Rozwiązywanie algebraicznych układów równań liniowych metodami iteracyjnymi

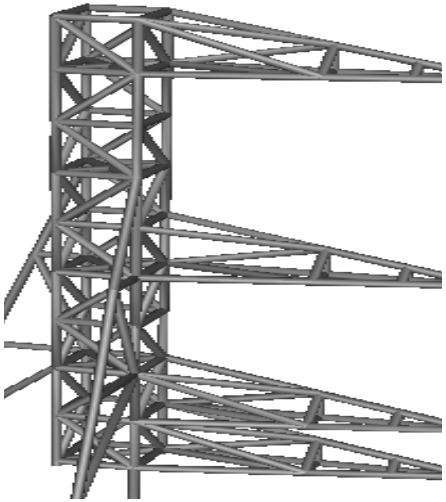
Plan wykładu:

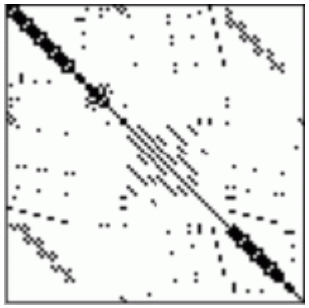
1. Przykłady macierzy rzadkich i formaty ich zapisu
2. Metody: Jacobiego, Gaussa-Seidla, nadrelaksacji
3. Zbieżność metod iteracyjnych
4. Metody: największego spadku, sprzężonego gradientu.

Literatura: Yousef Saad „*Iterative methods for sparse linear systems*”

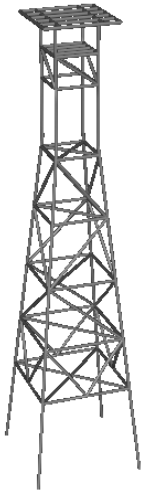
Matrix Market - DWT: Everstine's collection from the Harwell-Boeing Collection

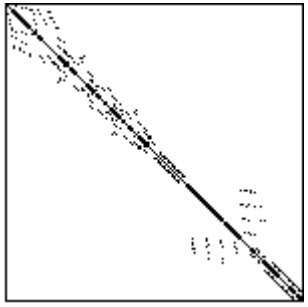
DWT 87: Wieża



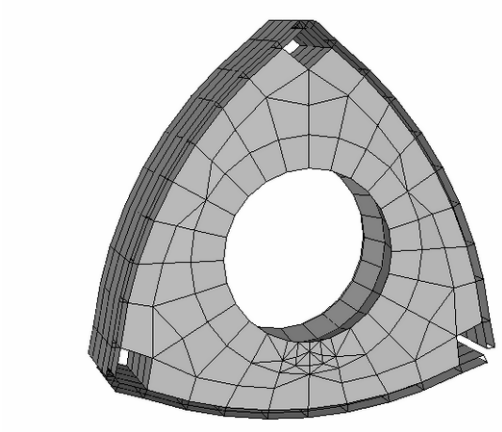


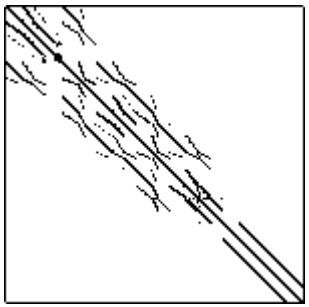
DWT 234: Wieża z platformą





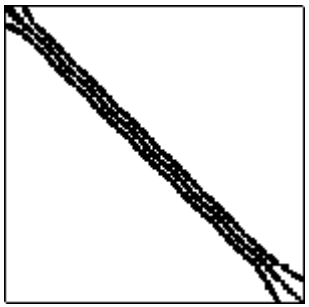
DWT 607: Wirnik Wankela





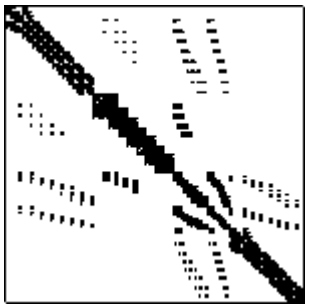
Łopata turbiny

2802 x 2802,
6342 NZ



Cylinder z kołnierzem

2919 x 2919,
7593 NZ



Wybrane formaty zapisu macierzy rzadkich

- **CSR** (compressed sparse row) – trzy wektory: wartości, numery kolumn, początki wierszy (pierwsze nie-zero w wierszu)

- **CSC** (compressed sparse column) – trzy wektory: wartości, numery wierszy, początki kolumn (pierwsze nie-zero w kolumnie)

- **CF** (coordinate format) – trzy wektory dla: wartości, oraz numery kolumn i wierszy dla nie-zer

CSR dla macierzy symetrycznej – zapamiętujemy tylko macierz U

$$A = \begin{bmatrix} 1 & -1 & 0 & -3 & 0 \\ -1 & 5 & 0 & 0 & 0 \\ 0 & 0 & 4 & 6 & 4 \\ -3 & 0 & 6 & 7 & 0 \\ 0 & 0 & 4 & 0 & -5 \end{bmatrix}$$

wartości = (1 -1 -3 5 4 6 4 7 -5)
kolumna = (1 2 4 2 3 4 5 4 5)
wiersz = (1 4 5 8 9 **10**)

