1 Rozdział

Strona: LeIA
Kurs: Metody numeryczne (2025Z)
Książka: 1 Rozdział
Wydrukowane przez użytkownika: Kinga Kondraciuk
Data: niedziela, 30 listopada 2025, 13:59

Spis treści

1. Elementy teorii błędów

Metody numeryczne dają nam możliwość rozwiązywania pewnych zagadnień w sposób przybliżony wtedy, gdy dokładne metody nie mogą być stosowane lub nie są znane.

Każde rozwiązywanie numeryczne wiąże się z popełnianiem błędów obliczeń. Błędy te mogą wynikać z różnych przyczyn.

Jedną z nich jest podawanie danych w sposób przybliżony - podczas pomiarów lub doświadczeń popełniamy nieścisłości związane np.: z dokładnością przyrządu pomiarowego.

Błędy danych mogą silnie wpływać na wyniki obliczeń, ale nie zawsze. Czasami można podać tzw.: wskaźniki uwarunkowania, które "przenoszą" błędy danych na błędy obliczeń końcowych. Podamy takie wskaźniki dla błędów funkcji jednej i wielu zmiennych.

Drugą przyczyną jest dokładność samego algorytmu stosowanego do obliczeń. Może się zdarzyć, że dokładny wzór np.: rekurencyjny nie nadaje się do obliczeń numerycznych, będziemy o nim mówić, że nie jest stabilny - podamy przykład takiego algorytmu. Nawet kolejność działań na liczbach przybliżonych może mieć znaczenie i wpływać na wynik, a niektóre działania np.: odejmowanie liczb przybliżonych bliskich daje czasami zaskakująco duże błędy.

Trzeba również pamiętać o błędach maszynowych, wynikające z reprezentacji liczb w komputerze. Wszystkie liczby w komputerze są zaokrąglane do wartości, która jest.

Wprawdzie dzisiejsze komputery są bardzo dokładne, to jednak nakładanie się tych wszystkich błędów może dawać absurdalne wyniki. Będziemy ilustrować takie sytuacje.

2. Błędy bezwzględne i względne

Dużymi literami A, B, C,... będziemy oznaczać liczby dokładne, a małymi a, b, c ... przybliżone wartości tych liczb.

Błędem bezwzględnym liczby przybliżonej a nazywamy wartość bezwzględną różnicy między liczbą dokładną, a jej przybliżeniem, co zostało przedstawione we wzorze **1.2.1** .

$$\Delta a = |A - a| \tag{1.2.1}$$

Na ogół nie jest znana wartość liczby A, wtedy a możemy tylko oszacować z góry. W praktyce błędem bezwzględnym nazywamy możliwie najmniejsze oszacowanie takiej różnicy. Na ogół błąd bezwzględny możemy przyjmować jako dokładność przyrządu pomiarowego.

Błędem względnym nazywamy stosunek błędu bezwzględnego do wartości bezwzględnej liczby przybliżonej (patrz wzór **1.2.2**):

$$\delta a = \frac{\Delta a}{|a|} \tag{1.2.2}$$

dla a różnego od zera.

Przykład 1.2.1

Przykład

Jeśli liczba dokładna $A = 1.88$, a chcemy podać jej przybliżenie z dokładnością do jednego miejsca po przecinku to $a = 1.9$ i błąd $\Delta a = |A - a| = 0.02$. Oczywiście nie "obcinamy" liczby dokładnej do jednego miejsca po przecinku (ang. truncate), tylko zaokrąglamy cyfrę 8 do cyfry 9. Gdybyśmy "obcięli" i za liczbę przybliżoną przyjęli $a^* = 1.8$ błąd bezwzględny byłby o wiele większy: $\Delta a^* = |A - a| = 0.08$.

Obliczymy jeszcze błędy względne obu przybliżeń. Błąd względny dla wartości zaokrąglonej wynosi:

$$\delta a = \frac{\Delta a}{|a|} = \frac{0.02}{1.9} = 0.0105, \tag{1.2.1.1}$$

co stanowi 1.05 liczby przybliżonej, natomiast błąd względny dla wartości obciętej wynosi (patrz **1.2.1.2**):

$$\delta a^* = \frac{\Delta a^*}{|a^*|} = \frac{0.08}{1.8} = 0.0444 \tag{1.2.1.2}$$

co stanowi 4.44 liczby przybliżonej.

Będziemy się posługiwać pojęciem cyfr dokładnych liczby przybliżonej. W podanym przykładzie w liczbie $a = 1.9$ wszystkie cyfry będą dokładne mimo, że w liczbie dokładnej nie występuje cyfra 9, natomiast w przybliżeniu $a^* = 1.8$ cyfra 8 nie jest dokładna mimo, że taka sama cyfra i na takim samym miejscu występuje w A. Liczba przybliżona będzie mieć wszystkie cyfry dokładne, jeśli jej błąd bezwzględny nie będzie przekraczać połowy ostatniego uwzględnionego miejsca dziesiętnego.

Przykład 1.2.2

Przykład

Dany jest szereg: $\sum_{n=1}^{\infty}(-1)^n \frac{2n}{(n+2)!}$, obliczymy jego przybliżoną sumę, przyjmując dokładność $\varepsilon = 10^{-10}$. Oczekiwany wynik przybliżony powinien być obarczony błędem bezwzględnym mniejszym od wartości ε.
Sumujemy szereg naprzemienny zbieżny do zera. Jeśli za przybliżoną sumę szeregu będziemy brać n-tą sumę częściową (n pierwszych wyrazów), to błąd bezwzględny między dokładną sumą a jej przybliżeniem nie będzie przekraczał wartości bezwzględnej pierwszego odrzuconego wyrazu czyli $|a_n + 1|$. Zatem będziemy brać tyle wyrazów, aż sąsiednie sumy będą się różnić o mniej niż podana dokładność $10^{-10} = 0,0000000001$. Zatem kolejne wyrazy możemy obliczyć za pomocą ciągu:
$s_1 = \frac{-2}{3!}, s_2 = \frac{-2}{3!} + \frac{4}{4!}, s_n = s_{n-1} + (-1)^n \frac{2n}{(n+2)!}$

> i sumujemy tak długo, aż $s_n - s_{n-1} < \varepsilon = 10^{-10}$. Okazuje się, że wystarczy zsumować $n = 13$ wyrazów i wtedy przybliżona suma będzie wynosić $s_n = -0.2072766470$.

Przyjrzyjmy się teraz krótkiemu programowi w MATLABie ilustrującemu powyższe obliczenia.

```
function p1_2_2
    format long
    % obliczamy sumę najpierw 13, a potem 14 pierwszych wyrazów
    c_13=ciag(13)
    c_14=ciag(14)
    c_13-c_14

    % obliczamy sume cześciową z założoną dokładnością
    % c_13 zawiera wartość, n_13 zawiera liczbę zsumowanych składników
    [c_13, n_13] = ciag_dokladnosc(1e-10)
end

% wariant funkcji z określoną maksymalną liczba składników
function s = ciag(max_n)
    s=0;
    for n=1:max_n
        s = s + (-1)^n * (2*n)/factorial(n+2);
    end
end

% wariant funkcji z określoną dokładnością
function [s,n] = ciag_dokladnosc(max_eps)
    s=0;
    for n=1:50
        sn = (-1)^n * (2*n)/factorial(n+2);
        s = s + sn;
        if abs(sn) < max_eps
            break;
        end
    end
 end
```

Powyższy listing powinien zwrócić wyniki zaprezentowane poniżej. Możemy zaobserwować, że obie wartości c_{13} są zgodne z podanymi w przykładzie. Wartość n_{13} zawiera rzeczywistą liczbę składników uwzględnionych w sumie. Jednocześnie wartość ans, która reprezentuje różnicę między dwoma kolejnymi przybliżeniami obliczonymi dla $n = 13$ oraz $n = 14$ jest mniejsza od założonej dokładności $\varepsilon = 10^{-10}$.

```
>> p1_2_2
c_13 =
  -0.207276647029913
c_14 =
  -0.207276647028574
ans =
    -1.338262833883164e-12
c_13 =
  -0.207276647029913
n_13 =
     13
```

3. Błędy funkcji jednej zmiennej

Błędy funkcji jednej zmiennej

Błędem funkcji jednej zmiennej należy interpretować jako jej właściwość związaną z wrażliwością na dokładność zadawanych jej danym wejściowym. Niektóre funkcje *przenoszą* błędy danych wejściowych zwiększając inne funkcje tłumią błędy danych wejściowych. Wrażliwość funkcji określa się wskaźnikiem uwarunkowania, którego definicję wyprowadzimy w tym rozdziale.

Przykład 1.3.1

Przykład

Zmierzyliśmy długość boku sześcianu i otrzymaliśmy wynik $x = 2.3$ cm, ale naszą miarką możemy zmierzyć z dokładnością do 0.03 cm.
Jak błąd długości boku wpłynie na błąd objętości tego sześcianu?
Mamy następujące dane: bok $x = 2.3$ cm , błąd $\Delta x = 0.03$ cm, objętość sześcianu jest funkcją boku i wynosi $v(x) = x^3$.
Szukamy Δv i δv, czyli błędu bezwzględnego i względnego objętości.
Aby wykonać obliczenia podamy ogólne wzory na te błędy.

