice $\mathbb{A}$ hermitovská a PD, pak Jacobiho metoda konverguje k řešení soustavy rovnic $\mathbb{A}\vec{x} = \vec{b}$ pro libovolnou volbu $\vec{x}^{(0)}$, právě když platí $2\mathbb{D} > \mathbb{A} > 0$ tj. $\mathbb{A}$ a $2\mathbb{D} - \mathbb{A}$ jsou pozitivně definitní. Konvergence je navíc monotónní vzhledem k vektorové normě $\|\cdot\|_A$.*

## Důkaz.

Důkaz postačující podmínky plyne z věty 15, kde $\mathbb{W} = \mathbb{H}^{-1} = \mathbb{D}$. Že jde o podmínku nutnou lze dokázat podobně jako u metody postupných aproximací, není to ale triviální.

# Jacobiho metoda - příklad

## Example 25

$$\mathbb{A} = \begin{pmatrix} -2 & 1 & 0 \\ 1 & -2 & 1 \\ 0 & 1 & -2 \end{pmatrix}, \vec{b} = \begin{pmatrix} -1 \\ 0 \\ -1 \end{pmatrix}, \vec{x} = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix},$$

| $k$ | $\vec{x}^{(k)}$ | $\vec{r}^{(k)}$ |
|---|---|---|
| 0 | $(0, 0, 0)$ | $(1, 0, 1)$ |
| 1 | $(\frac{1}{2}, 0, \frac{1}{2})$ | $(0, 1, 0)$ |
| 2 | $(\frac{1}{2}, \frac{1}{2}, \frac{1}{2})$ | $(\frac{1}{2}, 0, \frac{1}{2})$ |
| 3 | $(\frac{3}{4}, \frac{1}{2}, \frac{3}{4})$ | $(0, \frac{1}{2}, 0)$ |
| 4 | $(\frac{3}{4}, \frac{3}{4}, \frac{3}{4})$ | $(\frac{1}{4}, 0, \frac{1}{4})$ |
| 5 | $(\frac{7}{8}, \frac{3}{4}, \frac{7}{8})$ | $(0, \frac{1}{4}, 0)$ |
| 6 | $(\frac{7}{8}, \frac{7}{8}, \frac{7}{8})$ | $(\frac{1}{8}, 0, \frac{1}{8})$ |
| ... | ... | ... |

# Gaussova-Seidelova metoda

- roku 1823 jí popsal Carl Friedrich Gauss v jednom dopise svému studentovi Ch.L.Gerlingovi
- roku 1874 jí publikoval Phillip Ludwig von Seidel

# Gaussova-Seidelova metoda

- narozdíl od Jacobiho metody, tato metoda využívá k napočítání nové složky vektoru $\vec{x}_i^{(k+1)}$ již napočítané složky tohoto vektoru tj. $\vec{x}_1^{(k+1)}, \ldots, \vec{x}_{i-1}^{(k+1)}$
- metoda je tedy dána předpisem

$$
\begin{aligned}
x_i^{(k+1)} &= \frac{1}{a_{ii}} \Big( b_i - a_{i1} x_1^{(k+1)} - a_{i2} x_2^{(k+1)} - \ldots \\
&\quad - a_{i,i-1} x_{i-1}^{(k+1)} - a_{i,i+1} x_{i+1}^{(k)} - \ldots - a_{in} x_n^{(k)} \Big),
\end{aligned}
$$

pro $i = 1, \ldots, n$.

# Gaussova-Seidelova metoda

```
bool GaussovaSeidelovaMetoda( const Matrix& A,
                              const Vector& b,
                              Vector& x,
                              double eps )
{
   double normB = norm( b );
   double r = eps + 1.0;
   while( r > eps )
   {
      /****
       * Gaussova-Seidelova iterace
       */

      for( int i = 0; i < n; i++ )
      {
         double s = 0.0;
         for( int j = 0; j < n; j++ )
            if( j != i )
               s = s + A[ i ][ j ] * x[ j ];
         x[ i ] = ( b[ i ] - s ) / A[ i ][ i ];
      }
      /****
       * Vypocet rezidua
       */
      r = 0.0;
      for( int i = 0; i < n; i++ )
      {
         double Ax_i = 0.0;
         for( int j = 0; j < n; j++ )
            Ax_i = Ax_i + A[ i ][ j ] * x[ j ];
         r = r + ( Ax_i - b_i ) * ( Ax_i - b_i );
      }
      r = sqrt( r ) / normB;
   }
}
```