CSR dla macierzy niesymetrycznej

$$B = \begin{bmatrix} 1 & -1 & 0 & -3 & 0 \\ -2 & 5 & 0 & 0 & 0 \\ 0 & 0 & 4 & 6 & 4 \\ -4 & 0 & 2 & 7 & 0 \\ 0 & 8 & 0 & 0 & -5 \end{bmatrix}$$

wartości = (1 -1 -3 -2 5 4 6 4 -4 2 7 8 -5)
kolumna = (1 2 4 1 2 3 4 5 1 3 4 2 5)
wiersz = (1 4 6 9 12 **14**)

Mnożenie Ax=y w formacie **CSR** dla macierzy niesymetrycznej

$$\sum_{j=1}^{n} a_{ij} x_j = y_i$$

a[ ] - elementy macierzowe,
k[] - numery kolumn,
w[]-indeksy z poczatkami wierszy

$$B = \begin{bmatrix} 1 & -1 & 0 & -3 & 0 \\ -2 & 5 & 0 & 0 & 0 \\ 0 & 0 & 4 & 6 & 4 \\ -4 & 0 & 2 & 7 & 0 \\ 0 & 8 & 0 & 0 & -5 \end{bmatrix}$$

```
y=0;
for(i=1; i<=n; i++){

   l1=w[i];          // początek indeksow dla i-tego wiersza

   l2=w[i+1]-1;      // koniec indeksów dla i-tego wiersza

  for(l=l1; l<=l2; l++){

     j=k[l];    //numer kolumny

    y[i]=y[i]+a[l]*x[j];

  }
}
```

Mnożenie Ax=y w formacie **CSR** dla macierzy symetrycznej

A=U+D+L oraz $L=U^T$

$(U+D+L)x = y = (D+U)x + (x^TU)^T$

czyli $$\sum_{j=1}^{n} a_{ij}x_j = y_i \rightarrow \sum_{j\geq i}^{n} a_{ij}x_j + \sum_{j<i} x_j a_{ji}$$

- jeśli zamienimy wskaźniki w drugiej sumie to element $a_{ij}*x_i$ będzie dawać wkład do y_j , które zostanie wyznaczone później (j>i) – w ten sposób dochodząc do wiersza j-tego wartość drugiej sumy będzie znana.

a[ ] - wektor elementów macierzowych, k[] - numery kolumn, w[]-indeksy z początkami wierszy

```
y=0;
for(i=1; i<=n; i++){

   l1=w[i];           // początek indeksow dla i-tego wiersza

   l2=w[i+1]-1;  // koniec indeksów dla i-tego wiersza

  for(l=l1; l<=l2; l++){

     j=k[l];    //numer kolumny

    y[i]=y[i]+a[l]*x[j];

    if( i!=j ) y[j]=y[j]+a[l]*x[i];  // sumowanie tyko po elementach pozadiagonalnych
                                     // modyfikacja pochodzi od drugiej sumy
  }
}
```

$$A = \begin{bmatrix} 1 & -1 & 0 & -3 & 0 \\ -1 & 5 & 0 & 0 & 0 \\ 0 & 0 & 4 & 6 & 4 \\ -3 & 0 & 6 & 7 & 0 \\ 0 & 0 & 4 & 0 & -5 \end{bmatrix}$$

Cel - szukamy rozwiązania układu n równań liniowych

$$A\boldsymbol{x} = \boldsymbol{b}, \quad A \in R^{n\times n}, \quad \boldsymbol{x}, \boldsymbol{b} \in R^n$$

ale używając metod iteracyjnych.

Dlaczego używamy metod iteracyjnych?

Przykład

N=50 000 – liczba równań w układzie
fl_2 = 8 bajtów/liczbę – podwójna precyzja

a) Ograniczenia pamięci

$P_d < N^2 * fl_2 = 20$ GB (10GB) – zaalokowana pamięć w komputerze

Ale jeśli układ jest np. pięcioprzekątniowy to do zapisu macierzy A (w postaci wektorowej) potrzebujemy tylko

$P_i < 5N * fl_2 = 2$ MB pamięci

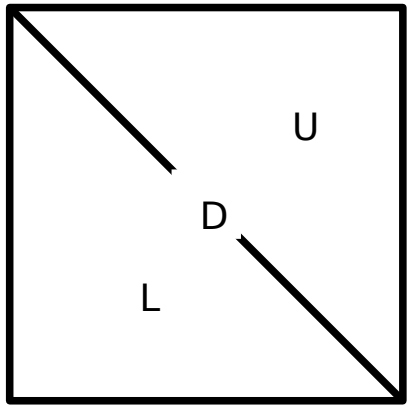
b) większa wydajność dla macierzy rzadkich
(liczba elementów macierzy różnych od 0 jest rzędu N)
w stosunku do metod bezpośrednich

Macierze takie często pojawiają się w obliczeniach naukowych i inżynierskich (FEM, PDE)

Uwaga: alternatywnie możemy zastosować rozkład LU dla macierzy rzadkich

Oznaczmy A jako sumę 3 macierzy

$$A = L + D + U$$



Metoda Jacobiego

$$\boldsymbol{x} = [\xi_1, \xi_2, \ldots, \xi_n]$$
$$\boldsymbol{b} = [\beta_1, \beta_2, \ldots, \beta_n]$$