Rozpatrujemy funkcję jednej zmiennej $f(x)$ i argument x jest obarczony błędem bezwzględnym Δx. Wtedy błąd bezwzględny funkcji, oznaczany przez Δf, równa się:

$$\Delta y = \Delta f = |f'(x)|\Delta x \tag{1.3.1}$$

gdzie pochodną we wzorze obliczamy dla wartości podanego argumentu . Wzór ten wynika ze wzoru Taylora funkcji jednej zmiennej:

$$f(x + \Delta x) = f(x) + \Delta x f'(x) + \frac{\Delta x^2}{2!} f''(x) + \frac{\Delta x^3}{3!} f'''(x) + \dots$$

Przenieśmy składnik $f(x)$ na lewą równania i "obetnijmy" szereg po składniku z pierwszą pochodną. Wówczas możemy zapisać wyrażenie na błąd przybliżony:

$$f(x + \Delta x) - f(x) = \Delta x f'(x)$$

ponieważ $\Delta f = f(x + \Delta x) - f(x)$. Zauważmy dodaną wartość bezwzględną w wyrażeniu **1.3.1** , która gwarantuje, że błąd jest wyrażony zawsze jako wartość dodatnia.
Z ogólnego wzoru na błąd względny możemy zapisać:

$$\delta y = \delta f = \frac{\Delta f}{|f(x)|} \tag{1.3.2}$$

Wzór ten można przekształcić, wstawiając do niego wzór na błąd bezwzględny i otrzymamy:

$$\delta y = \delta f = \frac{\Delta f}{|f(x)|} = \frac{|f'(x)\Delta x|}{|f(x)|} = \frac{|f'(x)\cdot x|}{|f(x)|}\frac{\Delta x}{x} = \omega \cdot \delta x$$

gdzie wielkość $\omega = \frac{|f'(x)\cdot x|}{|f(x)|}$ nazywamy **wskaźnikiem uwarunkowania** i za jego pomocą możemy zapisać wzór na błąd względny funkcji:

$$\delta f = \omega \cdot \delta x \tag{1.3.3}$$

Z tego wzoru widać, że wskaźnik ten "przenosi" błąd względny z argumentu na funkcję.

Wróćmy do przykładu 1.3.1. Korzystając z powyższych wzorów mamy:

$$v(x) = x^3, v'(x) = 3x^2, x = 2.3, \Delta x = 0.03$$

$$v(x) = 12.2, \Delta v = |3 \cdot (2.3)^2| \cdot 0.03 = 0.476$$

$$\delta v = \frac{0.476}{12.2} = 0.039, \delta x = \frac{0.03}{2.4} = 0.013, \omega = \frac{\delta v}{\delta x} = 3$$

Błąd względny funkcji powiększył się 3 razy w stosunku do błędu względnego argumentu.
Oczywiście wyniki są zaokrąglone. Ponieważ błąd bezwzględny objętości wynosi 0.5 nie ma sensu podawać w wyniku więcej cyfr po przecinku, nawet cyfra 2 po przecinku nie jest cyfrą dokładną.
Wyniki na błędy względne podane są z trzema cyframi po przecinku, żeby wyraźnie było widać, że błąd względny wzrósł 3 razy.

W następnym przykładzie błąd względny funkcji dla wartości $x = 1$ i $x = -1$ rośnie do nieskończoności, a im bliższe są argumenty tych wartości tym błąd jest większy. Wiąże się to z odejmowaniem liczb przybliżonych bliskich. Proszę porównać przykład z tematu: Błędy działań arytmetycznych.

Przykład 1.3.2

Przykład

Dana jest funkcja $f(x) = x^2 - 1$. Napisać wzór na wskaźnik uwarunkowania. Obliczyć go dla różnych wartości argumentu x. Obliczyć błędy względne funkcji dla różnych argumentów, biorąc za błąd względny argumentu 5% jego wartości (bezwzględnej).
Obliczymy pochodną funkcji i podamy wzory na błędy:

$$f'(x) = 2x, \quad \Delta f = |2x| \cdot \Delta x, \quad \delta x = 0.05, \ \Delta x|x|,$$

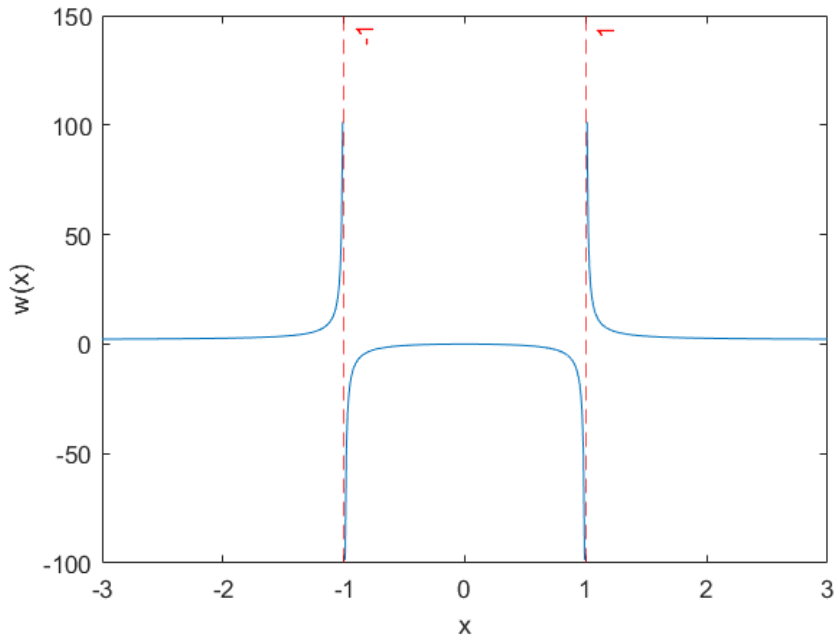
$$\omega(x) = |\frac{f'(x)\cdot x}{f(x)}| = |\frac{2x\cdot x}{x^2-1}| = |\frac{2x^2}{x^2-1}|$$

$$\delta f(x) = \omega(x) \cdot \delta x$$

Dla $x = 1$ oraz $x = -1$ nie można obliczyć błędu względnego funkcji. Jeśli x będzie bliski jedności, wskaźnik uwarunkowania będzie duży i błąd względny funkcji też będzie duży. Badając funkcję proszę wstawiać argumenty dalekie od 1 i bliskie np.: 1,03.
Przeanalizujmy krótki program w MATLABie, który pozwoli nam zbadać właściwości tej funkcji.

```
function p1_3_2
 format short
 % definiujemy wyrażenia (funkcje) dla przykładu
 f = @(x) (x*x-1);
 df = @(x) (2*x);
 w = @(x) (df(x)*x/f(x));

 % Obliczamy kilka wyników
 fprintf("Wartość w(1.01)\t= %f\n", w(1.01));
 fprintf("Wartość w(1.03)\t= %f\n", w(1.03));
 fprintf("Wartość w(-2)\t= %f\n", w(-2));
 fprintf("Wartość w(10.5)\t= %f\n", w(10.5));

 % aby przeanalizować zmienność funkcji narysujemy zależność
 % współczynnika uwarunkowania od x
 W = [];
 X = -3:0.01:3;
 for x=X
     % doklejamy do wektora wyników kolejną wartość
     W = [W w(x)];
 end

 % rysujemy przebieg zależności w(x)
 plot(X,W)
 xlabel('x')
 ylabel('w(x)')

 % dorysowujemy ważne linie wskazujące wartości osobliwe gdy wskaznik uwarunkowania dąży do
 % nieskończoności
 xline(-1, '--r', {'-1'})
 xline(1, '--r', {'1'})
end
```

Powyższy program wygeneruje rysunek reprezentujący zależność wskaźnika uwarunkowania $\omega(x)$ od x dla $x\epsilon$ < -3, 3 >.



Rys. 1.3.1. Przebieg zależności wskaźnika uwarunkowania dla funkcji z przykładu 1.3.2

Po uruchomieniu program powinien wyświetlić cztery wartości współczynnika uwarunkowania dla przykładowych wartości x. Uruchomienie programu oraz obserwację wyników pozostawiamy czytelnikowi.

4. Błędy funkcji wielu zmiennych

Przykład 1.4.1

Przykład

Zmierzyliśmy boki prostopadłościanu i otrzymaliśmy x=1.2cm, y=1.8cm oraz z=2.1cm. Nasz przyrząd pomiarowy ma dokładność 0,01cm. Jaki popełnimy błąd bezwzględny i względny licząc pole powierzchni całkowitej tego prostopadłościanu?

Mamy dane: x=1.2, y=1.8, z=2.1, błędy bezwzględne przyjmujemy dla wszystkich boków
$\Delta x = \Delta y = \Delta z = \Delta = 0.01$ wzór na pole $p(x, y, z) = 2xy + 2xz + 2yz$.

W obliczeniach skorzystamy z następujących wzorów definiujących błędy dla funkcji wielu zmiennych.

Rozważania przeprowadzimy dla funkcji 2 zmiennych, ale wszystkie wzory można uogólnić na więcej zmiennych. Dana jest funkcja $f(x, y)$ i argumenty są obarczone błędami $\Delta x, \Delta y$. Wzór na błąd bezwzględny funkcji wynika, tak jak dla funkcji jednej zmiennej, ze wzoru Taylora (nie będziemy go wyprowadzać) i jest następujący:

$$\Delta f(x, y) = |\frac{\partial f(x,y)}{\partial x}|\Delta x + |\frac{\partial f(x,y)}{\partial y}|\Delta y \quad \textbf{(1.4.1)}$$

Pochodne cząstkowe są liczone dla tych wartości argumentów, dla których liczymy błąd.