# Gaussova-Seidelova metoda

## Example 26

```
stationary-solver --method gauss-seidel \
                  --input-file data/matrix-pd.mtx
```

# Gaussova-Seidelova metoda - numerická analýza

- pro numerickou analýzu lze tuto metodu maticově zapsat takto

$$
\begin{aligned}
\vec{x}^{(k+1)} &= \mathbb{D}^{-1}\left[\vec{b} + \mathbb{L}\vec{x}^{(k+1)} + \mathbb{R}\vec{x}^{(k)}\right] \\
\mathbb{D}\vec{x}^{(k+1)} - \mathbb{L}\vec{x}^{(k+1)} &= \vec{b} + \mathbb{R}\vec{x}^{(k)} \\
\vec{x}^{(k+1)} &= (\mathbb{D}-\mathbb{L})^{-1}\left[\vec{b} + \mathbb{R}\vec{x}^{(k)}\right] \\
\vec{x}^{(k+1)} &= (\mathbb{D}-\mathbb{L})^{-1}\mathbb{R}\vec{x}^{(k)} + (\mathbb{D}-\mathbb{L})^{-1}\vec{b} \\
\vec{x}^{(k+1)} &= \left[\mathbb{I} - (\mathbb{D}-\mathbb{L})^{-1}\mathbb{A}\right]\vec{x}^{(k)} + \\
&\quad (\mathbb{D}-\mathbb{L})^{-1}\vec{b}
\end{aligned}
$$

# Gaussova-Seidelova metoda - numerická analýza

- vidíme tedy, že

$$
\begin{aligned}
\mathbb{B} &= (\mathbb{D} - \mathbb{L})^{-1}\mathbb{R} \\
\vec{c} &= (\mathbb{D} - \mathbb{L})^{-1}\vec{b} \\
\mathbb{H} &= (\mathbb{D} - \mathbb{L})^{-1}
\end{aligned}
$$

# Gaussova-Seidelova metoda - numerická analýza

## Theorem 27

*Nutnou a postačující podmínkou pro to, aby Gaussova-Seidelova metoda konvergovala pro libovolné* $\vec{x}^{(0)}$ *je*

$$\rho\left((\mathbb{D}-\mathbb{L})^{-1}\mathbb{R}\right) < 1.$$

*Postačující podmínkou je*

$$\left\|(\mathbb{D}-\mathbb{L})^{-1}\mathbb{R}\right\| < 1,$$

*v některé maticové normě* $\|\cdot\|$.

# Gaussova-Seidelova metoda - numerická analýza

## Theorem 28

*Má-li matice $\mathbb{A}$ převládající diagonálu, pak Gaussova-Seidelova metoda konverguje k řešení soustavy $\mathbb{A}\vec{x} = \vec{b}$ pro libovolnou volbu $\vec{x}^{(0)}$.*

## Theorem 29

*Je-li matice $\mathbb{A}$ hermitovská a pozitivně definitní, pak Gaussova-Seidelova metoda konverguje k řešení soustavy $\mathbb{A}\vec{x} = \vec{b}$ pro libovolnou volbu $\vec{x}^{(0)}$. Konvergence je navíc monotónní vzhledem k vektorové normě $\|\cdot\|_A$.*

## Důkaz.

Plyne z věty 15, kde $\mathbb{W} = \mathbb{D} - \mathbb{L}$ a
$\mathbb{W} + \mathbb{W}^* = 2\mathbb{D} - \mathbb{L} - \mathbb{L}^* = \mathbb{D} + \mathbb{A} > \mathbb{A} > 0.$ $\square$

# Gaussova-Seidelova metoda - numerická analýza

## Remark 30

*Buď $\mathbb{A}$ PD matice taková, že $2\mathbb{D} - \mathbb{A}$ není PD. Potom Jacobiho metoda nekonverguje, ale Gaussova-Seidelova ano. Gaussova-Seidelova metoda je považována ze lepší metodu, než Jacobiho, i když v některých případech Jacobiho metoda konverguje lépe nebo konverguje tehdy, když Gaussova-Seidelova metoda nekonverguje.*

