Układy Równań Liniowych - algorytmy iteracyjne

Metody Numeryczne

dr inż. Grzegorz Fotyga

Politechnika Gdańska
Wydział Elektroniki, Telekomunikacji i Informatyki
Katedra Inżynierii Mikrofalowej i Antenowej

23 marca 2018

Plan prezentacji

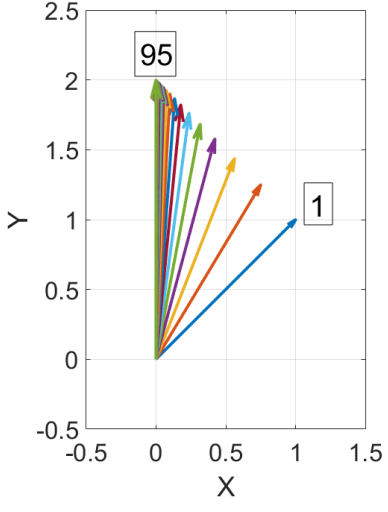
Algorytmy iteracyjne (1)

- Metody bezpośrednie (Gauss, LU) wymagają $\mathcal{O}(n^3)$ operacji. **Za dużo!**
- Metody iteracyjne - tylko $\mathcal{O}(n^2)$ [1].
- Algorytmy **iteracyjne** rozwiązywania układów równań liniowych $\mathbf{Ax} = \mathbf{b}$ rozpoczynają się od zdefiniowania wektora początkowego $\mathbf{x}^{(1)}$, który jest przybliżeniem rozwiązania dokładnego $\mathbf{x}$. Wraz z kolejnymi iteracjami algorytmu $\mathbf{x}^{(2)}$, $\mathbf{x}^{(3)}$ ... $\mathbf{x}^{(m)}$ wektor przybliżony zbiega się do rozwiązania dokładnego.
- Istnieją **setki** metod iteracyjnych, wykorzystywanych w praktyce. W zależności od rodzaju macierzy systemowych mogą się one nie zbiegać, mogą wymagać zbyt dużej liczby iteracji itp.
- Opanowanie wiedzy z tej dziedziny pozwala na dobranie odpowiedniej metody do danego problemu i ew. **poprawienie zbieżności** poprzez dobranie odpowiednich parametrów.
- Omówimy dwie podstawowe metody iteracyjne: **Jacobi'ego** i **Gaussa-Seidela.**

[1] pod warunkiem, że się zbiegają.

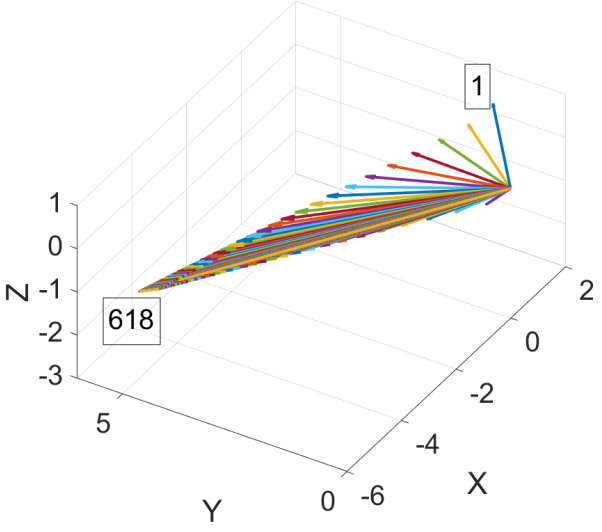
Algorytmy iteracyjne (2) - $\mathbb{R}^2$

- $\mathbf{A} = \begin{bmatrix} 2.0 & 1.5 \\ 1.5 & 2.0 \end{bmatrix}$
- $\mathbf{b} = \begin{bmatrix} 3 & 4 \end{bmatrix}^T$
- $\mathbf{x}^{(1)} = \begin{bmatrix} 1 & 1 \end{bmatrix}^T$
- $\mathbf{x}^{(95)} = \begin{bmatrix} 0.0 & 2.00 \end{bmatrix}^T$
- W 95 iteracji uzyskiwane jest rozwiązanie z *zadowalającą* dokładnością.