Z ogólnego wzoru na błąd względny otrzymujemy:

$$\delta f = \frac{\Delta f}{|f(x)|} \quad \textbf{(1.4.2)}$$

Przekształcimy ten wzór tak, jak wzór na błąd funkcji jednej zmiennej, aby wprowadzić wskaźniki uwarunkowania.

$$\delta f = \frac{\Delta f}{|f(x)|} = \frac{|f_x(x,y)|\Delta x + |f_y(x,y)|\Delta y}{|f(x,y)|} = |\frac{f_x(x,y)\cdot x}{f(x,y)}|\frac{\Delta x}{|x|} + |\frac{f_y(x,y)\cdot y}{f(x,y)}|\frac{\Delta y}{|y|} = \omega_x \cdot \delta x + \omega_y \cdot \delta y \quad \textbf{(1.4.3)}$$

Wielkości, które wprowadziliśmy:

$$w_x = |\frac{f_x(x,y)\cdot x}{f(x,y)}|, w_y = |\frac{f_y(x,y)\cdot y}{f(x,y)}| \quad \textbf{(1.4.4)}$$

nazywamy wskaźnikami uwarunkowania odpowiednio zmiennej x i y, "przenoszą" one błędy względne argumentów na błąd względny funkcji.

Powrócimy do przykładu 1.4.1, podając jednocześnie wzory dla funkcji 3 zmiennych.

Mamy dane: x=1.2, y=1.8, z=2.1, błędy bezwzględne przyjmujemy dla wszystkich boków $\Delta x = \Delta y = \Delta z = \Delta = 0.01$, wzór na pole $p(x, y, z) = 2xy + 2xz + 2yz$. Pochodne cząstkowe względem poszczególnych argumentów wynoszą:

$$p_x(x, y, z) = 2y + 2z, \quad p_y(x, y, z) = 2x + 2z, \quad p_z(x, y, z) = 2x + 2y$$

Wartość pola $p = 16, 9cm^2$.

Zatem wartość błędu możemy wyznaczyć następująco:

$$\Delta p(x, y, z) = |\frac{\partial p(x,y,z)}{\partial x}\Delta x| + |\frac{\partial p(x,y,z)}{\partial y}\Delta y| + |\frac{\partial p(x,y,z)}{\partial z}\Delta z| = [|2\cdot 1.8 + 2\cdot 2.1|] + [|2\cdot 1.2 + 2\cdot 2.1|] + [|2\cdot 1.2 + 2\cdot 1.8|]\cdot 0.01 = 0.2$$

$$\delta p = \frac{\Delta p}{p(x,y,z)} = 0.012, \quad w_x = 0.553, \quad w_y = 0.702, \quad w_z = 0.745$$

Program przykładowy ilustrujący rozwiązanie w MATLABie:

```
function p1_4_1
    x = 1.2;
    y = 1.8;
    z = 2.1;
    Dx = 0.01; Dy = 0.01; Dz = 0.01;
    dx = Dx/x; dy = Dy/y; dz = Dz/z;
    p = @(x,y,z) (2*x*y+2*x*z+2*y*z);
    dpx = @(x,y,z) (2*y+2*z);
    dpy = @(x,y,z) (2*x+2*z);
    dpz = @(x,y,z) (2*x+2*y);

    % pierwszy sposób z definicji obliczenia
    Dp = @(x,y,z) (dpx(x,y,z)*Dx + dpy(x,y,z)*Dy + dpz(x,y,z)*Dz);
    dp = Dp(x,y,z) / p(x,y,z)


    % drugi sposób wykorzystujący wskaźniki uwarunkowania
    wx = dpx(x,y,z)*x / p(x,y,z)
    wy = dpy(x,y,z)*y / p(x,y,z)
    wz = dpz(x,y,z)*z / p(x,y,z)
    dp = wx*dx + wy*dy + wz*dz
end
```

Powyższy program po uruchomieniu powinien wyświetlić wyniki zgodne z obliczeniami:

```
>> p1_4_1
dp =
    0.0121
wx =
    0.5532
wy =
    0.7021
wz =
    0.7447
dp =
    0.0121
```

5. Błędy działań arytmetycznych

Błędy działań arytmetycznych

Skorzystamy ze wzoru (**1.4.1**) na błąd bezwzględny funkcji dwóch zmiennych, aby wyprowadzić wzory na błędy działań arytmetycznych.

Błąd sumy dwóch liczb przybliżonych : Argumenty x i y są obarczone odpowiednio błędami bezwzględnymi $\Delta x, \Delta y$, sumę argumentów zapisujemy jako $s(x,y)=x=y$. Pochodne cząstkowe tej funkcji zarówno po x jak i po y są równe 1, zatem:

$$\Delta s(x,y)=1 \cdot \Delta x + 1 \cdot \Delta y=\Delta x + \Delta y \quad \textbf{(1.5.1)}$$

Zatem błąd bezwzględny sumy równa się sumie błędów bezwzględnych składników. Błąd względny trzeba obliczyć z ogólnego wzoru:

$$\delta s(x,y) = \frac{\Delta s}{|s(x,y)|}= \frac{\Delta x + \Delta y}{|x+y|} \quad \textbf{(1.5.2)}$$

Błąd różnicy dwóch liczb przybliżonych: Argumenty x i y są obarczone odpowiednio błędami bezwzględnymi $\Delta x, \Delta y$, różnicę argumentów zapisujemy jako $r(x,y)=x-y$. Pochodne cząstkowe tej funkcji: po x jest równa 1, po y jest równa -1, zatem

$$\Delta r(x,y)=1 \cdot \Delta x + |-1|\Delta y = \Delta x + \Delta y \quad \textbf{(1.5.3)}$$

Zatem błąd bezwzględny różnicy równa się **sumie** błędów bezwzględnych składników. Błąd względny trzeba obliczyć z ogólnego wzoru:

$$\delta r(x,y)= \frac{\Delta r}{|r(x,y)|} = \frac{\Delta x + \Delta y}{|x-y|} \quad \textbf{(1.5.4)}$$

Wzór na błąd względny ma sens jeśli x jest różne od y.

Przykład 1.5.1

Przykład

Dane są dwie liczby przybliżone $a=0.0035$ i $b=0.0033$. Wszystkie cyfry tych liczb są dokładne, tzn.: błędy bezwzględne tych liczb są równe nie więcej niż 0,00005. Obliczymy błędy względne tych liczb i błąd względny różnicy $a-b$.

$a=0.0035$, $b=0.0033$, $\Delta a=0.00005$, $\Delta b=0.00005$, $\delta a = \frac{\Delta a}{|a|}=0.015$, $\delta b = \frac{\Delta b}{|b|}=0.015$

Błędy względne składników to 1.5% dla a i 1.5% dla b, natomiast błąd względny różnicy jest bardzo duży w porównaniu z błędami składników i wynosi 100%. Ten niekorzystny efekt jest związany z odejmowaniem liczb przybliżonych bliskich. Jeśli jest możliwość zastąpienia różnicy innym działaniem, należy zastosować inny wzór, aby nie odejmować liczb przybliżonych bliskich.

Błąd iloczynu dwóch liczb przybliżonych: Argumenty x i y są obarczone odpowiednio błędami bezwzględnymi $\Delta x, \Delta y$, iloczyn argumentów zapisujemy jako $i(x,y)=x \cdot y$. Pochodna cząstkowa tej funkcji: po x jest równa y, po y jest równa x, zatem błąd bezwzględny iloczynu jest równy:

$$\Delta i(x,y)=|y| \cdot \Delta x + |x| \cdot \Delta y \quad \textbf{(1.5.5)}$$

Błąd względny trzeba obliczyć z ogólnego wzoru:

$$\delta i(x,y) = \frac{\Delta i}{|i(x,y)|} \frac{|y| \cdot \Delta x + |x| \cdot \Delta y}{x \cdot y}= \frac{\Delta x}{|x|}+\frac{\Delta y}{|y|} =\delta x = \delta y \quad \textbf{(1.5.6)}$$

Zatem błąd względny iloczynu równa się **sumie** błędów względnych czynników. Wzór na błąd względny ma sens jeśli x i y są różne od zera.

Błąd ilorazu dwóch liczb przybliżonych: Argumenty x i y są obarczone odpowiednio błędami bezwzględnymi $\Delta x, \Delta y$, iloraz argumentów zapisujemy jako $ir(x,y)= x/y$ dla y różnego od zera. Pochodna cząstkowa tej funkcji: po x jest równa $\frac{1}{y}$, po y jest

równa $\frac{-x}{y^2}$, zatem

$$\Delta ir(x,y)= \left|\frac{1}{y}\right| \cdot \Delta x + \left|\frac{-x}{y^2}\right| \cdot \Delta y \quad \textbf{(1.5.7)}$$

Błąd względny trzeba obliczyć z ogólnego wzoru:

$$\delta ir(x,y)= \frac{\Delta ir}{|ir(x,y)|}= \frac{\left| \frac{1}{y}\right| \cdot \Delta x + \left| \frac{-x}{y^2}\right| \cdot \Delta y}{\left| \frac{x}{y} \right|} = \frac{\Delta x}{|x|}+ \frac{\Delta y}{|y|}=\delta x + \delta y \quad \textbf{(1.5.8)}$$

Zatem błąd względny ilorazu równa się sumie błędów względnych czynników. Wzór na błąd względny ma sens jeśli x i y są różne od zera.
Na koniec tego tematu podamy przykład, który pokazuje, że sumowanie liczb za pomocą pewnych programów może nie być przemienne.

Przykład 1.5.2

Przykład

W programie, w którym różnica rzędu między liczbami nie może przekraczać 10^{15}, obliczymy na dwa sposoby sumę znacznej ilości liczb bardzo małych i jednej dużej. Zmiana kolejności sumowania (najpierw małe potem duża, albo najpierw duża potem małe) będzie miała znaczący wpływ na wynik.

I sposób:

Do największej liczby $c=26$ dodajemy po kolei liczby $d_i=9 \cdot 10^{16}$ gdzie $i=0,1,...n$, a liczba $n=300000$. Korzystamy ze wzoru rekurencyjnego $s_0=c,s_{i+1}=s_i+d_i$ i w wyniku otrzymujemy sumę $s_{n+1}=S=26.0000000000000$ (13 zer po przecinku, $n+1$ dlatego, że suma uwzględnia jeszcze dodatkowo dużą liczbę).

