

Układy Równań Liniowych

Metody Numeryczne

dr inż. Grzegorz Fotyga

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

9 marca 2018

Plan prezentacji

- 1 Podstawowe definicje
- 2 Gauss
- 3 Faktoryzacja LU
- 4 Faktoryzacja choleskiego
- 5 Uwagi końcowe

Wstęp (1)

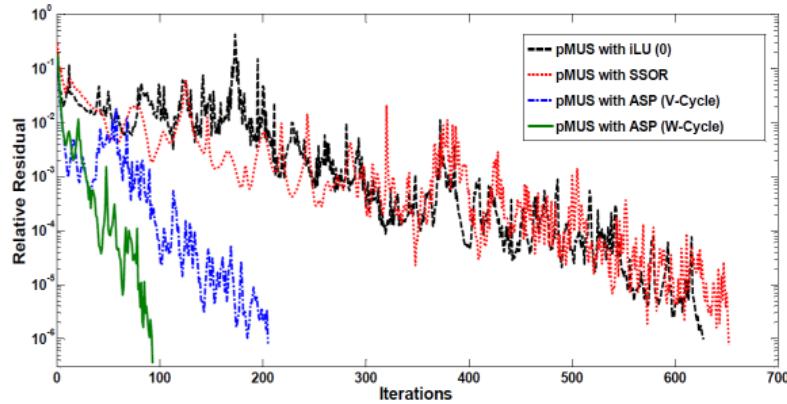
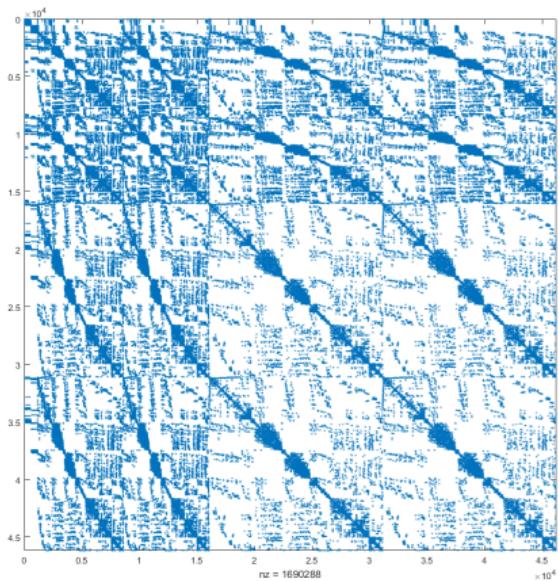


Fig. 2. Convergence history of COCG with different preconditioners for solving the conducting cube problem at p -adaptive step 3.

conjugate gradients (CG), Jacobi, weighted Jacobi, Gauss-Seidel, p -Type multiplicative Schwarz (pMUS), GPU, LU-factorization, algebraic multigrid, multilevel solver.

Wstęp (2)

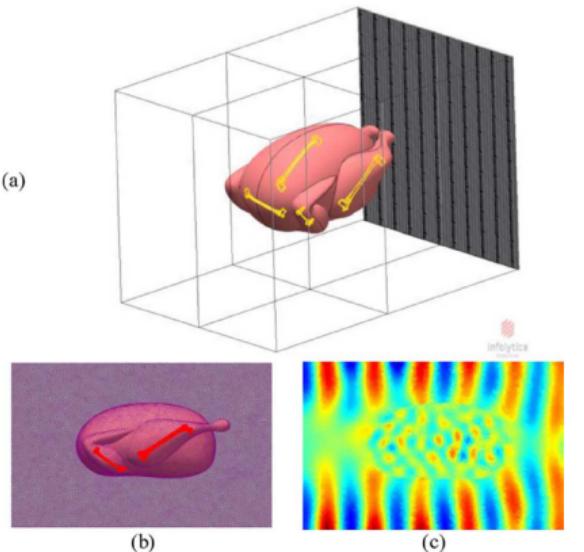


Fig. 3. (a) Illustration of the chicken in microwave oven and the domain partitioning applied. (b) Mesh discretization. (c) Visualization of field solution.

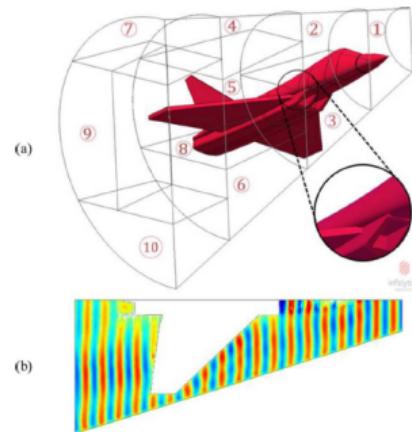


Fig. 6. (a) Decomposition scheme of the jet aircraft problem (b) Visualization of the computed field inside the computation domain.