Algorytmy iteracyjne (3) - $\mathbb{R}^3$

- $\mathbf{A} = \begin{bmatrix} 1.5 & 1.1 & -0.3 \\ 1.1 & 1.5 & 1.1 \\ 0.3 & 1.1 & 2.0 \end{bmatrix}$
- $\mathbf{b} = \begin{bmatrix} 1 & 2 & 1 \end{bmatrix}^T$
- $\mathbf{x}^{(1)} = \begin{bmatrix} 1 & 1 & 1 \end{bmatrix}^T$
- $\mathbf{x}^{(618)} = \begin{bmatrix} -4.025 & 5.825 & -2.10 \end{bmatrix}^T$
- W 618 iteracji uzyskiwane jest rozwiązanie z *zadowalającą* dokładnością.
- **Podobnia** analiza dla przestrzeni $\mathbb{R}^n$.



Algorytmy iteracyjne (4) błędy

- Układ równań $\mathbf{Ax} = \mathbf{b}$, $\mathbf{A} \in \mathbb{R}^{n \times n}$, $\mathbf{b} \in \mathbb{R}^n, \mathbf{x} \in \mathbb{R}^n$
- Rozwiązanie **przybliżone**: $\widetilde{\mathbf{x}}$
- Błąd rozwiązania (również **wektor**): $\mathbf{e} = \mathbf{x} - \widetilde{\mathbf{x}}$
- W praktyce **nie jest możliwe** wyznaczenia wektora $\mathbf{e}$, ponieważ nie znamy rozwiązania dokładnego ($\mathbf{x}$)
- W celu oszacowania błędu wnoszonego przez $\widetilde{\mathbf{x}}$ stosuje się tzw. wektor residuum (z języka Łacińskiego: *reszta*):

$$\mathbf{r} = \mathbf{A}\widetilde{\mathbf{x}} - \mathbf{b} \tag{1}$$

- Jeżeli $\widetilde{\mathbf{x}} \to \mathbf{x}$ to $\mathbf{r} \to \mathbf{0}$.
- *W niektórych przypadkach mała wartość normy residuum nie gwarantuje małej wartości normy błędu* $\mathbf{e}$.

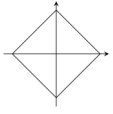
Algorytmy iteracyjne (5) - normy wektorów

- Wygodniej jest przedstawiać błąd w postaci **skalara**.
- *Norma* jest funkcją, która przyporządkowuje każdemu wektorowi nieujemną liczbę rzeczywistą ($\|\cdot\| : \mathbb{R}^n \to \mathbb{R}$)
- Norma wektora musi spełniać poniższe warunki (dla $\mathbf{e}, \mathbf{e}_1, \mathbf{e}_2 \in \mathbb{R}^n$, $\alpha \in \mathbb{R}$):
  - Norma wektora jest **nieujemną** liczbą rzeczywistą $\|\mathbf{e}\| \geqslant 0$, $\|\mathbf{e}\| = 0 \Leftrightarrow \mathbf{e} = 0$
  - Pomnożenie wektora przez liczbę mnoży jego normę przez wartość bezwzględną: $\|\alpha\mathbf{e}\| = |\alpha|\|\mathbf{e}\|$, $\alpha \in \mathbb{R}$
  - Norma sumy dwóch wektorów jest **nie większa** od sumy ich norm. Warunek ten nazywany jest **nierównością trójkąta**.$\|\mathbf{e}_1 + \mathbf{e}_2\| \leqslant \|\mathbf{e}_1\| + \|\mathbf{e}_2\|$,
- W praktyce często przedstawia się błąd aproksymacji $\widetilde{\mathbf{x}}$ jako normę z wektora residuum.

Algorytmy iteracyjne (6) - normy wektorów

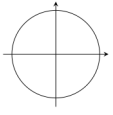
Najczęściej stosowane normy (w praktyce): p = 1,2,∞.
Okręgi jednostkowe: $\mathbf{e} \in \mathbb{R}^n : \|\mathbf{e}\| \leqslant 1$, $n = 2$ sa zobrazowane poniżej:

$$\|\mathbf{e}\|_1 = \sum_{j=1}^{n} |e_j|$$



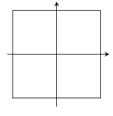
$p = 1$

$$\|\mathbf{e}\|_2 = \sqrt{\sum_{j=1}^{n} e_j^2}$$



$p = 2$

$$\|\mathbf{e}\|_\infty = \max_{1 \leqslant j \leqslant n} |e_j|$$



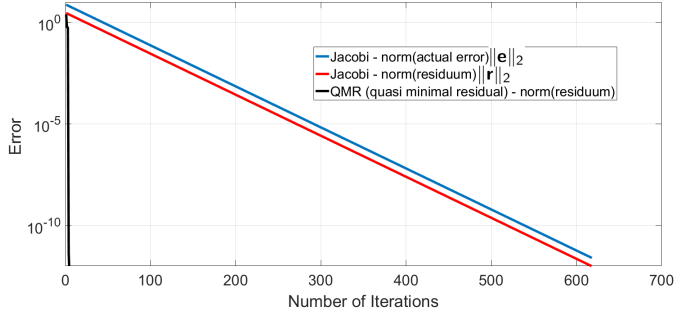
$p = \infty$

Algorytmy iteracyjne (7) - Sergel Plaza (Stockholm), Tables

Okrąg jednostkowy dla $p = 4$.

Algorytmy iteracyjne (8)

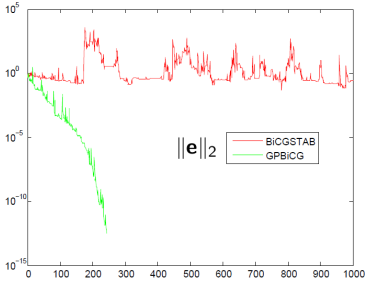
Badając normę $p = 2$ wektora residuum, możemy w każdej iteracji algorytmu obliczyć jaki błąd wnosi wektor $\widetilde{\mathbf{x}}$. Przeważnie jako kryterium stopu przyjmuje się normę z residuum o wartości mniejszej niż 10^{-6}.



- Jacobi, Gauss-Seidel, QMR - odpowiednio: 618, 269 i 4 (!) iteracje, żeby osiągnąć normę z residuum na poziomie 1e-12.
- QMR - metoda stworzona w 1991r. przez naukowców z MIT i CU Davis.

Algorytmy iteracyjne (9), dygresja - Teoplitz Matrix:

$$A = \begin{bmatrix} a_0 & a_{-1} & a_{-2} & \dots & \dots & a_{-(n-1)} \\ a_1 & a_0 & a_{-1} & \ddots & & \vdots \\ a_2 & a_1 & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & a_{-1} & a_{-2} \\ \vdots & & \ddots & a_1 & a_0 & a_{-1} \\ a_{n-1} & \dots & \dots & a_2 & a_1 & a_0 \end{bmatrix}$$



"Typical problems modelled by Toeplitz matrices are: the numerical solution of certain **differential equations**, and certain **integral equations** (regularization of inverse problems); the computation of **spline** functions; time **series** analysis; **signal** and **image processing**; **Markov** chains and queueing theory; polynomial and power series computations. Other problems involve Toeplitz-like matrices, or matrices having a displacement structure (Hankel, Cauchy, Hilbert, Loewner and Frobenius matrices)."

https://www.scirp.org/journal/PaperInformation.aspx?PaperID=18880
https://www.ercim.eu/publication/ErcimNews/enw22/toeplitz.html

Algorytmy iteracyjne (10) - Jacobi

- Rozpatrzmy układ 3 równań z 3 niewiadomymi ($\mathbf{Ax} = \mathbf{b}$) z **niezerowymi** elementami na diagonali:

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

- Powyższy układ możemy przedstawić jako:

$$\begin{cases} x_1 &= (b_1 - a_{12}x_2 - a_{13}x_3)/a_{11} \\ x_2 &= (b_2 - a_{21}x_1 - a_{23}x_3)/a_{22} \\ x_3 &= (b_3 - a_{31}x_1 - a_{32}x_2)/a_{33} \end{cases}$$

- Załóżmy, że $\mathbf{x}^{(k)}$ jest przybliżeniem dokładnego rozwiązania $\mathbf{x} = \mathbf{A}^{-1}\mathbf{b}$. Naturalnym sposobem na obliczenie kolejnego przybliżenia $\mathbf{x}^{(k+1)}$ jest:

$$\begin{cases} x_1^{(k+1)} &= (b_1 - a_{12}x_2^{(k)} - a_{13}x_3^{(k)})/a_{11} \\ x_2^{(k+1)} &= (b_2 - a_{21}x_1^{(k)} - a_{23}x_3^{(k)})/a_{22} \\ x_3^{(k+1)} &= (b_3 - a_{31}x_1^{(k)} - a_{32}x_2^{(k)})/a_{33} \end{cases} \quad (2)$$