II sposób:

Sumujemy po kolei liczby małe według wzoru rekurencyjnego: $s_0=0,s_i=s_i+d_i$, a potem dodajemy wynik do liczby dużej: $s_{n+1} =s_n+c$ i w wyniku otrzymujemy
$s_{n+1}=S=26.0000000002700$.

Poniżej przedstawiony został program ilustrując przykład w MATLABie.

```
function p1_5_2
    format long
    d=9e-16;
    c = 26;

    % I sposób (najpierw duża, potem małe)
    s = c;
    n = 300000;
    for i=0:n
        s = s + d;
    end
    s

    % II sposób (najpierw małe, potem duża)
    s = 0;
    n = 300000;
    for i=0:n
        s = s + d;
    end
    s = s + c;
    s
end
```

Uruchomienie programu powinno zwrócić wynik:

```
>> p1_5_2
s =
    26
s =
  26.000000000269999
```

5.1. 6. prawdzamy markdown

2.1. Numeryczna algebra liniowa

Algebra liniowa jest często osobnym kursem ujętym w programie typowych studiów inżynierskich. Niemniej istnieją pewne specjalne techniki, nieuwzględnione w standardowym kursie, które są związane ze specyfiką rozwiązywania zagadnień algebry liniowej na komputerach. Niniejszy rozdział przedstawia kilka wybranych takich metod.

Typowym zagadnieniem z zakresu algebry liniowej wykorzystywanym w ramach problemów inżynierskich jest rozwiązanie układu równań liniowych. Mamy tutaj na myśli nie układy z trzema czy czterema niewiadomymi, które standardowo są rozwiązywane analitycznie, lecz układy z setkami czy nawet dziesiątkami tysięcy niewiadomych. Tego typu problemy wymagają specjalnych technik, które nie tylko gwarantują znalezienie rozwiązania, ale również znajdą je w sposób minimalizujący nakłady oraz błędy obliczeniowe. Wśród metod rozwiązywania układów równań liniowych wyróżniamy metody bezpośrednie oraz metody iteracyjne. Do metod bezpośrednich zaliczamy takie jak: eliminacja Gaussa, faktoryzacje LU czy QR, faktoryzacje SVD. Do metod iteracyjnych zaliczamy Jacobiego, SOR, Gradientów sprzężonych, GMRES, i inne. W niniejszym kursie skupimy się tylko na wybranych metodach bezpośrednich.

Drugim klasycznym zagadnieniem algebry liniowej, który ma duże znaczenie z punktu widzenia inżynierskiego jest wyznaczanie wartości własnych macierzy. Należy zaznaczyć, że wyznaczanie wartości własnych macierzy metodami analitycznymi jest niezmiernie czasochłonne i często sprowadza się do rozwiązania bardzo źle uwarunkowanego wielomianowego równania charakterystycznego. Wśród metod numerycznych pozwalających wyznaczyć wartości i wektory własne są metody pozwalając wyznaczyć wartości własne rzeczywiste i urojone, maksymalną wartość własną, minimalną wartość własną lub wszystkie wartości. W niniejszym podręczniku przedstawimy jedynie podstawowe metody wyznaczania wartości własnych minimalnej i maksymalnej. Pozostałe metody wykraczają poza program studiów inżynierskich.

Na początku wprowadźmy podstawowe oznaczenia. Wiele zagadnień naukowych oraz inżynierskich prowadzi do układu równań liniowych $Ax=b$, który w formie macierzowej przyjmuje postać:

$$\mathbf {A x} ={ \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} } \cdot \begin{bmatrix} x_1 \ x_2 \ \vdots \ x_n \end{bmatrix} } =\left({ \begin{array}{rrrr}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{array}}\right) \left({ \begin{array}{c} x_1 \ x_2 \ \vdots \ x_n \end{array} }

\right)

\begin{bmatrix} b_1 \ b_2 \ \vdots \ b_n \end{bmatrix}\tag{2.1.1} $$

> Przykład 2.1
>
> Rozważmy przykład techniczny obwodu elektrycznego zaprezentowanego na rysunku 2.1. Naszym zadaniem jest przedstawić w zapisie macierzowym układ równań pozwalający znaleźć rozkład prądów w obwodzie. Należy wykorzystać prawa Kirchhoffa. Parametry obwodu to: I=10[A], E1=5[V], E2=8[V], R1=5[Ω], R2=5[Ω], R3=3[Ω], R4=7[Ω], R5=2[Ω].
>
> image-20221021185329183
>
> Rysunek 2.1 Obwód elektryczny zbudowany z trzech oczek, zawierający pięć rezystancji, dwa źródła napięcia E_1, E_2 oraz jedno źródło prądu I.
>
> Z równań Kirchhoffa otrzymujemy pięć równań. Pierwsze trzy równania przedstawiają bilans prądów w węzłach, a pozostałe dwa bilans napięć w oczkach. Układ pięciu niezależnych liniowo równań zawiera pięć niewiadomych i posiada jednoznaczne rozwiązanie.

$$ \begin{split} I_1 + I_4 &= I \ I_3 + I_4 &= I_5 \ I_2 + I_3 &= I_1 \ R_1 I_1 + R_3 I_3 - R_4 I_4 &= E_1 \ R_2 I_2 - R_3 I_3 - R_5 I_5 &= -E_2 \end{split} \label{eq:obwod_algebraicznie}\tag{2.1.2} $$

> Powyższy układ równań zapisany algebraicznie możemy przekształcić do postaci macierzowej:

$$ \begin{bmatrix} 1 & 0 & 0 & 1 & 0 \ 0 & 0 & 1 & 1 & -1 \ -1 & 1 & 1 & 0 & 0 \ R_1 & 0 & R_3 & -R_4 & 0 \ 0 & R_2 & - R_3 & 0 & -R_5 \end{bmatrix} \begin{bmatrix} I_1 \ I_2 \ I_3 \ I_4 \ I_5

\end{bmatrix}

\begin{bmatrix} I \ 0 \ 0 \ E_1 \ -E_2 \end{bmatrix} \label{obwod:macierzowo}\tag{2.1.3} $$

> Ostatecznie po podstawieniu wartości liczbowych otrzymujemy układ równań przedstawiony w równaniu $\ref{obwod:liczbowo}$.

$$ \begin{bmatrix} 1 & 0 & 0 & 1 & 0 \ 0 & 0 & 1 & 1 & -1 \ -1 & 1 & 1 & 0 & 0 \ 5 & 0 & 3 & -7 & 0 \ 0 & 5 & -3 & 0 & -2 \end{bmatrix} \begin{bmatrix} I_1 \ I_2 \ I_3 \ I_4 \ I_5

\end{bmatrix}

\begin{bmatrix} 10 \ 0 \ 0 \ 5 \ -8 \end{bmatrix} \label{obwod:liczbowo}\tag{2.1.3} $$

2.2 Normy wektorów

W algebrze liniowej, analizie funkcyjnej i pokrewnych dziedzinach matematyki, norma to funkcja przyporządkowująca dodatnią wartość liczbową określającą długość lub wielkość wektora, lub przestrzeni wektorowej gdy obliczamy normę macierzy. W ogólności normy są wykorzystywane do określania odległości między punktami wskazywanymi przez wektory oraz porównywania ich długości. Istnieje wiele różnych rodzajów norm (różnych funkcji), które mogą spełnić to zadanie. Aby funkcja przekształcająca wektor lub macierz w liczbę mogła być nazwana normą, to musi ona posiadać następujące właściwości:

1. $||av|| = |a| ||v||$ (skalowalność)
2. $||u + v|| \le ||u|| + ||v||$ (nierówność trójkątna)
3. $|v|| \ge 0$ (nieujemność)
4. Jeżeli $||v|| = 0$ to $v=0$, czyli v jest wektorem zerowym (jednoznaczność),

gdzie a to dowolna wartość skalarna, v i u to dowolne wektory. W dalszej części przedstawimy kilka najczęściej wykorzystywanych funkcji, które charakteryzują się przed chwilą wymienionymi właściwościami, zatem są normami macierzy:

Norma euklidesowa (L2-norm)

To najbardziej intuicyjna norma wśród norm wektorowych, która stanowi euklidesową długość wektora, czyli pierwiastek sumy kwadratów jego składowych x_i: $$ \left|{\boldsymbol {X}}\right|_{2}:={\sqrt {x_{1}^{2}+\cdots +x_{n}^{2}}}\label{eq:norma_euklidesowa}\tag{2.1.4} $$

p-norma

p-norma to funkcja zwracająca wartość skalarną zdefiniowaną jako pierwiastego p-tego stopnia sumy składowych wektora podniesionych do p-tej potęgi. W tym kontekście, zdefiniowana wcześniej norma euklidesowa jest szczególnym przypadkiem p-normy dla $p=2$. $$ \left| \mathbf {x} \right|_{p}:={\bigg (}\sum *{i=1}^{n}\left|x{i}\right|^{p}{\bigg)}^{1/p}\label{eq:pnorma}\tag{2.1.5} $$

Norma nieskończoność

Przykład klasycznej normy, w której funkcja przyporządkowuje wektorowi wartość maksymalnej składowej x_i. Zaletą tej normy jest jest bardzo prosta i szybka implementacja oraz brak wrażliwości na błędy operacji arytmetycznych. $$ {\displaystyle \left|\mathbf {x} \right|_{\infty }:=\max *{i}\left|x{i}\right|.}\tag{2.1.6} $$ Na rysunku 2.2 przedstawiona została interpretacja graficzna wymienionych norm indukowanych w przestrzeni wektorów dwuwymiarowych. Przedstawiono na nim „koła jednostkowe", stanowiące zbiory punktów znajdujących się na końcach wszystkich wektorów, które dla poszczególnych zdefiniowanych norm uzyskują wartość równą jeden. Wektory są zaczepione w początku układu współrzędnych.

image-20221021185621838

Rys. 2.2. "Koła jednostkowe" dla trzech definicji norm: L_1 dla p-normy i $p=1$, L_2 dla normy euklidesowej oraz L_{∞} dla normy nieskończoność.