# Gaussova-Seidelova metoda - příklad

## Example 31

$$\mathbb{A} = \begin{pmatrix} -2 & 1 & 0 \\ 1 & -2 & 1 \\ 0 & 1 & -2 \end{pmatrix}, \vec{b} = \begin{pmatrix} -1 \\ 0 \\ -1 \end{pmatrix}, \vec{x} = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix},$$

| k | *Jacobi* $\vec{x}^{(k)}$ | *Jacobi* $\vec{r}^{(k)}$ | *Gauss − Seidel* $\vec{x}^{(k)}$ | *Gauss − Seidel* $\vec{r}^{(k)}$ |
|---|---|---|---|---|
| 0 | $(0,0,0)$ | $(1,0,1)$ | $(0,0,0)$ | $(1,0,1)$ |
| 1 | $(\frac{1}{2},0,\frac{1}{2})$ | $(0,1,0)$ | $(\frac{1}{2},\frac{1}{4},\frac{5}{8})$ | $(\frac{3}{4},\frac{5}{8},0)$ |
| 2 | $(\frac{1}{2},\frac{1}{2},\frac{1}{2})$ | $(\frac{1}{2},0,\frac{1}{2})$ | $(\frac{5}{8},\frac{5}{8},\frac{13}{16})$ | $(\frac{3}{8},\frac{3}{16},0)$ |
| 3 | $(\frac{3}{4},\frac{1}{2},\frac{3}{4})$ | $(0,\frac{1}{2},0)$ | $(\frac{13}{16},\frac{13}{16},\frac{29}{32})$ | $(\frac{3}{16},\frac{3}{32},0)$ |
| 4 | $(\frac{3}{4},\frac{3}{4},\frac{3}{4})$ | $(\frac{1}{4},0,\frac{1}{4})$ | $(\frac{29}{32},\frac{29}{32},\frac{61}{64})$ | $(\frac{3}{32},\frac{3}{64},0)$ |
| 5 | $(\frac{7}{8},\frac{3}{4},\frac{7}{8})$ | $(0,\frac{1}{4},0)$ | $(\frac{61}{64},\frac{61}{64},\frac{125}{128})$ | $(\frac{3}{64},\frac{3}{128},0)$ |
| 6 | $(\frac{7}{8},\frac{7}{8},\frac{7}{8})$ | $(\frac{1}{8},0,\frac{1}{8})$ | $(\frac{125}{128},\frac{125}{128},\frac{253}{256})$ | $(\frac{3}{128},\frac{3}{256},0)$ |
| ... | ... | ... | ... | ... |

# Super-relaxační metoda

- David M. Young, Jr., 1950
- je známa zejména pod zkratkou SOR = *Successive Over Relaxation method*
- jde o modifikaci Gaussovy-Seidelovy metody
- metoda funguje velmi dobře zejména na lineární systémy pocházející z metody konečných diferencí pro řešení parabolických nebo eliptických parciálních diferenciálních rovnic

# Super-relaxační metoda

- Gaussovu-Seidelovu metodu můžeme přepsat do tvaru

$$\vec{x}_i^{(k+1)} = \vec{x}_i^{(k)} + \Delta\vec{x}_i^{(k)},$$

kde

$$\begin{aligned}\Delta\vec{x}_i^{(k)} &= \frac{1}{a_{ii}}\Big(-a_{i1}\vec{x}_1^{(k+1)} - a_{i2}\vec{x}_2^{(k+1)} - \ldots - a_{i,i-1}\vec{x}_{i-1}^{(k+1)} \\ &\quad -a_{ii}\vec{x}_i^{(k)} - a_{i,i+1}\vec{x}_{i+1}^{(k)} - \ldots - a_{in}\vec{x}_n^{(k)} + \vec{b}_i\Big)\end{aligned}$$