Algorytmy iteracyjne (11)

- Równanie (2) definiuje schemat **Jacobi'ego** dla przypadku $n = 3$.
- W ogólności:

$$x_i^{(k+1)} = (b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k)} - \sum_{j=i+1}^{n} a_{ij} x_j^{(k)})/a_{ii} \tag{3}$$

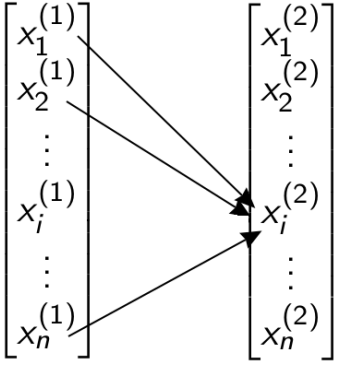
- Jednak w powyższym schemacie nie są uwzględnione aktualne wartości części elementów wektora. Na przykład: $x_1^{(k)}$ jest używane do obliczenia $x_2^{(k+1)}$, chociaż $x_1^{(k+1)}$ jest już znane.
- Jeżeli używamy **aktualnego** przybliżenia x_i, otrzymujemy:

$$x_i^{(k+1)} = (b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij} x_j^{(k)})/a_{ii} \tag{4}$$

- Jest to schemat **Gaussa-Seidela**.

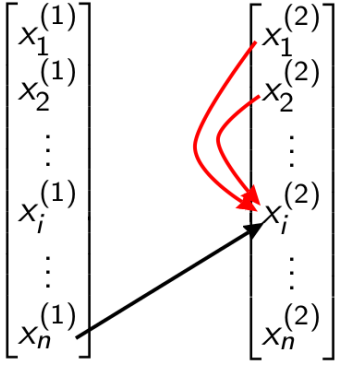
Iterative algorithms (12) - Jacobi idea

Jacobi idea - the vector from previous iteration is used to compute the elements of the vector in subsequent iteration ($x^{(k)}$ is used to compute $x^{(k+1)}$). Although, the $x_1^{(2)} \dots x_{i-1}^{(2)}$ are **already updated.**



Iterative algorithms (13) - Gauss-Seidel idea

Gauss-Seidel idea - use the **most recently available information** when computing $x_i^{(k+1)}$.



Algorytmy iteracyjne (14) A = - L - U + D

Podział macierzy **A**:

Uwaga L i **U** nie oznaczają tego samego, co **L** i **U** w faktoryzacji LU!

Algorytmy iteracyjne (15) - Jacobi

Jacobi w postaci macierzowej:

- Wyprowadzenie schematu iteracyjnego dla metody Jacobiego rozpoczynamy od przedstawienia macierzy $\mathbf{A}$ jako sumę 3 macierzy: $\mathbf{A} = -\mathbf{L} - \mathbf{U} + \mathbf{D}$, gdzie:
  - $\mathbf{D}$ - jest to macierz diagonalna, zawierająca elementy z głównej diagonali macierzy $\mathbf{A}$
  - $\mathbf{L}$ i $\mathbf{U}$ to są macierze - odpowiednio - trójkątna dolna i górna (zawierające elementy znajdujące się powyżej i poniżej głównej diagonali macierzy $\mathbf{A}$).
- Otrzymujemy:
  $(-\mathbf{L} - \mathbf{U} + \mathbf{D})\mathbf{x} = \mathbf{b}$
- Przenosząc odpowiednie wyrazy na prawą stronę równania:
  $\mathbf{D}\mathbf{x} = (\mathbf{L} + \mathbf{U})\mathbf{x} + \mathbf{b}$
- Przemnażając obustronnie przez $\mathbf{D}^{-1}$ otrzymujemy:
  $\mathbf{x} = \mathbf{D}^{-1}(\mathbf{L} + \mathbf{U})\mathbf{x} + \mathbf{D}^{-1}\mathbf{b}$
- Ostatecznie otrzymujemy schemat iteracyjny:
  $\widetilde{\mathbf{x}}^{(n+1)} = \mathbf{D}^{-1}(\mathbf{L} + \mathbf{U})\widetilde{\mathbf{x}}^{(n)} + \mathbf{D}^{-1}\mathbf{b}$
- **Uwaga** $\mathbf{L}$ i $\mathbf{U}$ nie oznaczają tego samego, co $\mathbf{L}$ i $\mathbf{U}$ w faktoryzacji LU!