Dla dowolnie wybranego przybliżenia rozwiązania $\mathbf{x}_0$ chcemy tak przekształacać iteracyjnie wektor $\mathbf{x}^{(k)}$ aby doprowadzić do znikania składowych wektora reszt w k iteracjach

$$(\boldsymbol{b} - A\boldsymbol{x}^{(k)})_i = 0$$

co można zapisać

$$\beta_i - \sum_j^n a_{ij}\xi_j^{(k)} = 0$$

$$a_{ii}\xi_i^{(k)} = \beta_i - \sum_{\substack{j \\ j\neq i}}^{n} a_{ij}\xi_j^{(k)}, \quad i = 1, 2, \ldots, n$$

Składowe wektora reszt znikają w kolejnych iteracjach, więc możemy zapisać

$$\xi_i^{k+1} = \frac{1}{a_{ii}} \left(\beta_i - \sum_{\substack{j \\ j\neq i}}^{n} a_{ij}\xi_j^{(k)} \right)$$

oraz dla całego wektora

$$\boldsymbol{x}^{(k+1)} = -D^{-1}(L+U)\boldsymbol{x}^{(k)} + D^{-1}\boldsymbol{b}$$

W metodzie Jacobiego obliczamy kolejno wszystkie składowe nowego przybliżenia wektora rozwiązań.

Metoda Gaussa-Seidla

Różni się od metody Jacobiego tym, że obliczone już składniki

$$\xi_i^k, \quad i = 1, 2, \ldots, j$$

wykorzystywane są w obliczeniach składników j+1,j+2,...,n.

$$\beta_i - \sum_{j=1}^{i-1} a_{ij}\xi_j^{(k+1)} - a_{ii}\xi_i^{(k+1)} - \sum_{j=i+1}^{n} a_{ij}\xi_j^{(k)} = 0$$

$$\xi_i^{(k+1)} = \frac{1}{a_{ii}} \left(- \sum_{j=1}^{i-1} a_{ij}\xi_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij}\xi_j^{(k)} + \beta_i \right)$$

$$\boldsymbol{b} - L\boldsymbol{x}^{(k+1)} - D\boldsymbol{x}^{(k+1)} - U\boldsymbol{x}^{(k)} = 0$$

$$\boldsymbol{x}^{(k+1)} = -D^{-1}L\boldsymbol{x}^{(k+1)} - D^{-1}U\boldsymbol{x}^{(k)} + D^{-1}\boldsymbol{b}$$

Metody Jacobiego i GS można zapisać ogólnie w postaci

$$M\boldsymbol{x}^{(k+1)} = N\boldsymbol{x}^{(k)} + \boldsymbol{b} = (M - A)\boldsymbol{x}^{(k)} + \boldsymbol{b}$$

$$A = M - N$$

$$M = D \text{ (Jacobi)}$$

$$M = D + L \text{ Gauss-Seidel}$$

Uwaga:

- aby użyć wzoru należy „odwrócić” macierz M – tj. musimy mieć możliwość łatwego rozwiązania układu równań
- macierz diagonalna lub trójkątna daje taką możliwość

Metoda relaksacji

$$\boldsymbol{b} - L\boldsymbol{x}^{(k+1)} - D\boldsymbol{x}^{(k+1)} - U\boldsymbol{x}^{(k)} = 0$$

Wprowadzamy dodatkowy parametr zbieżności: **ω**

$$\begin{aligned} A &= L + D + U \\ \omega A &= \omega D + \omega L + \omega U \\ \omega A &= (D + \omega L) + (\omega U - (1-\omega)D) \end{aligned}$$

Metoda nadrelaksacji (Successive Over Relaxation - SOR)

$$\xi_i^{(k+1)} = \omega \xi_i^{(k+1)GS} + (1-\omega)\xi_i^{(k)}$$

$$(D + \omega L)\boldsymbol{x}^{(k+1)} = [-\omega U + (1-\omega)D]\boldsymbol{x}_k + \omega \boldsymbol{b}, \qquad \omega \in [1, 2]$$

Macierze iteracji i ich przekształcenia (preconditioning)

Ogólny schemat iteracyjny

$$\mathrm{x}^{(k+1)} = G\boldsymbol{x}^{(k)} + \boldsymbol{f}$$

G – macierz iteracji

przy podziale macierzy A

$$A = M - N$$

definiujemy **iteracją do ustalonego punktu** w jako

$$\boldsymbol{x}^{(k+1)} = M^{-1}N\boldsymbol{x}^{(k)} + M^{-1}\boldsymbol{b}$$

Z porównania obu zapisów dostajemy

$$G = M^{-1}N = M^{-1}(M - A) = I - M^{-1}A$$

$$\boldsymbol{f} = M^{-1}\boldsymbol{b}$$

macierze iterujące:

$$\begin{aligned} G_J(A) &= I - D^{-1}A \\ G_{GS}(A) &= I - (D + L)^{-1}A \end{aligned}$$