An algebraic multigrid method for the finite element analysis of large scattering problems A Aghabarati, JP Webb - IEEE Transactions on Antennas and Propagation, 2013

Podstawowe definicje (1)

Klasyfikacja równań:

- równanie **liniowe** względem x - występuje w nim x podniesione jedynie do **pierwszej potęgi**:

$$y = ax + b \quad (1)$$

- równanie **algebraiczne** - w postaci $W(x) = 0$:

$$a_n x^n + a_{n-1} x^{n-1} + \dots + a_1 x + a_0 = 0 \quad (2)$$

gdzie $n \geq 0$, a_0, a_1, \dots, a_n są współczynnikami równania, a x niewiadomą,

- równanie **jednorodne** - współczynnik a_0 jest równy 0, np.:

$$2x^3 + 4x^2 + 9x = 0 \quad (3)$$

jest jednorodnym równaniem algebraicznym rzędu 3,

Podstawowe definicje (2)

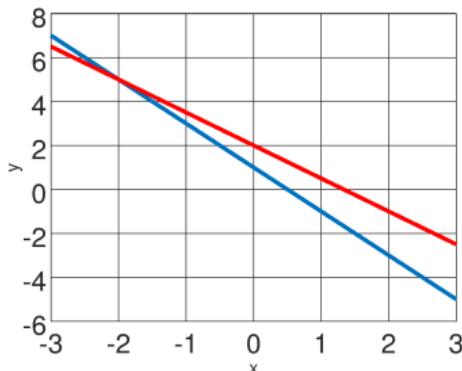
Układy równań mogą być **oznaczone, nieoznaczone i sprzeczne**.

- **Oznaczony** układ równań ma **jedno** rozwiązanie

$$3x + 2y = 4$$

$$2x + y = 1$$

- rozwiązuje powyższy układ równań otrzymujemy jednoznaczne rozwiązanie $(x, y) = (-2, 5)$. Proste przecinają się w **jednym** punkcie.



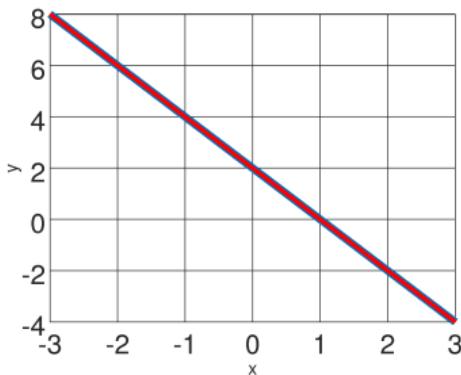
Podstawowe definicje (3)

- **Nieoznaczony** układ równań ma nieskończenie wiele rozwiązań

$$2x + y = 2$$

$$4x + 2y = 4$$

- Proste przecinają się w **nieskończonym** wielu punktach.



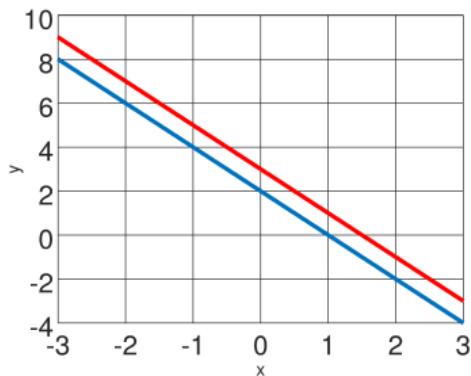
Podstawowe definicje (4)

- **Sprzeczny** układ równań nie ma rozwiązań

$$2x + y = 2$$

$$2x + y = 3$$

- Proste **nie** przecinają się.



Podstawowe definicje (5)

Powyższe założenia odnoszą się do rzeczywistych problemów, które mogą składać się z **milionów** równań (niewiadomych). Układ m równań **liniowych**, w którym występuje n **niewiadomych** ma następującą postać:

$$\left\{ \begin{array}{l} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n = b_2 \\ \vdots \qquad \vdots \qquad \ddots \qquad \vdots \qquad \vdots \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n = b_m \end{array} \right.$$

Gdzie:

- a_{ij} są współczynnikami, i jest numerem **wiersza**, j numerem **kolumny**,
- x_1, x_2, \dots, x_n są **niewiadomymi**
- b_1, b_2, \dots, b_m są **wyrazami wolnymi**
- $m, n \in \mathbb{N}$,
- $a_{ij}, x_i, b_j \in \mathbb{R}$,
- $i, j \in \mathbb{N}, 1 \leq i \leq m, 1 \leq j \leq n$

Podstawowe definicje (6)

Częściej spotykana forma zapisu:

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix}$$

$\mathbf{Ax} = \mathbf{b}$ (4)

- **A** - macierz (podstawowa, główna układu)
- **x** - wektor niewiadomych
- **b** - wektor wyrazów wolnych. Jeżeli **b** = **0** - układ **jednorodny**. W przeciwnym przypadku - **niejednorodny**.