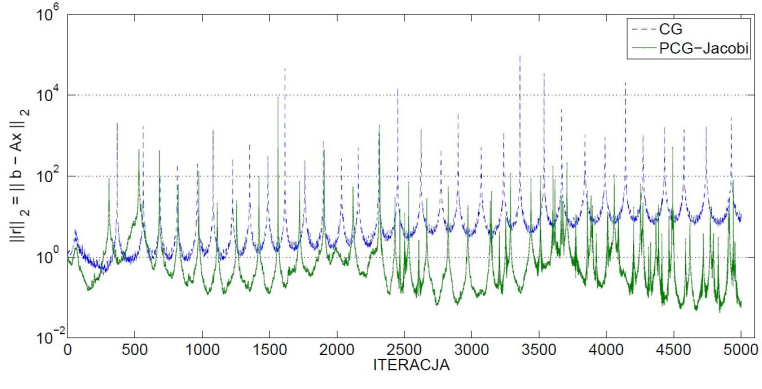
Algorytmy iteracyjne (16) - Gauss - Seidel

Gauss-Seidel w postaci macierzowej:

- Wyprowadzenie schematu iteracyjnego metody Gaussa-Seidla rozpoczynamy (podobnie jak w metodzie Jacobiego) od:
  $(-\mathbf{L} - \mathbf{U} + \mathbf{D})\mathbf{x} = \mathbf{b}$
- Przenosząc odpowiednie wyrazy na prawą stronę równania:
  $(\mathbf{D} - \mathbf{L})\mathbf{x} = \mathbf{U}\mathbf{x} + \mathbf{b}$
- Przemnażając obustronnie przez $(\mathbf{D} - \mathbf{L})^{-1}$ otrzymujemy:
  $\mathbf{x} = (\mathbf{D} - \mathbf{L})^{-1}(\mathbf{U}\mathbf{x}) + (\mathbf{D} - \mathbf{L})^{-1}\mathbf{b}$
- Ostatecznie otrzymujemy schemat iteracyjny:
  $\widetilde{\mathbf{x}}^{(n+1)} = (\mathbf{D} - \mathbf{L})^{-1}(\mathbf{U}\widetilde{\mathbf{x}}^{(n)}) + (\mathbf{D} - \mathbf{L})^{-1}\mathbf{b}$
- **Uwaga L** i **U** nie oznaczają tego samego, co **L** i **U** w faktoryzacji LU!

Algorytmy iteracyjne (17) - przykład

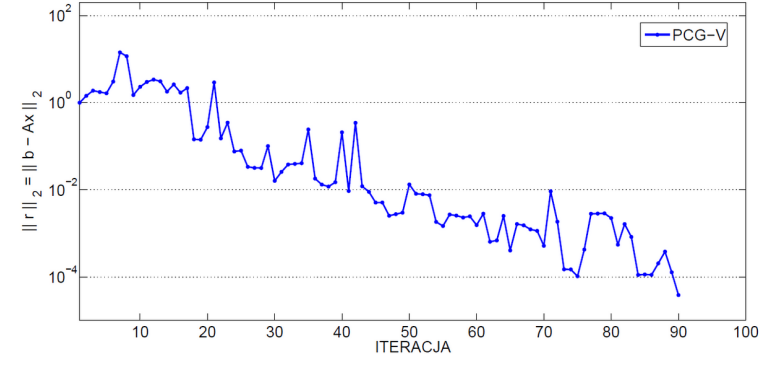
Filtr działający na b.w.cz. Dyskretyzacja równań Maxwella. CG i PCG-Jacobi w tym przypadku **nie zbiegają się**.



dr eng. Adam Dziekoński *Optymalizacja wydajności obliczeniowej metody elementów skończonych w architekturze CUDA*, PhD, 2015.

Algorytmy iteracyjne (18) - przykład

Filtr działający na b.w.cz. Dyskretyzacja równań Maxwella. PCG-V (z Jacobim) w tym przypadku **zbiega się**.



dr eng. Adam Dziekoński *Optymalizacja wydajności obliczeniowej metody elementów skończonych w architekturze CUDA*, PhD, 2015.