> [Rysunek dla normy L_1 przedstawia obrócony o $45\degree$ kwadrat, którego wierzchołki są na osiach współrzędnych ox i oy, a jego środek ciężkości w początku układu współrzędnych. Środkowy rysunek dla normy L_2 przedstawia okrąg ze środkiem w początku układu współrzędnych. Trzeci, dolny rysunek dla normy L_∞ przedstawia kwadrat ze środkiem ciężkości w początku układu współrzędnych i bokami równoległymi do osi ox i oy.]

2.3 Normy macierzowe

Macierze często są nazywane operatorami liniowymi, które przekształcają wektory, np. za pomocą operacji $x'=Ax$. Przekształceniem może być wydłużenie lub skrócenie wektora. W tym kontekście, normy macierzowe są funkcjami, które przyporządkowują danej macierzy liczbę skalarną wyrażającą zdolność macierzy do wydłużania wektorów. Każda norma wektorowa pozwala nam zdefiniować normę macierzową, która wyraża maksymalne wydłużenie wektora jednostkowego w danej normie po przekształceniu przez macierz A. Takie normy nazywa się normami indukowanymi. Normy indukowane, zatem charakteryzują jak dana macierz A rozciąga / przekształca wektory jednostkowe w odniesieniu do

danej normy. Wyraża to równanie $\ref{norma:indukowana}$, które normie przyporządkowuje maksymalne wydłużenie dowolnego wektora $ \mathbf{x}$ przez macierz A. $$ ||A||*p = \max{||\mathbf{x}||p \ne 0}\frac{ ||A \mathbf{x}||_p}{ ||\mathbf{x} ||_p} = \max{||\mathbf{x}||_p = 1} || A\mathbf{x}||_p\tag{2.3.1}\label{norma:indukowana} $$ Podobnie do norm wektorowych, normy macierzowe spełniają podstawowe kryteria pozwalające na ich wykorzystanie do porównywania macierzy. Dla skalarów α w K oraz wszystkich macierzy A oraz B w przestrzeni $K^{m \times n}$,

- $|\alpha A|=|\alpha ||A|$ (skalowalność)
- $|A+B|\leq |A|+|B|$ (nierówność trójkątna)
- $|A|\geq 0$ (nieujemność)
- $|A|= 0 \text{ jeżeli } A=0_{m,n}$ (jednoznaczność)

Na rysunku 2.3 przedstawiono graficzną interpretację przekształcenia zbioru "kół jednostkowych" dla norm L_1, L_2 oraz L_{∞} przez przykładową macierz: $$ A = \begin{bmatrix}1 & 2 \ 0 & 2\end{bmatrix}\notag $$

image-20230106105700167

Rys. 2.3. "Koła jednostkowe" dla trzech definicji norm: L_1 dla p-normy i $p=1$, L_2 dla normy euklidesowej oraz L_{∞} dla normy nieskończoność.

Normy macierzowe są naturalnym rozszerzeniem notacji norm dedykowanej dla wektorów. Na początku wymienimy p-normy indukowane. **Norma L_1-maksimum** $$ |A|_{1}=\max *{1\leq j\leq n}\sum _{i=1}^{m}|a*{ij}|\label{norma:max}\tag{2.3.2} $$

Norma L_{∞}-nieskończoność $$ |A|_{\infty }=\max *{1\leq i\leq m}\sum _{j=1}^{n}|a*{ij}|\label{norma:nieskonczonosc}\tag{2.3.3} $$

Norma L_2 $$ \sigma_{\max}(A) = \sqrt{\max \lambda_i\label{norma:l2}}\tag{2.3.4} $$ gdzie $\sigma_{\max}$ to maksymalna wartość osobliwa macierzy A oraz λ_i oznacza wartości własne macierzy A^HA (gdzie A^H to macierz Hermitowska, czyli transponowana macierz sprzężona spełniająca warunek $[a_{ij}] = \bar{[a_{ji}]}$).

Norma Frobeniusa $$ |A|_{2}=\sigma *{\max }(A)\leq \left(\sum _{i=1}^{m}\sum _{j=1}^{n}|a*{ij}|^{2}\right)^{1/2}=|A|_{\rm {F}}\label{norma:frobeniusa}\tag{2.3.5} $$

2.4 Współczynnik uwarunkowania

Powszechnym zastosowaniem norm macierzowych jest obliczanie współczynnika uwarunkowania macierzy. Współczynnik uwarunkowani macierzy $cond(A)$ informuje za pomocą liczby skalarnej jak bardzo dana macierz jest podatna na niedokładności danych wejściowych. Danymi wejściowymi są w tym przypadku wartości samej macierzy A oraz wartości wektora prawych stron b układu równań $Ax=b$. Współczynnik uwarunkowania określa jak bardzo zmiany wartości liczbowych przenoszą się na rozwiązanie układu równań.

Załóżmy, że e jest błędem wektora prawych stron b układu równań $Ax=b\pm e$, wówczas rozwiązanie jest również obarczone tym błędem i wyraża się równaniem: $$ x + \Delta x = A^{-1}b + A^{-1}e \notag $$ zatem stosunek względnego błędu rozwiązania $\frac{||A^{-1}b||}{||A^{-1}e||}$ do względnego błędu b $\frac{||e||}{||b||}$ wyraża się: $$ \frac {\frac {\left|A^{-1}e\right|}{\left|A^{-1}b\right|}}{\frac {|e|}{|b|}} = \frac {\Vert A^{-1}e\Vert /\Vert A^{-1}b\Vert }{\Vert e\Vert /\Vert b\Vert } = \frac {\left|A^{-1}e\right|}{|e|}{\frac {|b|}{\left|A^{-1}b\right|}}\notag $$ Wartość maksymalna tego stosunku może więc postrzegana jako iloczyn dwóch norm, zgodnie z poniższym przekształceniem: $$ \begin{aligned}\max *{e,b\neq 0}\left{{\frac {\left|A^{-1}e\right|}{|e|}}{\frac {|b|}{\left|A^{-1}b\right|}}\right}&=\max *{e\neq 0}\left{{\frac {\left|A^{-1}e\right|}{|e|}}\right}\,\max *{b\neq 0}\left{{\frac {|b|}{\left|A^{-1}b\right|}}\right}\&=\max *{e\neq 0}\left{{\frac {\left|A^{-1}e\right|}{|e|}}\right}\,\max *{x\neq 0}\left{{\frac {|Ax|}{|x|}}\right}\&=\left|A^{-1}\right|\,|A|.\end{aligned}\notag $$ Wartość maksymalna jest określana właśnie współczynnikiem uwarunkowania: $ $ cond(A)=\left|A^{-1}\right|\,\left|A\right|\geq \left|A^{-1}A\right|=1\label{wspolczynnik_uwarunkowania}\tag{2.3.6} $$ Współczynnik uwarunkowania o wartości 1 nie wprowadza zatem żadnego dodatkowego błędu rozwiązania. Rozwiązanie jest zatem nie gorsze od samych danych wejściowych. Wartość współczynnika uwarunkowania zależy od przyjętych norm. Najczęściej stosowane definicje to:

- współczynnik uzależniony od wartości osobliwych gdy zastosujemy normę L_2 $$ cond (A)={\frac {\sigma _{\text{max}}(A)}{\sigma _{\text{min}}(A)}}\notag $$
- współczynnik uzależniony od wartości minimalnych i maksymalnych gdy macierz jest trójkątna i zastosujemy normę L_{∞} $$ cond(A) \geq {\frac {\max {i}{\big (}|a{ii}|{\big)}}{\min {i}{\big (}|a{ii}|{\big)}}}\notag $$

Przykład 2.2

Przeanalizujmy wpływ niedokładności danych wejściowych dla przykładowego układu równań wraz z obliczeniem współczynnika uwarunkowania. Rozważamy układ równań $Ax=b$ $$ \mathbf {A} ={\begin{bmatrix}1&2\2&3{,}999\end{bmatrix}};\mathbf {b} ={\begin{bmatrix}4\7{,}999\end{bmatrix}}\notag $$ Współczynnik uwarunkowania tej macierzy możemy obliczyć wyznaczając go z normy L_{∞} w następujących krokach: $$ \begin{align} cond(A) &= ||A^{-1}|| ||A|| \notag \ ||A||_{\infty} &= \max_i \sum_j |a_{ij}| = 5,999 \notag \ A^{-1}=\frac{1}{det(A)} A^D &= \frac{1}{3.999-2\cdot2} \begin{bmatrix}3.999 & -2 \ -2 & 1\end{bmatrix} = \begin{bmatrix}-3999 & 2000 \ 2000 & -1000\end{bmatrix} \notag \ ||A^{-1}||_{\infty} = \max_i \sum_j |a_{ij}| = 5999 \notag \ cond(A) &= ||A^{-1}|| ||A|| = 5,999 * 5999 = 35988 \notag \end{align} $$ Współczynnik uwarunkowania dla tej macierzy o wymiarach 2×2 jest stosunkowo bardzo duży. Oznacza to, że spodziewamy się znacznego wpływu niedokładności na wynik rozwiązania. Sprawdźmy to.