Rozwiązywanie układu równań liniowych polega na wyznaczeniu wektora

$x = [x_1, x_2, \dots, x_n]$ liczb rzeczywistych spełniających ten układ, czyli takich, że dla każdego i ($1 \leq i \leq m$) zachodzi:

$$a_{i1}x_1 + a_{i2}x_2 + \cdots + a_{in}x_n = b_i \quad (5)$$

Podstawowe definicje (7)

Dyskusja na temat liczby rozwiązań liniowego układu równań:

- Twierdzenie Kroneckera – Capellego (wersja krótka) Układ równań $\mathbf{Ax} = \mathbf{b}$, gdzie $\mathbf{A} \in \mathbb{R}^{m \times n}$, $\mathbf{x} \in \mathbb{R}^n$, $\mathbf{b} \in \mathbb{R}^m$ posiada:
 - 1 rozwiązanie, gdy $r(\mathbf{A}) = r([\mathbf{A} \ \mathbf{b}]) = n$
 - nieskończenie wiele rozwiązań, gdy $r(\mathbf{A}) = r([\mathbf{A} \ \mathbf{b}]) < n$
 - nie posiada rozwiązań, gdy $r(\mathbf{A}) \neq r([\mathbf{A} \ \mathbf{b}])$
- gdzie $r(\mathbf{A})$ jest rzędem macierzy \mathbf{A} : rządem macierzy jest to maksymalna liczba liniowo niezależnych wektorów tworzących kolumny danej macierzy.
- Operacje elementarne nie zmieniają rzędu macierzy, $r(\mathbf{A}) = r(\mathbf{A}^T)$

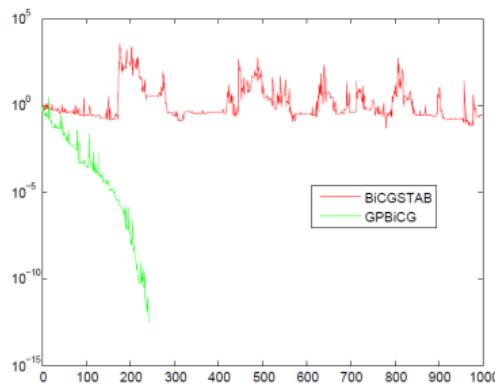
Podstawowe definicje (8)

Metody rozwiązyania układów równań liniowych

- **bezpośrednie** - pozwalają na uzyskanie wyniku po określonej liczbie operacji (działań arytmetycznych)
 - Cramera
 - **Gaussa**
 - Jordana
 - faktoryzacja LU
- **iteracyjne** - Polegają na wyznaczeniu ciągu kolejnych rozwiązań przybliżonych ($x_0, x_1, x_2 \dots x_n$), który jest zbieżny do dokładnego rozwiązyania układu równań liniowych.
 - Metoda iteracji prostych - **Jacobiego**
 - iteracyjna **Gaussa-Seidela**
 - **CG?**

Metody iteracyjne, dygresja - Macierz Toeplitza:

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

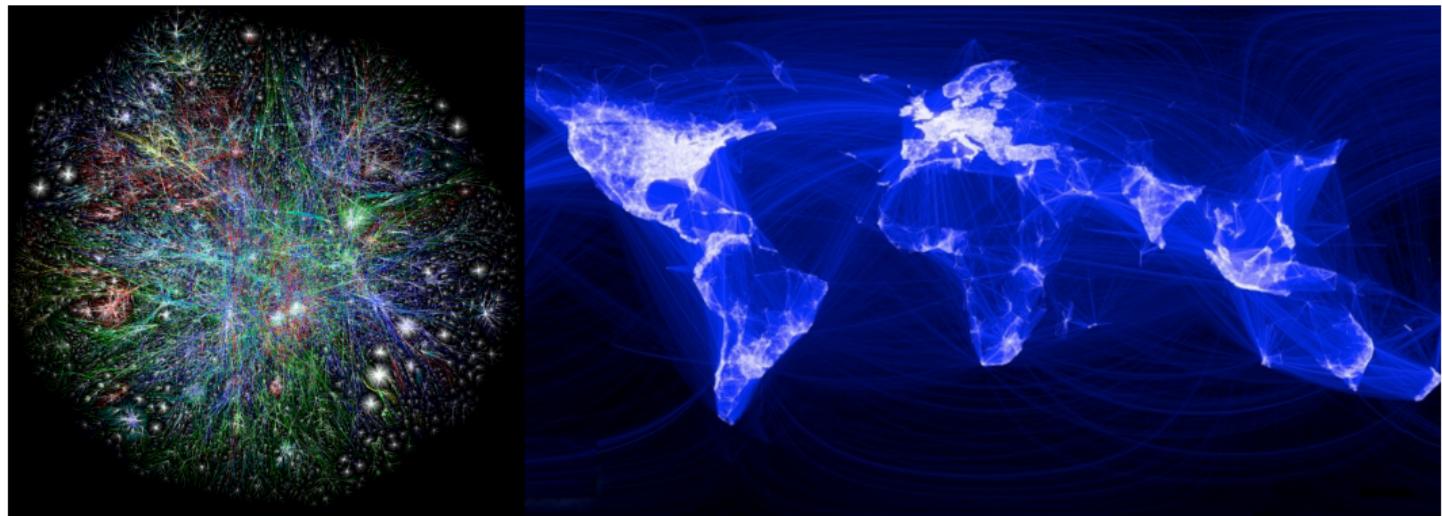


"Typical problems modelled by Toeplitz matrices are: the numerical solution of certain **differential equations**, and certain **integral equations** (regularization of inverse problems); the computation of **spline** functions; time **series** analysis; **signal** and **image processing**; **Markov** 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>

Dygresja



<http://mpictcenter.blogspot.com/2012/09/could-someone-really-destroy-whole.html>

Metoda Eliminacji Gaussa (1)

Metoda Eliminacji Gaussa

- ① Metoda **bezpośredniego** rozwiązywania układów równań liniowych
- ② n - równań, n - niewiadomych

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}$$

- ③ Rozwiązanie składa się z dwóch etapów: sprowadzenie układu do postaci trójkątnej, a następnie tzw. *postępowanie odwrotne*.