Algorytmy iteracyjne (19)

Uwagi na koniec:

- Zbieżność obu metod (oraz innych metod iteracyjnych) zależy od własności macierzy $\mathbf{A}$
- Gauss-Seidel zbiega się, jeżeli:
  - $\mathbf{A}$ jest symetryczna, dodatnio określona,
  - $\mathbf{A}$ jest diagonalnie dominująca ($|a_{ii}| > \sum_{j \neq i} |a_{ij}|$)
- Jakobi zbiega się, jeżeli:
  - promień spektralny macierzy $\mathbf{D}^{-1}(\mathbf{L} + \mathbf{U})$ jest mniejszy niż 1,
  - $\mathbf{A}$ jest diagonalnie dominująca.
- Metoda Gaussa-Seidla przeważnie potrzebuje mniejszej liczby iteracji, niż metoda Jacobiego, żeby zbiec się do założonego poziomu błędu.
- **Która metoda jest szybsza???**

Algorytmy iteracyjne (20)

Uwagi na koniec:

- Przykład macierzy diagonalnie dominującej (dyskretyzacja równań różniczkowych!):

$$\begin{bmatrix} 3 & -1 & 0 & 0 \\ -1 & 3 & -1 & 0 \\ 0 & -1 & 3 & -1 \\ 0 & 0 & -1 & 3 \end{bmatrix}$$

- **UWAGA** – dużym błędem jest jawne odwrócenie czynnika $\mathbf{D} - \mathbf{L}$, ponieważ operacja ta jest czasochłonna, powoduje duży błąd numeryczny i znaczny wzrost zapotrzebowania na pamięć RAM. Należy zastosować podstawienie wprzód (ang. forward substitutttion).
- Pojawiający się często we wzorach zapis inv($\mathbf{A}$)$\mathbf{b}$ lub $\mathbf{A}^{-1}\mathbf{b}$ oznacza rozwiązanie układu równań dowolną metodą (Gauss, Jacobi, CG, GMRES, BiCG, forward/backward substitttution...), czyli wyznaczenie wektora $\mathbf{x}$. Zapis ten **nigdy** nie oznacza jawnego odwrócenia macierzy $\mathbf{A}$.

Algorytmy iteracyjne (21) - Norma macierzy

1. **Norma macierzowa** musi spełniać poniższe warunki (dla $\mathbf{A}, \mathbf{B} \in \mathbb{R}^{n \times n}$, $\alpha \in \mathbb{R}$):
   - Norma macierzy jest **nieujemną** liczbą rzeczywistą $\|\mathbf{A}\| \geqslant 0$, $\|\mathbf{A}\| = 0 \Leftrightarrow \mathbf{A} = 0$
   - Pomnożenie macierzy przez liczbę mnoży jego normę przez wartość bezwzględną: $\|\alpha\mathbf{A}\| = |\alpha|\|\mathbf{A}\|$, $\alpha \in \mathbb{R}$
   - Norma sumy dwóch macierzy jest **nie większa** od sumy ich norm. Warunek ten nazywany jest **nierównością trójkąta**.$\|\mathbf{A} + \mathbf{B}\| \leqslant \|\mathbf{A}\| + \|\mathbf{B}\|$,
2. Norma **Frobeniusa**:

$$\|\mathbf{A}\|_F = \left(\sum_{i=1}^{m} \sum_{j=1}^{n} |a_{ij}|^2 \right)^{1/2} \tag{5}$$

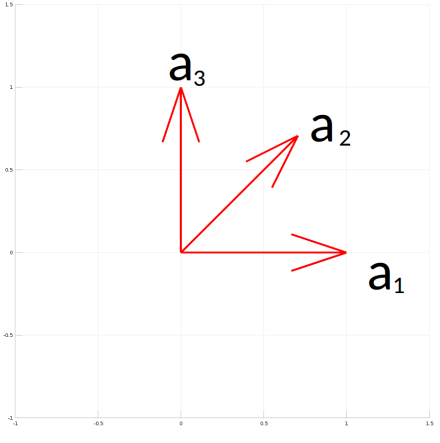
   Odpowiada wektorowej normie dla $p = 2$, jeżeli ułożymy kolumny (wiersze) macierzy $\mathbf{A}$ w jeden wektor.