Obliczmy rozwiązanie układu równań przy założeniu oryginalnego wektora $b=\begin{bmatrix}4 \\ 7,999\end{bmatrix}$. Wówczas rozwiązanie układu równań możemy obliczyć odejmując pierwszy wiersz od drugiego: $$ \begin{align} x_1 + 2x_2 &= 4 \hspace{2cm}| -(\cdot 2) \ 2x_1 + 3,999x_2 &= 7,999 \end{align} $$

$$ \begin{align*} -2 x_1 - 4x_2 &= -8 \ 2x_1 + 3,999x_2 &= 7,999 \

- -------&---- \ -0,001x_2 & = -0,001 \hspace{2cm} \rightarrow x_2 = 1 \end{align*} $$

następnie $$ x_1 = 4-2x_2 = 4 - 2\cdot1=2\notag $$ otrzymujemy zatem rozwiązanie: $$ x = \begin{bmatrix}2 \ 1\end{bmatrix}\notag $$ Zmieńmy teraz nieznacznie wektor prawych stron b i przyjmijmy: $b=\begin{bmatrix}4 \\ 8\end{bmatrix}$. Wówczas analogicznie przeprowadzająca oblcizenia jak powyżej otrzymamy wynik: $$ x = \begin{bmatrix}4 \0 \end{bmatrix}\notag $$ Jak można zauważyć, niewielka zmiana o wartość $0,001$ wektora b spowodowała bardzo dużą zmianę rozwiązania. **Mówimy o takim układzie równań, że jest źle uwarunkowany.**

Zmieńmy współczynniki macierzy A: $$ \mathbf {A} ={\begin{bmatrix}1&2\2&3\end{bmatrix}};\mathbf {b} ={\begin{bmatrix}4\7{,}999\end{bmatrix}}\notag $$ Gdybyśmy powtórzyli obliczenia współczynnika uwarunkowania dla tej macierzy to otrzymalibyśmy wartość $cond(A)=25$, która jest mniejsza od poprzedniej o ponad 100 razy (dwa rzędy). Widać, że macierz ta jest znacznie lepiej uwarunkowana. Powtórzmy serię obliczeń dla oryginalnego ${b} ={\begin{bmatrix}4\7{,}999\end{bmatrix}}$ oraz zmodyfikowanego $b=\begin{bmatrix}4 \\ 8\end{bmatrix}$ i porównajmy wyniki.

dla ${b} ={\begin{bmatrix}4\7{,}999\end{bmatrix}}$ otrzymamy wynik $\mathbf {x} ={\begin{bmatrix}3{,}998\0{,}001\end{bmatrix}}$,

a dla $b=\begin{bmatrix}4 \\ 8\end{bmatrix}$ otrzymamy $\mathbf {x} ={\begin{bmatrix}4\0\end{bmatrix}}$.

Jak widać, tym razem wyniki rozwiązania różnią się nieznacznie.

2.5 Metoda eliminacji Gaussa

Eliminacja Gaussa jest podstawowym sposobem rozwiązywania układu równań linowych. W eliminacji Gaussa wykorzystujemy operacje wierszowe dodawania do siebie wierszy układu równań. Wykonujemy jednak te operacje w ściśle określony sposób i kolejności. Zasadniczym celem eliminacji Gaussa jest przekształcenie układu równań do postaci górno-trójkątnej. To znaczy, że wszystkie współczynniki macierzy A układu równań poniżej diagonali są zerowane. Uważny czytelnik zauważył zapewne, że mówimy o eliminacji elementów macierzy A, ale operacje wykonujemy na całych wierszach układu równań. Nie możemy zatem zapomniec o wektorze prawych stron b. W celu uproszczenia implementacji algorytmu zazwyczaj tworzy się zazwyczaj macierz rozszerzoną, która powstaje poprzez dołączenie wektora prawych stron b dodatkowej kolumny do macierzy A. $$ [Ag] = [A | b]\label{macierz:rozszerzona}\tag{2.5.1} $$

Następnie przekształcamy macierz Ag do postaci górno-trójkątnej za pomocą dodawania wierszy wierszy 'diagonalnych' wymnożonych przez stosowne współczynniki skalujące (odejmujemy wiersze ‘diagonalne’ od wierszy poniżej). Kolejność operacji (zaprezentowana na rysunku 2.4) jest następująca:

- najpierw odejmujemy pierwszy wiersz macierzy Ag od drugiego wiersza wymnożony przez współczynnik, który spowóduje po odjęciu wyzerowanie elementu a_{21}, równy $l_{21}=\left(\frac{a_{21}}{a_{11}}\right)$,
- w kolejnych kilku krokach odejmuje pierwszy wierz macierzy wymnożony przez stosowne współczynniki, aż wyzerowane zostaną wszystkie elementy poniżej diagonali,
- dalej, przechodzimy do zerowania elementów poniżej diagonali w drugiej kolumnie,
- itd., aż wyzerujemy wszystkie elementy poniżej diagonali.

Powyższy schemat został zilustrowany na rysunku 2.4.

image-20230106120225859

Rysunek 2.4 Ilustracja przebiegu eleminacji Gaussa

Przykład 2.3

Przeprowadź eliminację Gaussa dla poniższego układu równań z trzema niewiadomymi. $$ \begin{align} x_1 + 3x_2 + 4x_3 &= 2 \ -2x_1 + 2x_2 + 3x_3 &= -1 \ x_1 + x_2 + 2x_3 &= 3 \ \end{align} $$ Powyższy układ równań możemy zapisać w postaci macierzowej: $$ \begin{bmatrix} 1 & 3 & 4 \ -2 & 2 & 3 \ 1 & 1 & 2 \end{bmatrix} \cdot \begin{bmatrix} x_1 \ x_2 \ x_3

\end{bmatrix}

\begin{bmatrix} 2 \ -1 \ 3 \end{bmatrix}\notag $$ Następnie możemy odjąć pierwszy wiersz od kolejno drugiego i trzeciego: $$ \begin{bmatrix} 1 & 3 & 4 \ -2 - \left(\frac{-2}{1}\right)1 & 2-\left(\frac{-2}{1}\right)3 & 3-\left(\frac{-2}{1}\right)4 \ 1 - \left(\frac{1}{1}\right)1 & 1 - \left(\frac{1}{1}\right)3 & 2 - \left(\frac{1}{1}\right) 4 \end{bmatrix} \cdot \begin{bmatrix} x_1 \ x_2 \ x_3

\end{bmatrix}

\begin{bmatrix} 2 \ -1-\left(\frac{-2}{1}\right)2 \ 3 - \left(\frac{1}{1}\right) 2 \end{bmatrix}\notag $$ W wyniku otrzymamy: $$ \begin{bmatrix} 1 & 3 & 4 \ 0 & 8 & 11 \ 0 & -2 & -2 \end{bmatrix} \cdot \begin{bmatrix} x_1 \ x_2 \ x_3

\end{bmatrix}

\begin{bmatrix} 2 \ 3 \ 1 \end{bmatrix}\notag $$ Kolejnym krokiem jest odjęcie drugiego wiersza od trzeciego w celu eliminacji elementu a_{32}: $$ \begin{bmatrix} 1 & 3 & 4 \ 0 & 8 & 11 \ 0 & -2- \left(\frac{-2}{8}\right) 8 & -2 - \left(\frac{-2}{8}\right) 11 \end{bmatrix} \cdot \begin{bmatrix} x_1 \ x_2 \ x_3

\end{bmatrix}

\begin{bmatrix} 2 \ 3 \ 1- \left(\frac{-2}{8}\right) 3 \end{bmatrix}\notag $$ Ostatecznie otrzymuje górno-trójkątną postać układu równań: $$ \begin{bmatrix} 1 & 3 & 4 \ 0 & 8 & 11 \ 0 & 0 & \frac{3}{4} \end{bmatrix} \cdot \begin{bmatrix} x_1 \ x_2 \ x_3

\end{bmatrix}

\begin{bmatrix} 2 \ 3 \ 1\frac{3}{4} \end{bmatrix}\notag $$

W celu implementacji komputerowej algorytmu eliminacji Gaussa warto posłużyć się zapisem algorytmicznym procedury. Zakładając, że operujemy na macierzy rozszerzonej ${\mathbf{Ag}}*{n \times n+1} = \left(a*{i,j} \right| a_{i,n+1} = b_{i}\right) \text{ dla } i,j = 1,2,\ldots,n$ o wymiarach $n \times n+1$, $$ a_{i,j}^{k} = a_{i,j}^{(k-1)} - \frac{a_{i,k}^{(k-1)}}{a_{k,k}^{(k-1)}}a_{k,j}^{(k-1)}\label{gauss}\tag{2.5.2} $$ dla $$ \begin{align} k &= 1,2, \ldots, n-1 \text{ (kolejne kolumny bez ostatniej)} \ i &= k+1, k+2, \ldots, n \text{ (kolejne wiersze poniżej k-tego)} \ j &= k,k+1,\ldots, n+1 \text{ (wszystkie kolumny z pominięciem już wczesniej wyzerowanych)} \ \end{align} $$ Tak sformułowany algorytm pozwala nam wygodnie zaimplementować funkcje w MATLAB, która przyjmuje jako argument macierz A oraz wektor prawych stron b, a zwraca macierz rozszerzoną Ag, która jest górno-trójkątna. Proszę zwrócić uwagę, na bardzo podobne oznaczenia, które ułatwiają interpretację kodu.

```
function Ag = gaussian(A,b)
  Ag = [A b]
  n = size(Ag,1);
  for k=1:n-1
    for i = k+1:n
      l = Ag(i,k) / Ag(k,k);
      for j = k:n+1
        Ag(i,j) = Ag(i,j) - l * Ag(k,j);
      end
    end
  end
end
```

Jednym z bardzo ważnych elementów, których nie możemy pominąć jest wrażliwość algorytmu na wystąpienie w dowolnym momencie procesu zera na diagonali, które będzie skutkowało dzieleniem przez zero podczas obliczania współczynniki wykorzystywanego do eliminacji. W celu eliminacji tego problemu stosuje się **selekcję elementu głównego**.