# Super-relaxační metoda

- SOR metoda je pak definována jako

$$\vec{x}_i^{(k+1)} = \vec{x}_i^{(k)} + \omega \Delta \vec{x}_i^{(k)},$$

- $\omega$ je tzv. relaxační parametr
- je-li $\omega = 1$, dostáváme Gaussovu-Seidelovu metodu
- podobnou modifikaci lze snadno provést i pro Jacobiho metodu

# Super-relaxační metoda

```
bool sorMethod( const Matrix& A,
                const Vector& b,
                Vector& x,
                double eps,
                double omega )
{
   double normB = norm( b );
   double r = eps + 1.0;
   while( r > eps )
   {
      /****
       * SOR iterace
       */

      for( int i = 0; i < n; i++ )
      {
         double s = 0.0;
         for( int j = 0; j < n; j++ )
            s = s + A[ i ][ j ] * x[ j ];
         x[ i ] += omega * ( b[ i ] - s ) / A[ i ][ i ];
      }
      /****
       * Vypocet rezidua
       */
      r = 0.0;
      for( int i = 0; i < n; i++ )
      {
         double Ax_i = 0.0;
         for( int j = 0; j < n; j++ )
            Ax_i = Ax_i + A[ i ][ j ] * x[ j ];
         r = r + ( Ax_i - b_i ) * ( Ax_i - b_i );
      }
      r = sqrt( r ) / normB;
   }
}
```

# Super-relaxační metoda

## Example 32

```
stationary-solver --method sor \
                  --relaxation 1.2 \
                  --input-file data/matrix-pd.mtx
```

# Super-relaxační metoda - numerická analýza

- maticově lze metodu zapsat jako

$$\vec{x}^{(k+1)} = \vec{x}^{(k)} + \omega \mathbb{D}^{-1} \left( \mathbb{L}\vec{x}^{(k+1)} - \mathbb{D}\vec{x}^{(k)} + \mathbb{R}\vec{x}^{(k)} + \vec{b} \right)$$

tj.

$$\vec{x}^{(k+1)} = (\mathbb{I} - \omega(\mathbb{D} - \omega\mathbb{L})^{-1}\mathbb{A})\vec{x}^{(k)} + \omega(\mathbb{D} - \omega\mathbb{L})^{-1}\vec{b},$$

tj.

$$\begin{aligned} \mathbb{B}_\omega &= \mathbb{I} - \omega(\mathbb{D} - \omega\mathbb{L})^{-1}\mathbb{A} \\ &= (\mathbb{D} - \omega\mathbb{L})^{-1}\left((1-\omega)\,\mathbb{D} + \omega\mathbb{R}\right), \\ \mathbb{H}_\omega &= \omega(\mathbb{D} - \omega\mathbb{L})^{-1}, \\ \vec{c}_\omega &= \omega(\mathbb{D} - \omega\mathbb{L})^{-1}\vec{b}. \end{aligned}$$

# Super-relaxační metoda - numerická analýza

## Theorem 33

*Nutnou a postačující podmínkou pro to, aby SOR metoda konvergovala pro libovolné $\vec{x}^{(0)}$ je*

$$\rho\left(\mathbb{B}_\omega\right) < 1.$$

*Postačující podmínkou je*

$$\|\mathbb{B}_\omega\| < 1,$$

*v některé maticové normě $\|\cdot\|$.*

# Super-relaxační metoda - numerická analýza

## Theorem 34

*Pro libovolné $\omega \in \mathbb{R}$ platí*

$$\rho\left(\mathbb{B}_\omega\right) \geq |\omega - 1|,$$

*a tedy SOR metoda nekonverguje pro $\omega \leq 0$ nebo $\omega \geq 2$.*

## Theorem 35

*Má-li matice $\mathbb{A}$ převládající diagonálu, pak SOR metoda konverguje k řešení soustavy $\mathbb{A}\vec{x} = \vec{b}$ pro libovolnou volbu $\vec{x}^{(0)}$ pokud je $0 < \omega \leq 1$.*

## Důkaz.

Zatím neznám. $\square$

# Super-relaxační metoda - numerická analýza

## Theorem 36

*(Ostrowski): Je-li matice $\mathbb{A}$ hermitovská a pozitivně definitní, pak SOR metoda konverguje k řešení soustavy $\mathbb{A}\vec{x} = \vec{b}$ pro libovolnou volbu $\vec{x}^{(0)}$ právě když, $0 < \omega < 2$. Konvergence je navíc monotónní vzhledem k vektorové normě $\|\cdot\|_A$.*