Algorytmy iteracyjne (22) - indukowana norma macierzy

1. Rozpatrzmy macierz $\mathbf{A} \in \mathbb{R}^{2\times 2}$ i trzy wektory: $\mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3 \in \mathbb{R}^2$ o normie $\|\mathbf{a}_i\|_2 = 1$:
   - $\mathbf{A} = \begin{bmatrix} 1 & 4 \\ -3 & 2 \end{bmatrix}$
   - $\mathbf{a}_1 = \begin{bmatrix} 1 & 0 \end{bmatrix}^T$
   - $\mathbf{a}_2 = \begin{bmatrix} \sqrt{2}/2 & \sqrt{2}/2 \end{bmatrix}^T$
   - $\mathbf{a}_3 = \begin{bmatrix} 0 & 1 \end{bmatrix}^T$
2. Przemnażając wektory $\mathbf{x}_i$ przez macierz $\mathbf{A}$, otrzymujemy:
   - $\mathbf{b}_1 = \mathbf{A}\mathbf{x}_1 = \begin{bmatrix} 1 & -3 \end{bmatrix}^T$
   - $\mathbf{b}_2 = \mathbf{A}\mathbf{x}_2 = \begin{bmatrix} 3.53553 & -0.70711 \end{bmatrix}^T$
   - $\mathbf{b}_3 = \mathbf{A}\mathbf{x}_3 = \begin{bmatrix} 4 & 2 \end{bmatrix}^T$
3. Normy wektorów $\mathbf{y}_1$ wynoszą:
   - $\|\mathbf{b}_1\|_2 = 3.1623$
   - $\|\mathbf{b}_2\|_2 = 3.6056$
   - $\|\mathbf{b}_3\|_2 = 4.4721$

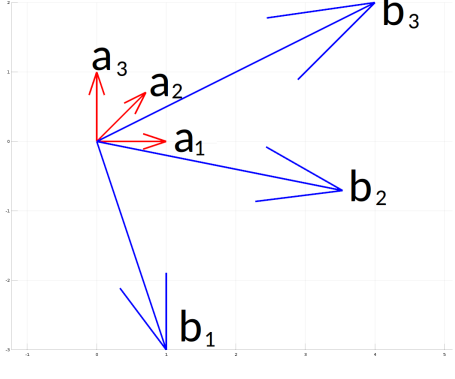
Algorytmy iteracyjne (23)

Trzy wektory: $\mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3 \in \mathbb{R}^2$ ($\|\mathbf{a}_i\|_2 = 1$)



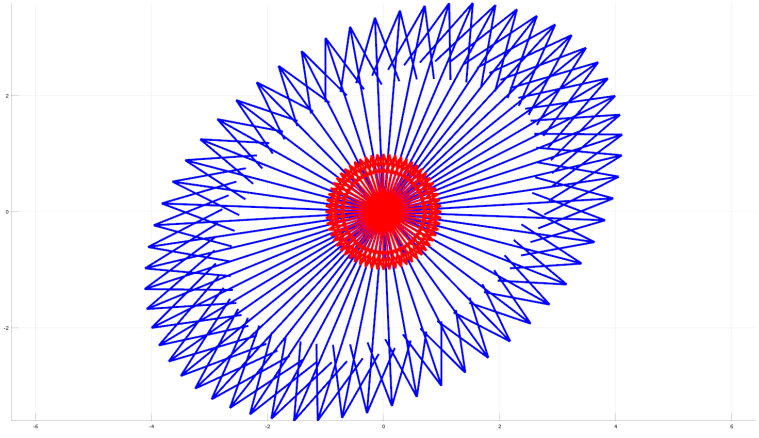
Algorytmy iteracyjne (24)

Wektory $\mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3$ przemnożone przez $\mathbf{A}$:



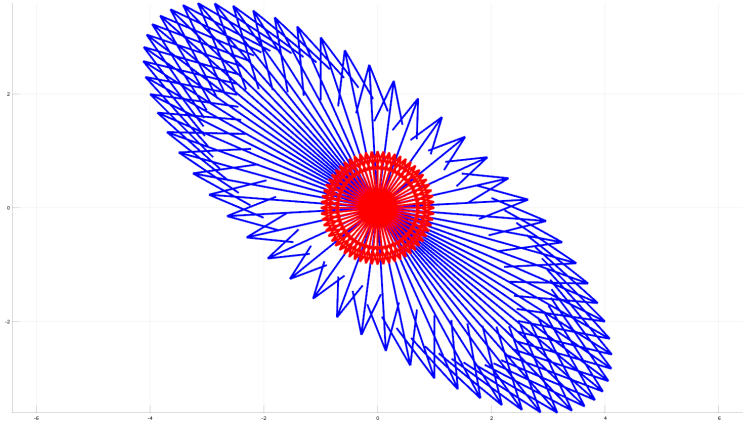
Algorytmy iteracyjne (23)