Proces iteracyjny

$$\boldsymbol{x}^{(k+1)} = G\boldsymbol{x}^{(k)} + \boldsymbol{f}$$

$$G = I - M^{-1}A$$

możemy potraktować także jako problem rozwiązania układu

$$(I - G)\boldsymbol{x} = \boldsymbol{f}$$

$$\lim_{k\to\infty} \boldsymbol{x} \approx \boldsymbol{x}^{(k+1)} \approx \boldsymbol{x}^{(k)}$$

co daje układ równań

$$M^{-1}A\boldsymbol{x} = M^{-1}\boldsymbol{b}$$

$$\boldsymbol{f} = M^{-1}\boldsymbol{b}$$

Układ ten ma identyczne rozwiązanie jak układ pierwotny.
Co nam to daje? Z przepisu iteracyjnego

$$\boldsymbol{x}^{(k+1)} = M^{-1}N\boldsymbol{x}^{(k)} + M^{-1}\boldsymbol{b}$$

wynika, że musimy w każdej iteracji obliczyć

$$M^{-1}N\boldsymbol{x}^{(k)} = M^{-1}\boldsymbol{y}^{(k)} = \boldsymbol{z}^{(k)}$$

Ponieważ M^{-1} nie znamy, więc chcemy **niewielkim kosztem** rozwiązać układ równań

$$M\boldsymbol{z}^{(k)} = \boldsymbol{y}^{(k)}$$

Dla metod Jacobiego, Gaussa-Seidla i SOR
macierz ta ma postać diagonalną lub trójkątną
(postępowanie odwrotne jest szybkie):

$$\begin{aligned} M_J &= D \\ M_{GS} &= D + L \\ M_{SOR} &= \frac{1}{\omega}(D + \omega L) \end{aligned}$$

Zbieżność metod iteracyjnych

Dla macierzy $A \in R^{n\times n}$ definiujemy liczbę

$$\rho(A) = \max_{i=1,2,\ldots,n} |\lambda_i|$$

którą nazywamy **promieniem spektralnym macierzy**.

Dla dowolnej macierzy kwadratowej zgodnej z normą wektorów prawdziwa jest nierówność

$$|\lambda_i| \leq \|A\|, \quad \lambda_i \in Z$$

Lemat $$\bigwedge_{\varepsilon>0} \bigvee_{\|A\|_p} \|A\|_p \leq \rho(A) + \varepsilon$$

Tw. Dla każdego wektora $\boldsymbol{x} \in R^n$ elementy ciągu

$$A\boldsymbol{x},\ A^2\boldsymbol{x}, \ldots,\ A^i\boldsymbol{x}, \ldots$$

dążą do zera wtedy i tylko wtedy gdy $\rho(A) < 1$

Dowód

$$\varepsilon = \frac{1-\rho(A)}{2}$$
$$\|A\|_p \leq \frac{1+\rho(A)}{2} < 1$$
$$\|A^n\boldsymbol{x}\|_p \leq \|A\|_p^n \cdot \|\boldsymbol{x}\|_p \to 0$$
$$A^n\boldsymbol{x} \to 0$$

Tw. Ciąg wektorów

$$x^{(0)},\ x^{(1)},\ldots,x^{(i)},\ldots$$

którego elementy wyznaczamy według wzoru

$$x^{(i+1)} = Gx^{(i)} + f, \quad i = 0,1,\ldots$$

jest zbieżny do jedynego punktu granicznego wtedy i tylko wtedy, gdy

$$\rho(G) < 1$$

Dowód

$$\boldsymbol{x}^{(i+1)} = G\boldsymbol{x}^{(i)} + \boldsymbol{f} = G(G\boldsymbol{x}^{(i-1)} + \boldsymbol{f}) + \boldsymbol{f} = \ldots$$

$$= G^{i+1}\boldsymbol{x}^{(0)} + (G^i\boldsymbol{f} + G^{i-1}\boldsymbol{f} + \ldots + \boldsymbol{f})$$

$$\lim_{i\to\infty} G^{i+1}\boldsymbol{x}^{(0)} \to 0$$

$$\|\boldsymbol{f} + G^1\boldsymbol{f} + \ldots + G^i\boldsymbol{f} + \ldots\|_p \leq \sum_{i=0}^{\infty} \|\boldsymbol{f}\|_p \, \|G\|_p^i = \frac{\|\boldsymbol{f}\|_p}{1 - \|G\|_p}$$

Zbieżność w metodzie SOR

$$G_{SOR} = (D + \omega L)^{-1}[-\omega U + (1 - \omega)D]$$

$$\det(G_{SOR}) = \det\left((D + \omega L)^{-1}\right) \det\left(-\omega U + (1 - \omega)D\right)$$

$$\det\left((D + \omega L)^{-1}\right) = \frac{1}{\det(D + \omega L)} = \frac{1}{\det(D)}$$

$$\det\left(-\omega U + (1 - \omega)D\right) = \det((1 - \omega)D) = (1 - \omega)^n \det(D)$$

$$\det(G_{SOR}) = (1 - \omega)^n$$

$$\det(G_{SOR}) = \lambda_1 \lambda_2 \ldots \lambda_n$$

$$|1 - \omega| \leq \max_{i=1,\ldots,n} \lambda_i = \rho(G_{SOR}) < 1$$

- jeśli macierz układu jest symetryczna, dodatniookreślona i nieosobliwa to procedura iteracyjna jest zawsze zbieżna dla

$$0 < \omega < 2$$

Minimalizacja formy kwadratowej

Jeśli A**x**=**b** i **r**=**b**-A**x** to możemy utworzyć formę kwadratową postaci

$$R = \boldsymbol{r}^T \boldsymbol{r} = (\boldsymbol{b} - A\boldsymbol{x})^T (\boldsymbol{b} - A\boldsymbol{x})$$

która jest dodatniookreślona i przyjmuje wartość minimalną dla dokładnego rozwiązania **x**.