Metoda Eliminacji Gaussa (2)

Metoda eliminacji Gaussa - etap **pierwszy**:

- Trójkątny układ równań otrzymuje się dzięki następującym operacjom **elementarnym** wykonywanym na tzw. macierzy **rozszerzonej** $[\mathbf{A} \; \mathbf{b}]$:
 - Zamiana** miejscami dwóch wierszy macierzy: $W_i \leftrightarrow W_j$,
 - Pomnożenie** wiersza przez skalar: $W_i \leftarrow \alpha W_i$, dla $\alpha \neq 0$,
 - Dodanie** do danego wiersza wielokrotności innego wiersza $W_i \leftarrow W_i + \alpha W_j$, dla $\alpha \neq 0$.
- Powyzsze operacje nie wpływają na wartości x_i i ich pozycje w wektorze.**
- Zaczynamy od początkowego układu równań $\mathbf{A}^{(1)}\mathbf{x} = \mathbf{b}^{(1)}$:

$$\begin{bmatrix} a_{11}^{(1)} & a_{12}^{(1)} & \cdots & a_{1n}^{(1)} \\ a_{21}^{(1)} & a_{22}^{(1)} & \cdots & a_{2n}^{(1)} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1}^{(1)} & a_{n2}^{(1)} & \cdots & a_{nn}^{(1)} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1^{(1)} \\ b_2^{(1)} \\ \vdots \\ b_n^{(1)} \end{bmatrix}$$

Metoda Eliminacji Gaussa (3)

- Następnie odejmujemy od i -tego wiersza wiersz pierwszy, przemnożony przez $\alpha = a_{i1}^{(1)} / a_{11}^{(1)}$. Otrzymujemy $\mathbf{A}^{(2)}\mathbf{x} = \mathbf{b}^{(2)}$:

$$\begin{bmatrix} a_{11}^{(1)} & a_{12}^{(1)} & \cdots & a_{1n}^{(1)} \\ 0 & a_{22}^{(2)} & \cdots & a_{2n}^{(2)} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & a_{n2}^{(2)} & \cdots & a_{nn}^{(2)} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1^{(1)} \\ b_2^{(2)} \\ \vdots \\ b_n^{(2)} \end{bmatrix}$$

- gdzie $a_{22}^{(2)} = a_{22}^{(1)} - a_{12}^{(1)} \cdot \alpha$, $b_2^{(2)} = b_2^{(1)} - b_1^{(1)} \cdot \alpha$ itd.
- Postępujemy w ten sposób z kolejnymi wierszami. Otrzymujemy $\mathbf{A}^{(3)}\mathbf{x} = \mathbf{b}^{(3)}$, $\mathbf{A}^{(4)}\mathbf{x} = \mathbf{b}^{(4)}$ Ostatecznie:

$$\begin{bmatrix} a_{11}^{(1)} & a_{12}^{(1)} & \cdots & a_{1n}^{(1)} \\ 0 & a_{22}^{(2)} & \cdots & a_{2n}^{(2)} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & a_{nn}^{(n)} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1^{(1)} \\ b_2^{(2)} \\ \vdots \\ b_n^{(n)} \end{bmatrix}$$

Metoda Eliminacji Gaussa (4)

Metoda eliminacji Gaussa - etap **drugi**:

- **Trójkątny** układ równań:

$$\begin{bmatrix} a_{11}^{(1)} & a_{12}^{(1)} & \cdots & a_{1n}^{(1)} \\ 0 & a_{22}^{(2)} & \cdots & a_{2n}^{(2)} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & a_{nn}^{(n)} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1^{(1)} \\ b_2^{(2)} \\ \vdots \\ b_n^{(n)} \end{bmatrix}$$

- Rozwiązuje się wykorzystując małą liczbę działań arytmetycznych, korzystając z zależności rekurencyjnej:

- $x_n = b_n^{(n)} / a_{nn}^{(n)}$
- $x_i = \frac{b_i^{(i)} - a_{i1}^{(i)}x_1 - \cdots - a_{i,n-1}^{(i)}x_{n-1}}{a_{ii}^{(i)}}$
- gdzie $i = n-1, n-2, \dots, 1$

Metoda Eliminacji Gaussa (5)

Dodatkowe uwagi:

- Co zrobić, jeśli na diagonali występują elementy równe 0?
- **Liczba operacji** w celu wyznaczenia rozwiązania:
 - $M = 1/3n^3 + n^2 - 1/3n$ mnożeń i dzielen,
 - $D = 1/3n^3 + 1/2n^2 - 5/6n$ dodawań.
 - w sumie złożoność obliczeniowa: $\mathcal{O}(n^3)$
- Jeżeli chcemy uzyskać rozwiązanie dla tej samej macierzy **A** ale **wielu** (m) prawych stron ($\mathbf{Ax}_i = \mathbf{b}_i$), złożoność obliczeniowa wynosi: $\mathcal{O}(n^3) + \mathcal{O}(mn^2)$
- \Rightarrow potrzebna metoda, która *modyfikuje* tylko lewą stronę (macierz **A**) i pozwala stosować **dowolną** prawą stronę (wektor **b**) \Rightarrow **faktoryzacja LU**.