## Důkaz.

Důkaz postačující podmínky z věty 15:

$$\mathbb{W} = \frac{1}{\omega}\mathbb{D} - \mathbb{L} \Rightarrow \mathbb{W} + \mathbb{W}^* = \left(\frac{2}{\omega} - 1\right) D + A > A > 0$$

□

# Super-relaxační metoda - numerická analýza

## Remark 37

*Budeme zkoumat konvergenci SOR metody v závislosti na parametru $\omega$. Kromě konvergence samotné nás bude zajímat i její rychlost, tj. budeme chtít najít $\omega_0$ takové, aby bylo $\rho\left(\mathbb{B}_\omega\right)$ resp. $\|\mathbb{B}_\omega\|$ co nejmenší.*

## Remark 38

*Vliv parametru $\omega$ na rychlost konvergence budeme analyzovat pro speciální typ matic tzv.* **dvoucyklických, shodně uspořádaných**. *Dá se ukázat, že tyto matice vznikají právě při řešení eliptických nebo parabolických rovnic pomocí metody konečných diferencí.*

# Super-relaxační metoda - numerická analýza

## Definition 39

**Elementární permutační maticí** $\mathbb{I}_{ij}$ pro $(1 \leq i, j \leq n)$ nazveme takovou čtvercovou matici $n$-tého řádu, která má všechny diagonální prvky rovny 1, kromě prvků na $i$-tém a $j$-tém řádku, které jsou rovny 0. Mimodiagonální prvky jsou nulové kromě prvků na pozicích $(i, j)$ a $(j, i)$, které jsou rovny 1.

$$
\mathbb{I}_{ij} = \begin{pmatrix}
1 & & & & & & & & & \\
& \ddots & & & & & & & & \\
& & 1 & & & & & & & \\
& & & 0 & \dots & \dots & \dots & 1 & & \\
& & & \vdots & 1 & & & \vdots & & \\
& & & \vdots & & \ddots & & \vdots & & \\
& & & \vdots & & & 1 & \vdots & & \\
& & & 1 & \dots & \dots & \dots & 0 & & \\
& & & & & & & & 1 & \\
& & & & & & & & & \ddots & \\
& & & & & & & & & & 1
\end{pmatrix}
$$

**Permutační maticí nazveme** každou matici, kterou lze vyjádřit jako součin libovolného počtu elementárních permutačních matic.

# Super-relaxační metoda - numerická analýza

- násobením libovolné obdélnikové matice elementární permutační maticí zleva resp. zprava odpovídá prohození $i$-tého a $j$-tého řádku resp. sloupce
- snadno vidíme, že

$$\mathbb{I}_{ij} = \mathbb{I}_{ij}^T = \mathbb{I}_{ij}^{-1}$$

  a tedy elementární permutační matice je symetrická a ortogonální
- ... a tedy i permutační matice je symetrická a ortogonální

# Super-relaxační metoda - numerická analýza

## Definition 40

Čtvercová matice $\mathbb{C}$ se nazývá **slabě cyklická s indexem 2**, jestliže existuje permutační matice $\mathbb{P}$ taková, že matice $\mathbb{PCP}^T$ je blokového tvaru

$$\mathbb{PCP}^T = \begin{pmatrix} \theta & \mathbb{M}_1 \\ \mathbb{M}_2 & \theta \end{pmatrix}$$

# Super-relaxační metoda - numerická analýza

## Definition 41

Nechť $\mathbb{A} = \mathbb{D} - \mathbb{L} - \mathbb{R}$. Matice $\mathbb{A}$ se nazývá **dvoucyklická**, je-li odpovídající Jacobiho matice $\mathbb{B}_J = \mathbb{D}^{-1}(\mathbb{L} + \mathbb{R})$ slabě cyklická s indexem 2.

## Definition 42

Dvoucyklická matice $\mathbb{A}$ se nazývá **shodně uspořádaná**, jestliže vlastní číslo matice

$$\alpha \mathbb{D}^{-1}\mathbb{L} + \frac{1}{\alpha}\mathbb{D}^{-1}\mathbb{R}$$

nazávisí na volbě čísla $\alpha \neq 0$.