W dalszych rozważaniach zakładamy, że **macierz A jest symetryczna i dodatniookreślona**, wówczas możemy użyć formy kwadratowej postaci

$$Q = \frac{1}{2}\boldsymbol{x}^T A\boldsymbol{x} - \boldsymbol{x}^T \boldsymbol{b}$$

która ma minimum w **x**, ponieważ

$$dQ = Q(\boldsymbol{x} \pm \Delta\boldsymbol{x}) - Q(\boldsymbol{x}) = \frac{1}{2}\Delta\boldsymbol{x}^T A \Delta\boldsymbol{x} > 0$$

Proces poszukiwania rozwiązania dokładnego przebiega iteracyjnie, tj. szukamy ciągu przybliżeń

$$\boldsymbol{x}_0 \rightarrow \boldsymbol{x}_1 \rightarrow \boldsymbol{x}_3 \rightarrow \ldots \rightarrow \boldsymbol{x}_n$$

gdzie:

$$\boldsymbol{x}_{i+1} = \boldsymbol{x}_i + \alpha_i \boldsymbol{v}_i$$

Od sposobu wyznaczania α_i i $\mathbf{v}_i$ zależy zbieżność i szybkość metody.

Metoda największego spadku

Przybliżone rozwiązanie w i+1 iteracji ma postać

$$\boldsymbol{x}_{i+1} = \boldsymbol{x}_i + \alpha_i \boldsymbol{v}_i$$

Jako v_i wybieramy kierunek gradientu Q

$$\nabla Q = A\boldsymbol{x}_i - \boldsymbol{b} = -\boldsymbol{r}_i \quad \Rightarrow \quad \boldsymbol{v}_i = -\boldsymbol{r}_i$$

W celu znalezienia współczynnika α_i obliczamy $Q(x_{i+1})$

$$Q(\boldsymbol{x}_i - \alpha_i \boldsymbol{r}_i) = -\frac{1}{2}\boldsymbol{x}_i^T \boldsymbol{r} - \frac{1}{2}\boldsymbol{x}_i^T \boldsymbol{b} + \frac{1}{2}\alpha_i^2 \boldsymbol{r}_i^T A \boldsymbol{r}_i + \alpha_i \boldsymbol{r}_i^T \boldsymbol{r}_i$$

i różniczkujemy je po parametrze wariacyjnym w celu znalezienia minimum

$$\frac{\partial Q}{\partial \alpha_i} = \boldsymbol{r}_i^T \boldsymbol{r}_i + \alpha_i \boldsymbol{r}_i^T A \boldsymbol{r}_i$$

$$\frac{\partial Q}{\partial \alpha_i} = 0 \quad \Rightarrow \quad \alpha_i = -\frac{\boldsymbol{r}_i^T \boldsymbol{r}_i}{\boldsymbol{r}_i^T A \boldsymbol{r}_i}$$

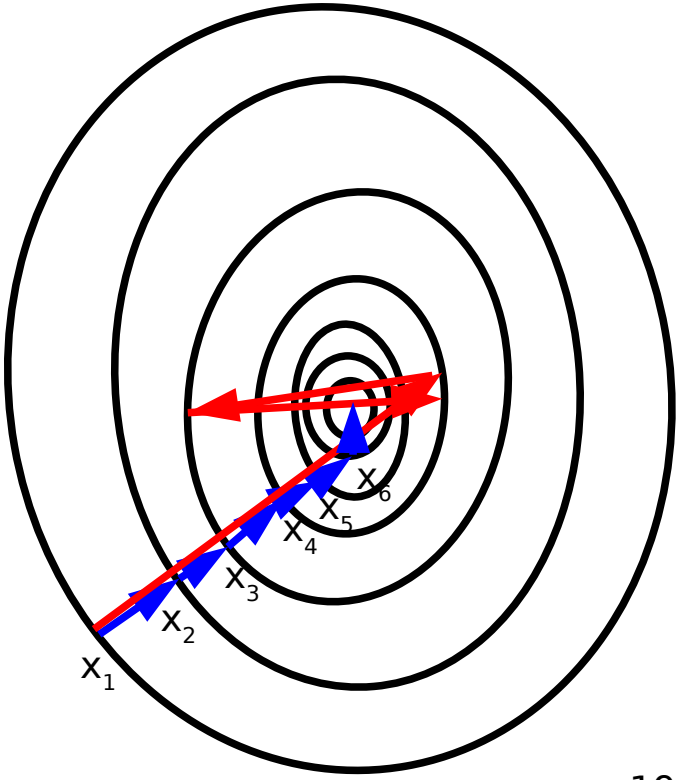
Kolejne przybliżenie w metodzie największego spadku opisuje wyrażenie