Dygresja (6)

#FinTech?

Dygresja (7)

TANG *et al.*: EXTREME LEARNING MACHINE FOR MULTILAYER PERCEPTRON

811

and for radial basis function (RBF) nodes with activation function g , G_i is defined as

$$G_i(\mathbf{x}, \mathbf{a}_i, b_i) = g(b_i ||\mathbf{x} - \mathbf{a}_i||). \quad (3)$$

Huang *et al.* [5] have proved that the SLFNs are able to approximate any continuous target functions over any compact subset $X \in R^d$ with above random initialized adaptive or RBF nodes. Let $L^2(X)$ be a space of functions f on a compact subset X in the d -dimensional Euclidean space R^d such that $|f|^2$ is integrable, that is, $\int_X |f(\mathbf{x})|^2 d\mathbf{x} < \infty$. For $u, v \in L^2(X)$, the inner product $\langle u, v \rangle$ is defined by

$$\langle u, v \rangle = \int_X u(\mathbf{x})v(\mathbf{x}) d\mathbf{x}. \quad (4)$$

The norm in $L^2(X)$ space is denoted as $||\cdot||$, and the closeness between network function f_n and the target function f is measured by the $L^2(X)$ distance

$$||f_L - f|| = \left(\int_X |f_L(\mathbf{x}) - f(\mathbf{x})|^2 d\mathbf{x} \right)^{1/2}. \quad (5)$$

Theorem 2.1. Given any bounded nonconstant piecewise continuous function $g: R \rightarrow R$, if span $\{G(\mathbf{a}, b, \mathbf{x}) : (\mathbf{a}, b) \in R^d \times R\}$ is dense in L^2 , for any target function f and any function sequence $g_i(\mathbf{x}) = G(\mathbf{a}_i, b_i, \mathbf{x})$ randomly generated based on any continuous sampling distribution, $\lim_{n \rightarrow \infty} ||f - f_n|| = 0$ holds with probability one if the output weights β_i are determined by ordinary least square to minimize $||f(\mathbf{x}) - \sum_{i=1}^L \beta_i g_i(\mathbf{x})||$.

The theorem above [5], [16], [17] shows that randomly generated networks with the outputs being solved by least mean square are able to maintain the universal approximation capability, if and only if the activation function g is nonconstant piecewise and span $\{G(\mathbf{a}, b, \mathbf{x}) : (\mathbf{a}, b) \in R^d \times R\}$ is dense in L^2 . Based on this theorem, ELM can be established for fast learning, which will be described in detail in the next section.

B. ELM Learning Algorithm

According to the Theorem 2.1, the ELM can be built with randomly initialized hidden nodes. Given a training set $\{(\mathbf{x}_i, t_i) | \mathbf{x}_i \in R^d, t_i \in R^m, i = 1, \dots, N\}$, where \mathbf{x}_i is the training data vector, t_i represents the target of each sample, and L denotes the number of hidden nodes.

From the learning point of view, unlike traditional learning algorithms (see the related works referred to in [7]), ELM does not aim to search the smallest training error but also

and \mathbf{T} is the training data target matrix

$$\mathbf{T} = \begin{bmatrix} \mathbf{t}_1^T \\ \vdots \\ \mathbf{t}_N^T \end{bmatrix} = \begin{bmatrix} t_{11} & \cdots & t_{1m} \\ \vdots & \ddots & \vdots \\ t_{N1} & \cdots & t_{Nm} \end{bmatrix}. \quad (8)$$

The ELM training algorithm can be summarized as follows [1].

- 1) Randomly assign the hidden node parameters, e.g., the input weights \mathbf{a}_i and biases b_i for additive hidden nodes, $i = 1, \dots, L$.
- 2) Calculate the hidden layer output matrix \mathbf{H} .
- 3) Obtain the output weight vector

$$\boldsymbol{\beta} = \mathbf{H}^\dagger \mathbf{T} \quad (9)$$

where $\mathbf{T} = [\mathbf{t}_1, \dots, \mathbf{t}_N]^T$, \mathbf{H}^\dagger is the Moore-Penrose generalized inverse of matrix \mathbf{H} .