Działanie macierzy **A** na okrąg jednostkowy (z wykorzystaniem 2-normy):



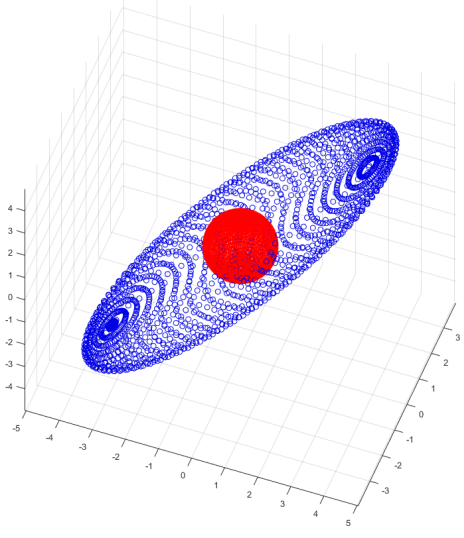
Algorytmy iteracyjne (24)

Działanie macierzy $\mathbf{B} = [1\ \ 4; -3\ \ -2]$ na okrąg jednostkowy (z wykorzystaniem 2-normy):



Algorytmy iteracyjne (25)

Działanie macierzy $\mathbf{C} \in \mathbb{R}^{3\times 3}$ na *kulę* jednostkową *(unit sphere)*



Algorytmy iteracyjne (25)

Indukowana norma macierzy

$$\|\mathbf{A}\|_2 = \sup_{\substack{\mathbf{x}\in\mathbb{R}^n \\ \mathbf{x}\neq 0}} \frac{\|\mathbf{A}\mathbf{x}\|_2}{\|\mathbf{x}\|_2} \tag{6}$$

$\|\mathbf{A}\|_2$ jest to supremum wartości $\|\mathbf{A}\mathbf{x}\|_2/\|\mathbf{x}\|_2$ dla wszystkich wektorów $\mathbf{x}$ z przestrzeni $\mathbb{R}^n$. Innymi słowy - jest to maksymalna wartość o jaką $\mathbf{A}$ może *rozciągnąć* wektor $\mathbf{x}$.

Algorytmy iteracyjne (26) - dla **AMBITNYCH**

Korelacja między wektorem residuum a błędem rzeczywistym:

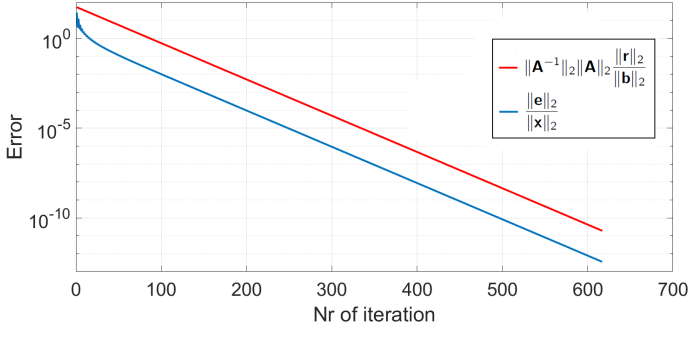
$$\frac{\|\mathbf{e}\|_2}{\|\mathbf{x}\|_2} \leqslant \|\mathbf{A}^{-1}\|_2 \|\mathbf{A}\|_2 \frac{\|\mathbf{r}\|_2}{\|\mathbf{b}\|_2} \tag{7}$$

gdzie $\kappa = \|\mathbf{A}^{-1}\|_2 \|\mathbf{A}\|_2$ jest współczynnikiem uwarunkowania macierzy (condition number of a matrix) $\mathbf{A}$ i ma wartość nie mniejszą, niż 1. Jeżeli znamy współczynnik uwarunkowania oraz residuum w danej iteracji, możemy z dużą dokładnością oszacować wartość normy wektora błędu $\mathbf{e}$.

Algorytmy iteracyjne (26) - dla **AMBITNYCH**

Korelacja między wektorem residuum a błędem rzeczywistym:

$$\frac{\|\mathbf{e}\|_2}{\|\mathbf{x}\|_2} \leqslant \|\mathbf{A}^{-1}\|_2 \|\mathbf{A}\|_2 \frac{\|\mathbf{r}\|_2}{\|\mathbf{b}\|_2} \tag{8}$$



Algorytmy iteracyjne (28)

Dziękuję