Selekcja elementu głównego polega na takiej zamianie wierszy macierzy rozszerzonej Ag, aby w kolejnym kroku algorytmu na diagonali znalazł się **maksymalny co modułu element**, jednocześnie pozostawiając już wyzerowane elementy nadal zerowymi.

Występują trzy rodzaje selekcji:

1. w kolumnie (selekcja częściowa)- najprostsza wymaga tylko zamiany równań (rysunek 2.5a),
2. w wierszu (selekcja częściowa) - zamieniamy kolejność niewiadomych w wektorze x (rysunek 2.5b),

3. podmacierz A[r:end, c:end] (selekcja pełna) - zamieniamy zarówno wiersze jak i kolejność zmiennych w wektorze x (rysunek 2.5c).

image-20230106140428588

Rys. 2.5 Elementy macierzy, w których poszukuje się wartości maksymalnych co do modułu w przypadku procedury selekcji elementu głównego: a) dla selekcji częściowej w kolumnie, b) dla selekcji częściowej w wierszu, c) dla selekcji pełnej w bloku macierzy.

Przykład 2.4

Rozwiążemy układ równań stosując skończoną precyzję sztucznie zaokrąglając wyniki obliczeń arytmetycznych aby pokazać wpływ selekcji elementu głównego również na dokładność rozwiązania.

Na początku rozwiążmy układ z następującym układem wierszy.

$\begin{bmatrix} 0.003 & 59.1 \ 5.291 & -6.13 \end{bmatrix} \begin{bmatrix} x_1 \ x_2
\end{bmatrix}$

\begin{bmatrix} 59.17 \ 46.78 \end{bmatrix}$

$m = \frac{5.291}{0.003} = 1763.666... \approx = 1764.0$

$\begin{bmatrix} 0.003 & 59.1 \ 0 & -104200 \end{bmatrix} \begin{bmatrix} x_1 \ x_2
\end{bmatrix}$

\begin{bmatrix} 59.17 \ -104300 \end{bmatrix}$

$x_2 = \frac{-104300}{-104200} \approx 1.001$

$x_1 = \frac{59.17 - 59.1 \cdot 1.001}{0.003} \approx 3.633$

Teraz, powtórzmy obliczenia, ale wcześniej zamieniając miejscami wiersz pierwszy i drugi układu równań.

$\begin{bmatrix} 5.291 & -6.13 \ 0.003 & 59.1 \end{bmatrix} \begin{bmatrix} x_1 \ x_2
\end{bmatrix}$

\begin{bmatrix} 46.78 \ 59.17 \end{bmatrix}$

$m = \frac{0.003}{5.291} \approx 0.000567$

$\begin{bmatrix} 5.291 & -6.13 \ 0 & 59.1 \end{bmatrix} \begin{bmatrix} x_1 \ x_2
\end{bmatrix}$

\begin{bmatrix} 46.78 \ 59.14 \end{bmatrix}$

$x_2 = \frac{59.14}{59.1} \approx 1.001$

$x_1 = \frac{46.78 - (-6.13) \cdot 1.001}{5.291} \approx 10.001$

Widzimy, że zmiana wierszy (zaznaczmy, że przy zastosowaniu sztucznie zawyżonego błędu zaokrągleń) dała nam wynik znacznie inny od poprzedniego. Który wynik jest poprawny? Ten drugi, ponieważ na diagonali znajdował się największy co do modułu w danej kolumnie element.

Wykonanie tych obliczeń w MATLABie (nawet z oryginalnym układem wierszy) daje nam wynik drugi. Po pierwsze, MATLAB zamienia wiersze automatycznie, po drugie operacje wykonane są z dużo większą precyzją ($eps\approx 10^{-14}$).

```
A = [0.003 59.1;
     5.291 -6.13];
b = [59.17
     46.78];
A\b

ans =

   10.0008
    1.0007
```

2.6 Wsteczne podstawienie

Jeżeli macierz rozszerzona reprezentująca nasz układ równań jest już w postaci trójkątnej, to wynik rozwiązania bardzo łatwo znaleźć stosując

procedurę wstecznego podstawienia. Przyjrzyjmy się przykładowemu układowi równań w postaci górnotrójkątnej: $$ \begin{bmatrix} a_{11} & a_{12} & a_{13} & b_1 \ 0 & a_{22} & a_{23} & b_2 \ 0 & 0 & a_{33} & b_3 \end{bmatrix}\notag $$ Mając taką postać, procedura kolejno znajduje wartości x_i zaczynając od ostatniego, czyli dla $i=n,n-1,\ldots,1$. Otrzymujemy zatem: $$ \begin{align} x_3 &= \frac{b_3}{a_{33}}\ x_2 &= \frac{b_2 - a_{23}\cdot x_3}{a_{22}}\ x_1 &= \frac{b_1 - a_{13}\cdot x_2 - a_{13}\cdot x_3}{a_{11}}\ \end{align} $$ Zauważmy, że w kolejnych wierszach wykorzystujemy wartości x_i obliczone wcześniej. Powyższą procedurę możemy uogólnić dla macierzy górno-trójkątnej o dowolnym rozmiarze i przedstawić matematycznie: $$ x_i = \frac{ c_i - \sum\limits_{j=i+1}^{n} \left(u_{ij} \cdot x_j \right)}{u_{ii}} \text{ dla } i = n,n-1, \ldots,1\label{podstawienie:wsteczne}\tag{2.6.1} $$ a w przypadku macierzy dolno-trójkątnej, odwrotnie będziemy kolejno obliczać wartości zaczynając tym razem od pierwszego wiersza: $$ x_i = \frac{ c_i - \sum\limits_{j=1}^{i-1} \left(l_{ij} \cdot x_j \right)}{l_{ii}} \text{ dla } i = 1,2, \ldots,n\label{podstawienie:zstepujace}\tag{2.6.2} $$ Oto dodatkowo dwie funkcje implementujące oba podstawienia.

```
% U - macierz górnotrójkątna, c - wektor prawych stron
function x = wsteczne_gornotrojkatne(U,c)
    n = size(U,1);
    x = zeros(n,1);
    for i = n:-1:1
        s = 0;
        for j = i+1:n
            s = s + U(i,j)*x(j);
        end
        x(i) = (c(i) - s) / U(i,i);
    end
end

% U - macierz dolnotrójkątna, c - wektor prawych stron
function x = wsteczne_dolnotrojkatne(L,c)
    n = size(L,1);
    x = zeros(n,1);
    for i = 1:n
        s = 0;
        for j = 1:i-1
            s = s + L(i,j)*x(j);
        end
        x(i) = (c(i) - s) / L(i,i);
    end
end
```

2.7. Eliminacja Gaussa-Jordana

Eliminacja Gaussa-Jordana jest rozwinięciem eliminacji Gaussa o dodatkowe kroki. W eliminacji tej oprócz zerowania elementów poniżej diagonali układu równań, eliminujemy również elementy powyżej diagonali. Dodatkowo, skalujemy wartości we wszystkich wierszach, tak aby po eliminacji na diagonali były same jedynki. Pamiętając, że operacje, które wykonujemy są operacjami wierszowymi to rozwiązania układu równań oryginalnego i układu po eliminacji są takie same. Łatwo zauważyć, że po eliminacji Gaussa-Jordana, ponieważ w części macierzy A będziemy mieli wyzerowane wszystkie elementy poniżej i powyżej diagonali oraz na diagonali będą wartości 1, to ostatnia kolumna macierzy rozszerzonej będzie rozwiązaniem układu równań. Nie potrzebujemy zatem wstecznego podstawienia. Pozornie, algorytm wydaje się być szybszy obliczeniowo, ale z uwagi na konieczność wyzerowania elementów powyżej diagonali jego złożoność jest praktycznie taka sama jak eliminacji Gaussa połączonej z wstecznym podstawieniem. Niemniej występują sytuacje, że zastosowanie eliminacji Gaussa-Jordana jest korzystne. Jedną z nich jest np. konieczność obliczenia macierzy odwrotnej.

Przebieg algorytmu eliminacji Gaussa-Jordana jest następujący:

1. Inicjalizacja (definicja macierzy rozszerzonej) $$ \begin{align} a_{ij}^{(0)} &= a_{ij}\text{ dla }i=1,2,\ldots,n;j=1,2,\ldots,n \ a_{ij}^{(0)} &= b_{i}\text{ dla }i=1,2,\ldots,n;j=n+1 \end{align} $$
2. Normalizacja (znormalizować element diagonalny do wartości 1) $$ a_{kj}^{(k)} = \frac{a_{kj}^{(k-1)}}{a_{kk}^{(k-1)}} \text{ dla }j=k,k+1, \ldots,n+1\notag $$
3. Redukcja (zredukować wszystkie elementy pozadiagonalne w kolumnie k) $$ a_{ij}^{(k)} = a_{ij}^{(k-1)} - a_{ik}^{(k-1)} \cdot a_{kj}^{(k)} \text{ for }j=k,k+1,\ldots,n+1; i=1,2,\ldots,n (\text{and }i \neq k)\notag $$

Kroki 2-3 muszą zostać wykonane dla wszystkich kolumn $k=1,2,\dots,n$.

Przykład 2.5

Znajdź rozwiązanie układu równań przedstawionego w postaci macierzy rozszerzonej.