$$\boldsymbol{x}_{i+1} = \boldsymbol{x}_i + \frac{\boldsymbol{r}_i^T \boldsymbol{r}_i}{\boldsymbol{r}_i^T A \boldsymbol{r}_i} \boldsymbol{r}_i$$

dla którego zachodzi warunek (teoretycznie)

$$Q(\boldsymbol{x}_{i+1}) < Q(\boldsymbol{x}_i)$$

- związek gradientu Q z kierunkiem poszukiwania przybliżonego rozwiązania
- prosta interpretacja geometryczna w 2D – powierzchnie o stałej wartości Q mają kształt elipsy (hiperelipsy dla większej liczby wymiarów)
- metoda może być wolnozbieżna w przypadku, gdy hiperelipsoida ma wydłużony kształt, co odpowiada złemu uwarunkowaniu układu



Metoda sprzężonego gradientu

Założenia:

- x_d jest rozwiązaniem dokładnym
- ciąg wektorów

$$\{\boldsymbol{y}_1, \boldsymbol{y}_2, \boldsymbol{y}_3, \ldots, \boldsymbol{y}_n\}$$

stanowi bazę w n-wymiarowej przestrzeni euklidesowej

warunek ortogonalności bazy

$$\boldsymbol{y}_i^T \boldsymbol{y}_j = \begin{cases} = 0 & \iff \quad i \neq j \\ \neq 0 & \iff \quad i = j \end{cases}$$

$$\boldsymbol{y}_i^T \boldsymbol{y}_j = \delta_{i,j} \boldsymbol{y}_i^T \boldsymbol{y}_j$$

Różnicę rozwiązania dokładnego i przybliżonego możemy zapisać w postaci kombinacji liniowej elementów bazy

$$\Delta \boldsymbol{x} = \boldsymbol{x}_d - \boldsymbol{x}_i = \sum_{j=1}^{n} \alpha_j \boldsymbol{y}_j$$

Jeśli elementy bazy są ortogonalne to można łatwo wyznaczyć współczynniki kombinacji liniowej

$$\boldsymbol{y}_m^T (\boldsymbol{x}_d - \boldsymbol{x}_i) = \sum_{j=1}^{n} \alpha_j \boldsymbol{y}_m^T \boldsymbol{y}_j = \sum_{j=1}^{n} \alpha_j \delta_{m,j} \boldsymbol{y}_m^T \boldsymbol{y}_j = \alpha_m \boldsymbol{y}_m^T \boldsymbol{y}_m$$

$$\alpha_m = \frac{\boldsymbol{y}_m^T (\boldsymbol{x}_d - \boldsymbol{x}_i)}{\boldsymbol{y}_m^T \boldsymbol{y}_m}, \quad j = 1, 2, \ldots$$

Dostaliśmy wynik (poprzedni slajd)

$$\alpha_m = \frac{\boldsymbol{y}_m^T(\boldsymbol{x}_d - \boldsymbol{x}_i)}{\boldsymbol{y}_m^T \boldsymbol{y}_m}, \quad j = 1, 2, \ldots$$

$$\boldsymbol{y}_i^T \boldsymbol{y}_j = \delta_{i,j} \boldsymbol{y}_i^T \boldsymbol{y}_i$$

Ale powyższy wzór wymaga modyfikacji, ponieważ nie znamy wektora $\mathbf{x_d}$, wiemy jednak, że

$$A\boldsymbol{x}_d = \boldsymbol{b}$$

policzmy ponownie współczynnik α, ale użyjmy nowej bazy A-ortogonalnej

$$\{\boldsymbol{v}_1, \boldsymbol{v}_2, \boldsymbol{v}_3, \ldots, \boldsymbol{v}_n\} \iff$$

warunek A-ortogonalności bazy

$$\boldsymbol{v}_i^T A \boldsymbol{v}_j = \delta_{i,j} \boldsymbol{v}_i^T A \boldsymbol{v}_i$$

$$A(\boldsymbol{x}_d - \boldsymbol{x}_i) = \sum_{j=1}^{n} \alpha_j A \boldsymbol{v}_j$$

$$\boldsymbol{v}_m^T A(\boldsymbol{x}_d - \boldsymbol{x}_i) = \sum_{j=1}^{n} \alpha_j \boldsymbol{v}_m^T A \boldsymbol{v}_j = \sum_{j=1}^{n} \alpha_j \delta_{m,j} \boldsymbol{v}_m^T A \boldsymbol{v}_j = \alpha_m \boldsymbol{v}_m^T A \boldsymbol{v}_m$$

$$\alpha_j = \frac{\boldsymbol{v}_j^T A(\boldsymbol{x}_d - \boldsymbol{x}_i)}{\boldsymbol{v}_j^T A \boldsymbol{v}_j} = \frac{\boldsymbol{v}_j^T (\boldsymbol{b} - A\boldsymbol{x}_i)}{\boldsymbol{v}_j^T A \boldsymbol{v}_j} = \frac{\boldsymbol{v}_j^T \boldsymbol{r}_i}{\boldsymbol{v}_j^T A \boldsymbol{v}_j}$$

Żądamy więc, aby wektory bazy spełniały warunek A-ortogonalności (wektory A-sprzężone)

$$\boldsymbol{v}_j^T A \boldsymbol{v}_i = 0 \leftrightarrow i \neq j$$

Dla macierzy dodatniookreślonej zachodzi warunek

$$\boldsymbol{v}_i^T A \boldsymbol{v}_i \neq 0$$

Jak skonstruować bazę A-ortogonalną?