The orthogonal projection method can be efficiently used for the calculation of MP inverse: $\mathbf{H}^\dagger = (\mathbf{H}^T \mathbf{H})^{-1} \mathbf{H}^T$, if $\mathbf{H}^T \mathbf{H}$ is nonsingular; or $\mathbf{H}^\dagger = \mathbf{H}^T (\mathbf{H}^T \mathbf{H})^{-1}$, if $\mathbf{H} \mathbf{H}^T$ is nonsingular. According to the ridge regression theory, it was suggested that a positive value $(1/\lambda)$ is added to the diagonal of $\mathbf{H}^T \mathbf{H}$ or $\mathbf{H} \mathbf{H}^T$ in the calculation of the output weights $\boldsymbol{\beta}$. By doing so, according to [1] and [7], the resultant solution is equivalent to the ELM optimization solution with $\sigma_1 = \sigma_2 = u = v = 2$, which is more stable and has better generalization performance. That is, in order to improve the stability of ELM, we can have

$$\boldsymbol{\beta} = \mathbf{H}^T \left(\frac{1}{\lambda} + \mathbf{H} \mathbf{H}^T \right)^{-1} \mathbf{T} \quad (10)$$

and the corresponding output function of ELM is

$$f(\mathbf{x}) = h(\mathbf{x})\boldsymbol{\beta} = h(\mathbf{x})\mathbf{H}^T \left(\frac{1}{\lambda} + \mathbf{H} \mathbf{H}^T \right)^{-1} \mathbf{T} \quad (11)$$

or we can have

$$\boldsymbol{\beta} = \left(\frac{1}{\lambda} + \mathbf{H} \mathbf{H}^T \right)^{-1} \mathbf{H}^T \mathbf{T} \quad (12)$$

and the corresponding output function of ELM is

$$f(\mathbf{x}) = h(\mathbf{x})\boldsymbol{\beta} = h(\mathbf{x}) \left(\frac{1}{\lambda} + \mathbf{H} \mathbf{H}^T \right)^{-1} \mathbf{H}^T \mathbf{T}. \quad (13)$$

C. ELM-Based Autoencoder

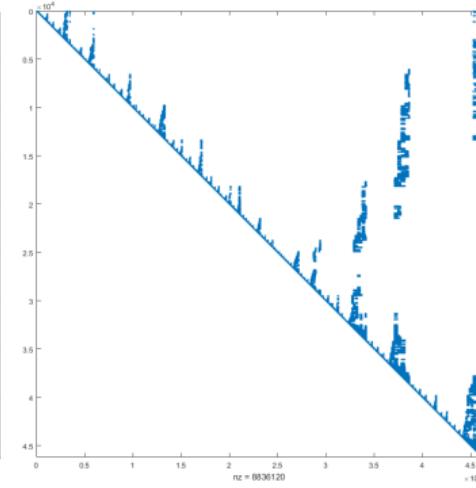
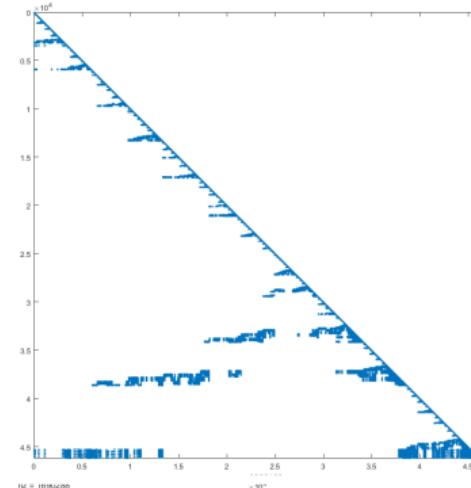
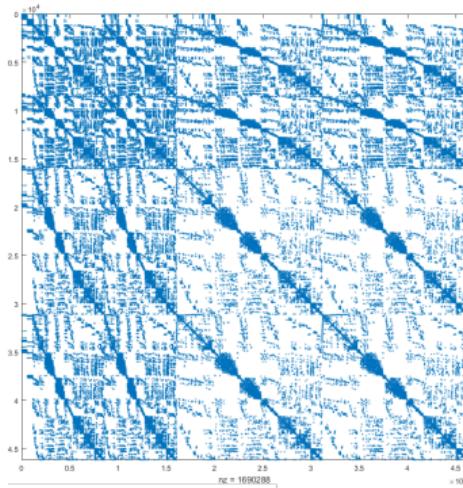
Apart from the ELM-based SLFNs, the ELM theory has also been applied to build autoencoders for MLPs.

Faktoryzacja LU (1)

Cechy faktoryzacji LU:

- ① **L** - lower (macierz trójkątna dolna), **U** - upper (macierz trójkątna górska),
- ② Jeden z **wariantów** metody eliminacji Gaussa,
- ③ **Powszechnie** stosowany w praktycznych aplikacjach (MKL Pardiso!)
- ④ Szczególnie, kiedy układ równań musi być rozwiązywany dla **wielu** prawych stron (wektor **b**), natomiast lewa strona równania (macierz **A**) **nie zmienia się**
- ⑤ **Dygresja** - faktoryzacja symboliczna, numeryczna, rozwiązywanie.