# Super-relaxační metoda - numerická analýza

## Theorem 43

*Nechť matice $\mathbb{A}$ je dvoucyklická, shodně uspořádaná. Nechť $\lambda \neq 0$ je vlastní číslo matice $\mathbb{B}_\omega$ a $\omega \neq 0$. Nechť $\mu$ splňuje rovnici*

$$(\lambda + \omega - 1)^2 = \omega^2 \mu^2 \lambda.$$

*Pak $\mu$ je vlastní číslo Jacobiho matice $\mathbb{B}_J$ a naopak, je-li $\mu$ vlastní číslo matice $\mathbb{B}_J$, pak příslušné $\lambda$ je vlastním číslem matice $\mathbb{B}_\omega$. Navíc platí, že pro*

$$\omega_{opt} = \frac{2}{1 + \sqrt{1 - \rho\left(\mathbb{B}_J\right)^2}},$$

*nabývá $\rho\left(\mathbb{B}_\omega\right)$ své minimum a SOR metoda tedy konverguje nejrychleji.*

# Shrnutí

Podmínky na matici $\mathbb{A}$ pro konvergenci metod:

| | s převládající diagonálou | hermitovská a ... |
|---|---|---|
| Jacobi | ✓ | $2\mathbb{D} > \mathbb{A} > 0$ |
| Gauss-Seidel | ✓ | $A > 0$ |
| SOR | ✓ | $A > 0$ |

# Výhody iterativních metod

- jsou-li vhodně použity, jsou mnohem rychlejší než přímé metody založené na GEM
  - GEM má složitost $n^3$, pro velké matice je téměř nepoužitelná
  - napočítání nového $\vec{x}^{(k)}$ je založeno na násobení matice a vektoru nebo velice podobné operaci, která má složitost $n^2$
  - v praxi se často podaří, že dobrou aproximaci řešení získáme po pár iteracích nehledě na velikost matice

# Výhody iterativních metod

Iterativní metody dokáží lépe využít vlastností řídkých matic.

- při šikovném uložení řídkých matic může mít násobení matice a vektoru ještě menší náročnost, než $n^2$
- modifikace GEM pro obecné řídké matice je možná, ale algoritmicky velmi složitá
- při aplikaci GEM na řídkou matici může vzniknout mnoho nových nenulových prvků a narůst tak nároky na paměť
- iterativní metody nemění samotnou maticí $\mathbb{A}$, díky tomu jsou snažší na implementaci
- v některých situacích ani nemáme matici $\mathbb{A}$ explicitně uloženou v paměti, pak je GEM nemožné použít

# Výhody iterativních metod

- iterativní metody dokážou velice efektivně využít znalosti přibližného řešení, konvergují pak velice rychle
- v případě GEMu je nám jakýkoliv odhad řešení k ničemu

# Nevýhody iterativních metod

- většina z nich nedá ani teoreticky přesné řešení po konečném počtu kroků
- musíme umět rozhodnout, kdy zastavit napočítávání dalších iterací $\vec{x}^{(k)}$
- počet nutných iterací silně závisí na konkrétní matici a tím pádem i doba výpočtu, u GEM doba výpočtu na samotné matici téměř nezávisí
- vyšetřování konvergence je často velice složité
  - modifikovaná GEM funguje pro libovolnou regulární matici

# Výpočetní příklad

## Example 44

Pomocí programu `stationary-solver` otestujte jednotlivé metody.

- výhoda stacionárních metod oproti GEM jsou vidět na PD matici `gr_30_30.mtx`
- špatná konvergence na PD matici `bcsstk01.mtx`
- indefinitní matice `pores_1.mtx`

# Shrnutí

- konvergence stacionárních metod, apriorní a aposteriorní odhady chyb
- **Jacobiho, Gaussova-Seidelova a SOR metoda**
  - odvození, maticový tvar (matice $\mathbb{B}, \mathbb{H}$), konvergence
- **porovnání přímých a iteračních metod pro řešení lineárních soustav rovnic**
  - podle čeho byste se rozhodli, kterou třídu metod použijete?