Jeśli dysponujemy zwykłą bazą wektorów

$$\boldsymbol{u}_1, \boldsymbol{u}_2, \boldsymbol{u}_3, \ldots$$

to możemy ją poddać procesowi **ortogonalizacji Grama-Schmidta**

$$\begin{aligned} \boldsymbol{v}_1 &= \boldsymbol{u}_1 \\ \boldsymbol{v}_{i+1} &= \boldsymbol{u}_{i+1} - \sum_{k=1}^{i} \beta_{i+1,k} \boldsymbol{v}_k \\ \beta_{i+1,k} &= \frac{\boldsymbol{v}_k^T A \boldsymbol{u}_{i+1}}{\boldsymbol{v}_k^T A \boldsymbol{v}_k} \end{aligned}$$

Jak utworzyć ciąg wektorów $\mathbf{u_i}$? W metodzie CG bazę stanowią wektory reszt (kierunki gradientów), które dzięki A-ortogonalizacji są sprzężone.

Kolejne przybliżenia w podstawowej metodzie CG wyznaczamy zgodnie z poniższym schematem:

$$
\begin{aligned}
\boldsymbol{v}_1 &= \boldsymbol{r}_1 = \boldsymbol{b} - A\boldsymbol{x}_1 \\
- &- \quad - - - - - - - \\
\alpha_i &= \frac{\boldsymbol{v}_i^T \boldsymbol{r}_i}{\boldsymbol{v}_i^T A \boldsymbol{v}_i} \\
\boldsymbol{x}_{i+1} &= \boldsymbol{x}_i + \alpha_i \boldsymbol{v}_i \\
\boldsymbol{r}_{i+1} &= \boldsymbol{r}_i - \alpha_i A \boldsymbol{v}_i \\
\beta_i &= \frac{\boldsymbol{v}_i^T A \boldsymbol{r}_{i+1}}{\boldsymbol{v}_i^T A \boldsymbol{v}_i} \\
\boldsymbol{v}_{i+1} &= \boldsymbol{r}_{i+1} - \beta_i \boldsymbol{v}_i
\end{aligned}
$$

Dzięki A-ortogonalności w każdej iteracji wystarczy wyznaczyć tylko jeden współczynnik β (reszta współczynników znika).

W podstawowej metodzie CG w każdej iteracji należy wykonać dwa mnożenia macierz-wektor

Av_i Ar_{i+1}

i to te dwie operacje determinują nakład obliczeń. Algorytm metody CG można przedstawić w alternatywnej postaci, gdzie wymagamy tylko jednego mnożenia macierz-wektor:

$$
\begin{aligned}
\boldsymbol{v}_1 &= \boldsymbol{r}_1 = \boldsymbol{b} - A\boldsymbol{x}_1 \\
- &- \quad - - - - - - - \\
\alpha_i &= \frac{\boldsymbol{r}_i^T \boldsymbol{r}_i}{\boldsymbol{v}_i A \boldsymbol{v}_i} \\
\boldsymbol{x}_{i+1} &= \boldsymbol{x}_i + \alpha_i \boldsymbol{v}_i \\
\boldsymbol{r}_{i+1} &= \boldsymbol{r}_i - \alpha_i A \boldsymbol{v}_i \\
\beta_i &= \frac{\boldsymbol{r}_{i+1}^T \boldsymbol{r}_{i+1}}{\boldsymbol{r}_i^T \boldsymbol{r}_i} \\
\boldsymbol{v}_{i+1} &= \boldsymbol{r}_{i+1} - \beta_i \boldsymbol{v}_i
\end{aligned}
$$

Maksymalna liczba iteracji w metodzie CG wynosi n+1 – więc jest metodą skończoną. Zazwyczaj do uzyskania akceptowalnego rozwiązania wystarcza wykonanie znacznie mniejszej liczby iteracji.

- rozważyliśmy tylko szczególny przypadek: macierz A symetryczna i dodatniookreślona
- jeśli macierz jest niesymetryczna konieczne jest użycie innej metody
- metody te wykorzystują podprzestrzeń Kryłowa
  (różni je sposób generowania ciągu wektorów $\mathbf{v}_i$)