Faktoryzacja LU (1)



Faktoryzacja LU (2) - przykład

Dana jest macierz $\mathbf{A} \in \mathbb{R}^{4 \times 4}$

$$\begin{bmatrix} 2 & 1 & 1 & 0 \\ 4 & 3 & 3 & 1 \\ 8 & 7 & 9 & 5 \\ 6 & 7 & 9 & 8 \end{bmatrix}$$

- Operacja $\mathbf{L}_1\mathbf{A}$:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ -2 & 1 & 0 & 0 \\ -4 & 0 & 1 & 0 \\ -3 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 2 & 1 & 1 & 0 \\ 4 & 3 & 3 & 1 \\ 8 & 7 & 9 & 5 \\ 6 & 7 & 9 & 8 \end{bmatrix} = \begin{bmatrix} 2 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 3 & 5 & 5 \\ 0 & 4 & 6 & 8 \end{bmatrix}$$

- Operacja $\mathbf{L}_2\mathbf{L}_1\mathbf{A}$:

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

Faktoryzacja LU (3) - przykład

- Operacja $\mathbf{L}_3\mathbf{L}_2\mathbf{L}_1\mathbf{A} = \mathbf{U}$ - otrzymaliśmy macierz trójkątną górną:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -1 & 1 \end{bmatrix} \begin{bmatrix} 2 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 2 & 2 \\ 0 & 0 & 2 & 4 \end{bmatrix} = \begin{bmatrix} 2 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 2 & 2 \\ 0 & 0 & 0 & 2 \end{bmatrix}$$

- W celu przekształcenia równania do postaci: $\mathbf{A} = \mathbf{LU}$ wykonujemy:
 - $\mathbf{L}_3^{-1}\mathbf{L}_3\mathbf{L}_2\mathbf{L}_1\mathbf{A} = \mathbf{L}_3^{-1}\mathbf{U} \implies \mathbf{IL}_2\mathbf{L}_1\mathbf{A} = \mathbf{L}_3^{-1}\mathbf{U}$
 - $\mathbf{L}_2^{-1}\mathbf{L}_2\mathbf{L}_1\mathbf{A} = \mathbf{L}_2^{-1}\mathbf{L}_3^{-1}\mathbf{U} \implies \mathbf{IL}_1\mathbf{A} = \mathbf{L}_2^{-1}\mathbf{L}_3^{-1}\mathbf{U}$
 - $\mathbf{L}_1^{-1}\mathbf{L}_1\mathbf{A} = \mathbf{L}_1^{-1}\mathbf{L}_2^{-1}\mathbf{L}_3^{-1}\mathbf{U} \implies \mathbf{A} = \mathbf{L}_1^{-1}\mathbf{L}_2^{-1}\mathbf{L}_3^{-1}\mathbf{U}$
- Macierz odwrotna do \mathbf{L}_1 (\mathbf{L}_i) - **odwracamy** znak elementów poza diagonalą:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ -2 & 1 & 0 & 0 \\ -4 & 0 & 1 & 0 \\ -3 & 0 & 0 & 1 \end{bmatrix}^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 2 & 1 & 0 & 0 \\ 4 & 0 & 1 & 0 \\ 3 & 0 & 0 & 1 \end{bmatrix}$$

Faktoryzacja LU (4) - przykład

- Ostatecznie, $\mathbf{L}_1^{-1} \mathbf{L}_2^{-1} \mathbf{L}_3^{-1}$ jest macierzą trójkątną dolną, z elementami macierzy \mathbf{L}_1^{-1} , \mathbf{L}_2^{-1} , \mathbf{L}_3^{-1} wstawionymi w odpowiednie miejsca:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 2 & 1 & 0 & 0 \\ 4 & 3 & 1 & 0 \\ 3 & 4 & 1 & 1 \end{bmatrix}$$

- Macierz \mathbf{A} możemy więc przedstawić jako iloczyn dwóch macierzy trójkątnych: $\mathbf{A} = \mathbf{LU}$. Jest to tzw. **faktoryzacja (dekompozycja) LU**:

$$\begin{bmatrix} 2 & 1 & 1 & 0 \\ 4 & 3 & 3 & 1 \\ 8 & 7 & 9 & 5 \\ 6 & 7 & 9 & 8 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 2 & 1 & 0 & 0 \\ 4 & 3 & 1 & 0 \\ 3 & 4 & 1 & 1 \end{bmatrix} \begin{bmatrix} 2 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 2 & 2 \\ 0 & 0 & 0 & 2 \end{bmatrix}$$

- Macierz \mathbf{L} ma na diagonali same jedynki!

Faktoryzacja LU (5)

Rozwiązywanie liniowego układu równań $\mathbf{Ax} = \mathbf{b}$ za pomocą faktoryzacji LU:

- Tworzymy macierze \mathbf{L} i \mathbf{U} , takie, że: $\mathbf{LUx} = \mathbf{b}$
- Tworzymy wektor pomocniczy: $\mathbf{y} = \mathbf{Ux}$
- Rozwiążujemy układ równań: $\mathbf{Ly} = \mathbf{b}$ za pomocą podstawiania wprzód ($\mathcal{O}(n^2)$)
- Rozwiążujemy układ równań: $\mathbf{Ux} = \mathbf{y}$ za pomocą podstawiania wstecz ($\mathcal{O}(n^2)$)

Ogólna formuła na tworzenie macierzy \mathbf{L} i \mathbf{U} :

Algorithm 1: faktoryzacja LU macierzy \mathbf{A} o rozmiarze $m \times m$

```

 $\mathbf{U} = \mathbf{A}, \mathbf{L} = \mathbf{I}$ 
for  $k = 1$  to  $m-1$  do
    for  $j = k+1$  to  $m$  do
         $l_{jk} = u_{jk}/u_{kk}$ 
         $u_{j,k:m} = u_{j,k:m} - l_{jk}u_{k,k:m}$ 
    end
end

```

Faktoryzacja LU (6)

Dodatkowe uwagi na temat faktoryzacji LU:

- LU w Matlabie: $[L \ U] = lu(A)$;
- Co zrobić, kiedy na diagonali \mathbf{A} znajdują się elementy o wartości 0?
Odp. *Pivoting strategy*
- Złożoność obliczeniowa, podobnie jak w metodzie Gaussa: $\mathcal{O}(n^3)$
- Faktoryzacja LU może być wykorzystana do uzyskania macierzy odwrotnej:

$$\mathbf{A}^{-1}\mathbf{A} = \mathbf{I}$$

jednak operacja ta jest bardzo rzadko wykorzystywana w praktyce, ze względu na dużą niestabilność numeryczną:

$$\mathbf{A} = \mathbf{LU} \implies \mathbf{A}^{-1} = (\mathbf{LU})^{-1} \implies \mathbf{A}^{-1} = \mathbf{U}^{-1}\mathbf{L}^{-1}$$

- Przewaga rozkładu LU nad eliminacją Gaussa: faktoryzacja LU pozwala na rozwiązanie układu równań $\mathbf{Ax} = \mathbf{b}$ dla dowolnej prawej strony, a kosztowną numerycznie część obliczeń (faktoryzację) wykonuje się tylko raz.

Faktoryzacja Choleskiego (1)



- Jeżeli macierz $\mathbf{A} \in \mathbb{R}^{n \times n}$ jest symetryczna i dodatnio określona ($\forall \mathbf{x} \in \mathbb{R}^n, \mathbf{x} \neq 0: \mathbf{x}^T \mathbf{A} \mathbf{x} > 0$), można zastosować Faktoryzację **Choleskiego** (Banachowicza)
- $\mathbf{A} = \mathbf{L} \mathbf{L}^T$
- Koszt numeryczny ok. połowę mniejszy (w porównaniu do LU)
- Inne rodzaje faktoryzacji: QR, LDLT, SVD itp.

[https://www.polytechnique.edu/
bibliotheque/sites/all/bibliotheque/
bloch_eleve_coll_archives_X.jpg](https://www.polytechnique.edu/bibliotheque/sites/all/bibliotheque/bloch_eleve_coll_archives_X.jpg)

Faktoryzacja Choleskiego (2)

Kolejne kroki faktoryzacji Choleskiego: $\mathbf{LL}^T = \mathbf{A}$

$$\begin{bmatrix} l_{11} & 0 & 0 & 0 \\ l_{21} & l_{22} & 0 & 0 \\ l_{31} & l_{32} & l_{33} & 0 \\ l_{41} & l_{42} & l_{43} & l_{44} \end{bmatrix} \begin{bmatrix} l_{11} & l_{12} & l_{13} & l_{14} \\ 0 & l_{22} & l_{23} & l_{24} \\ 0 & 0 & l_{33} & l_{34} \\ 0 & 0 & 0 & l_{44} \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix}$$

Rozwiążujemy kolejno następujące równania (dla pierwszej kolumny \mathbf{A}):

$$\begin{cases} l_{11}^2 = a_{11} \\ l_{21}l_{11} = a_{21} \\ l_{31}l_{11} = a_{31} \\ l_{41}l_{11} = a_{41} \end{cases}$$

Faktoryzacja Choleskiego (3)

Zadanie domowe: wykonać faktoryzację Choleskiego poniższej macierzy:

$$\begin{bmatrix} 4 & -2 & 1 & 0 \\ -2 & 6 & -2 & -2 \\ 1 & -2 & 8 & -2 \\ 0 & -2 & -2 & 10 \end{bmatrix}$$

Octave GNU

Uwagi końcowe

- ➊ Rozwiązywanie układów równań liniowych jest sercem wielu innych algorytmów numerycznych — podobno szacuje się, że **około 75%** czasu obliczeniowego **superkomputerów** jest wykorzystywanych właśnie na rozwiązywanie takich zadań. - <http://wazniak.mimuw.edu.pl/index.php?title=MN05>
- ➋ Metoda Cramera (wyznacznikowa) oraz jawne odwracanie macierzy **nie jest stosowane w praktyce** do rozwiązywania układów równań liniowych.
- ➌ W praktyce, żeby zminimalizować błąd numeryczny, wykorzystujemy tzw. **strategię wyboru elementu głównego w kolumnie**. Polega to na tym, że zanim wykonamy k-ty krok algorytmu rozkładu LU:
 - w pierwszej kolumnie podmacierzy $\mathbf{A}(k : N, k : N)$ szukamy elementu o największym module,
 - zamieniamy ze sobą wiersz $\mathbf{A}(k, 1 : N)$ z wierszem, w którym znajduje się element główny,
 - zapamiętujemy dokonaną permutację, bo potem będziemy musieli dokonać analogicznej permutacji wektora prawej strony.