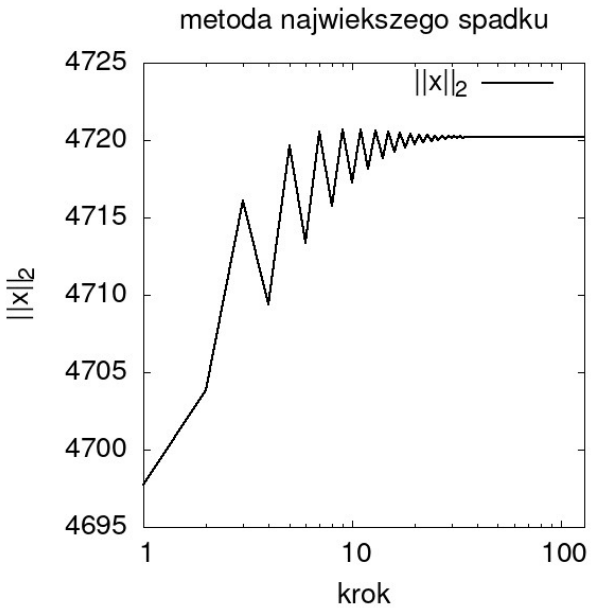
Inne metody o dużej wydajności:

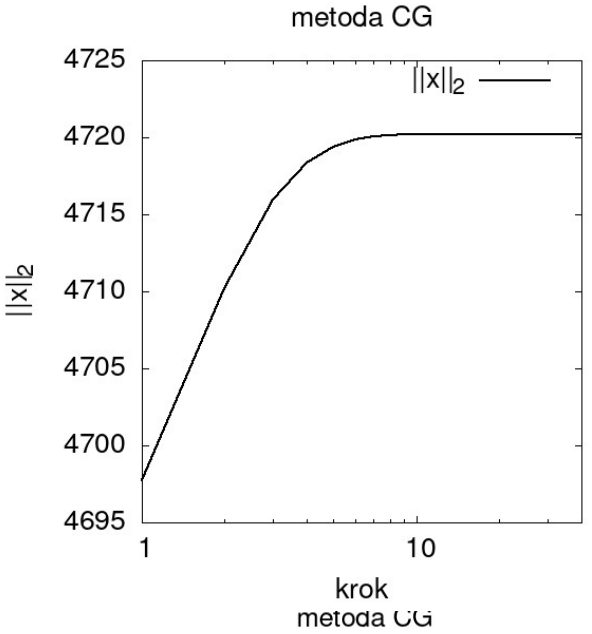
- Conjugate Gradient Squared (CGS)
- Bi-Conjugate Gradient Stabilized (BiCGStab)
- General Minimal Residual Method (GMRES)
- Transpose-Free Quasi-Minimal Residual (TFQMR)

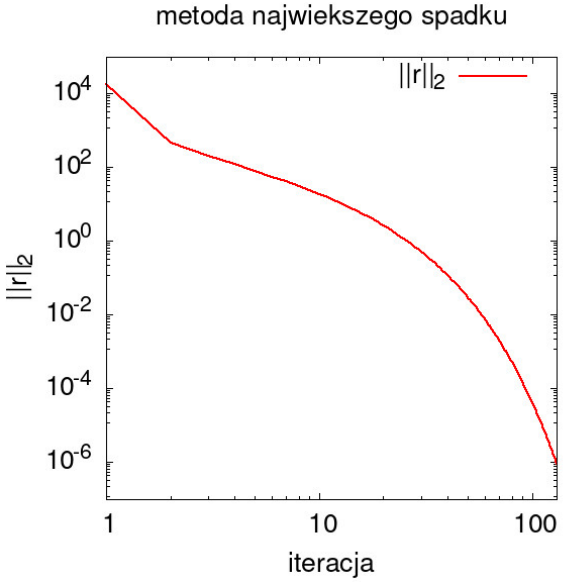
- w zasadzie nie ma gwarancji, że uzyskamy rozwiązanie układu równań
  w metodzie iteracyjnej (a przyczyną są oczywiście błędy umeryczne)
- dodatkowo, aby zwiększyć wydajność metody wykorzystuje się często **preconditioning**
  (zamiast macierzy A używa się macierzy „bliskiej” jej odwrotności np. ILUT).

Macierz A_{nxn}

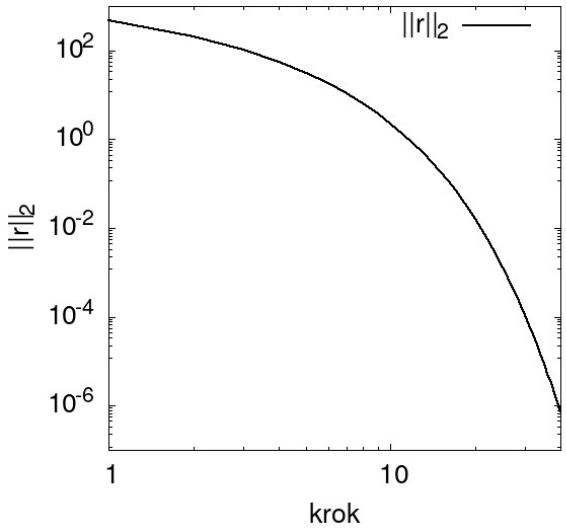
$$a_{ij} = \frac{1}{1+|i-j|}, \quad |i-j| \le 5 \quad \vee \quad a_{ij} = 0, \quad |i-j| > 5 \qquad n = 1000$$







kierunki poszukiwań
w kolejnych iteracjach
są ortogonalne



kolejne kierunki poszukiwań
nie są ortogonalne