1. W pierwszym kroku podzielimy wszystkie elementy pierwszego wiersza przez wartość elementu diagonalnego 2.
2. W drugim kroku odejmiemy pierwszy wiersz od drugiego wymnożony przez 4.
3. W trzecim kroku odejmiemy pierwszy wiersz od trzeciego wymnożony przez 3.
4. W czwartym kroku podzielimy elementy w drugim wierszu przez wartość diagonalną równą 5.
5. W piątym kroku odejmiemy drugi wiersz od pierwszego wymnożony przez $\frac{-1}{2}$.
6. W szóstym kroku odejmiemy drugi wiersz od trzeciego wymnożony przez $\frac{7}{2}$.
7. ... kontynuując obliczenia otrzymamy ostateczny wynik w czwartej kolumnie macierzy rozszerzonej: $x=[1,2,4]$.

$$ \begin{align} &\begin{bmatrix} 2 & -1 & 1 & 4 \ 4 & 3 & -1 & 6 \ 3 & 2 & 2 & 15 \end{bmatrix} \longrightarrow \begin{bmatrix} 1 & \frac{-1}{2} & \frac{1}{2} & 2 \ 4 & 3 & -1 & 6 \ 3 & 2 & 2 & 15 \end{bmatrix} \longrightarrow \begin{bmatrix} 1 & \frac{-1}{2} & \frac{1}{2} & 2 \ 4 - 4\cdot 1 & 3 - 4\cdot \frac{-1}{2} & -1 - 4\cdot \frac{1}{2} & 6 - 4 \cdot 2 \ 3 & 2 & 2 & 15 \end{bmatrix} \longrightarrow \notag \ &\begin{bmatrix} 1 & \frac{-1}{2} & \frac{1}{2} & 2 \ 0 & 5 & -3 & -2 \cdot 2 \ 3 & 2 & 2 & 15 \end{bmatrix} \longrightarrow \begin{bmatrix} 1 & \frac{-1}{2} & \frac{1}{2} & 2 \ 0 & 5 & -3 & -2 \cdot 2 \ 0 & \frac{7}{2} & \frac{1}{2} & 9 \end{bmatrix} \longrightarrow \begin{bmatrix} 1 & \frac{-1}{2} & \frac{1}{2} & 2 \ 0 &1 & \frac{-3}{5} & \frac{-2}{5} \cdot \frac{2}{5} \ 0 & \frac{7}{2} & \frac{1}{2} & 9 \end{bmatrix} \longrightarrow \ &\begin{bmatrix} 1 & 0 & \frac{1}{5} & \frac{9}{5} \ 0 &1 & \frac{-3}{5} & \frac{-2}{5} \cdot \frac{2}{5} \ 0 & \frac{7}{2} & \frac{1}{2} & 9 \end{bmatrix} \longrightarrow \begin{bmatrix} 1 & 0 & \frac{1}{5} & \frac{9}{5} \ 0 &1 & \frac{-3}{5} & \frac{-2}{5} \cdot \frac{2}{5} \ 0 & 0 & \frac{13}{5} & \frac{52}{5} \end{bmatrix} \longrightarrow \ldots \longrightarrow \begin{bmatrix} 1 & 0 & 0 & 1 \ 0 &1 & 0 & 2 \ 0 & 0 & 1 & 4 \end{bmatrix} \end{align} $$

Jak wcześniej wspominano, eliminację Gaussa-Jordana można skutecznie wykorzystać do obliczania macierzy odwrotnej o niewielkim rozmiarze. Aby wyprowadzić ten sposób przypomnijmy definicję macierzy odwrotnej: $$ \mathbf {A A^{-1}} ={ \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} \cdot \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} 1&0 &\cdots &0\ 0&1&\cdots &0\ \vdots &\vdots &\ddots &\vdots \ 0&0&\cdots &1 \end{bmatrix} \label{def:maceirz_odwrotna}\tag{2.7.1} $$ Zauważmy, że w tym równaniu pierwszą i kolejne kolumny macierzy A^{-1} możemy oznaczyć jako zmienne x_{ij}: $$ { \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} \cdot \begin{bmatrix} x'*{11}&x'*{12}&\cdots &x'*{1n}\ x'*{21}&x'*{22}&\cdots &x'*{2n}\ \vdots &\vdots &\ddots &\vdots \ x'*{m1}&x'*{m2}&\cdots &x'_{mn} \end{bmatrix}

}

\begin{bmatrix} 1&0 &\cdots &0\ 0&1&\cdots &0\ \vdots &\vdots &\ddots &\vdots \ 0&0&\cdots &1 \end{bmatrix} \notag $$ Przy takim zapisie, okazuje się, że aby znaleźć kolejne kolumny macierzy odwrotnej A^{-1} wystarczy rozwiązać n niezależnych układów równań, w których niewiadomymi będą kolumny macierzy A^{-1} a wektorami prawych stron kolejne kolumny macierzy jednostkowej. $$ { \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} \cdot \begin{bmatrix} x'*{11}\ x'{21}\ \vdots \ x'_{m1} \end{bmatrix}

}

\begin{bmatrix} 1\ 0\ \vdots \ 0 \end{bmatrix} \notag $$

$$ { \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} \cdot \begin{bmatrix} x'*{12}\ x'{22}\ \vdots \ x'_{m2} \end{bmatrix}

}

\begin{bmatrix} 0\ 1\ \vdots \ 0 \end{bmatrix} \notag $$

i tak dalej.

Możemy jednak podejść do tego zagadnienia skuteczniej i zamiast rozwiązywać n osobnych układów równań rozwiążemy jeden ale z wszystkimi wektorami prawych stron (całą macierzą jednostkową) doklejoną do macierzy A. $$ { \begin{bmatrix} a_{11}&a_{12}&\cdots &a_{1n} & 1&0 &\cdots &0\ a_{21}&a_{22}&\cdots &a_{2n} & 0&1&\cdots &0\ \vdots &\vdots &\ddots &\vdots & \vdots &\vdots &\ddots &\vdots\ a_{m1}&a_{m2}&\cdots &a_{mn} & 0&0&\cdots &1 \end{bmatrix} } \notag $$ Wykorzystamy do tego metodę eliminacji Gaussa-Jordana, w wyniku której przekształcimy układ do postaci $$ { \begin{bmatrix} 1&0 &\cdots &0 & a'*{11}&a'{12}&\cdots &a'*{1n} \ 0&1&\cdots &0 & a'*{21}

&a'{22}&\cdots &a'{2n} \ \vdots &\vdots &\ddots &\vdots & \vdots &\vdots &\ddots &\vdots\ 0&0&\cdots &1 & a'*{m1}&a'*{m2}&\cdots &a'*{mn} \end{bmatrix} \notag } $$ gdzie współczynniki $a'*{ij}$ stanowią współczynniki szukanej macierzy odwrotnej.

> Przykład 2.6
>
> Wykorzystując eliminację Gaussa-Jordana znajdź macierz odwrotną do A:
>
> ${\mathbf A} = \begin{bmatrix} 2 & 1 \ 4 & 5 \end{bmatrix}$
>
> Rozwiązanie
>
> $\left[\begin{array}{cc|cc} 2 & 1 & 1 & 0 \ 4 & 5 & 0 & 1 \end{array} \right]$ -> $\left[\begin{array}{cc|cc} 1 & \frac{1}{2} & \frac{1}{2} & 0 \ 4 & 5 & 0 & 1 \end{array} \right]$ -> $\left[\begin{array}{cc|cc} 1 & \frac{1}{2} & \frac{1}{2} & 0 \ 0 & 5 - 4 \cdot \frac{1}{2} & 0 - 4 \cdot \frac{1}{2} & 1 \end{array} \right]$ -> $ \left[\begin{array}{cc|cc} 1 & \frac{1}{2} & \frac{1}{2} & 0 \ 0 & 3 & -2 & 1 \end{array} \right]$ ->
>
> $\left[\begin{array}{cc|cc} 1 & \frac{1}{2} & \frac{1}{2} & 0 \ 0 & 1 & -\frac{2}{3} & \frac{1}{3} \end{array} \right]$ -> $\left[\begin{array}{cc|cc} 1 & 0 & \frac{1}{2} - \frac{1}{2}\left(\frac{-2}{3} \right) & -\frac{1}{3} \frac{1}{2} \ 0 & 1 & -\frac{2}{3} & \frac{1}{3} \end{array} \right]$ -> $ \left[\begin{array}{cc|cc} 1 & 0 & \frac{5}{6} & -\frac{1}{6} \ 0 & 1 & -\frac{2}{3} & \frac{1}{3} \end{array} \right]$
>
> Odpowiedź
>
> ${\mathbf A}^{-1} = \left[\begin{array}{cc} \frac{5}{6} & -\frac{1}{6} \ -\frac{2}{3} & \frac{1}{3} \end{array} \right]$

2.8 Rozkład na czynniki - faktoryzacja macierzy

Innym sposobem rozwiązywania układów równań jest wstępny rozkład na czynniki macierzy A. W algebrze liniowej stosuje się powszechnie kilka rodzajów faktoryzacji:

1. Faktoryzacja LU: $A=L\cdot U$, macierz L jest macierzą dolno-trójkątną, a macierz U górno-trójkątną,
2. Faktoryzacja QR: $A = Q\cdot R$, macierz Q jest macierzą ortonormalną $Q^{T}Q=1 \rightarrow Q^{-1}=Q^{T}$), macierz jest macierzą górno-trójkątną,
3. Faktoryzacja SVD: $A=SVD$, macierze S, V są ortogonalne, a D diagonalna.

W ramach niniejszego podręcznika przyjrzymy się wyłącznie faktoryzacji LU. Zakładając w właściwości górno i dolno-trójkątne macierzy L i U, oryginalny układ równań możemy zapisać następująco: $$ Ax=b \notag\ LUx=b \notag $$ Rozwiązanie możemy uzyskać dwukrotnie stosując wsteczne podstawienie. Najpierw podstawimy $Ux=y$, wówczas uzyskamy $$ Ly=b\notag $$ Macierz L jest dolno-trojkątna więc rozwiązanie tego równani możemy szybko znaleźć stosując wsteczne podstawienie od góry. Znając wynik y, możemy przejść do drugiego etapu, korzystając z wprowadzonego wcześniej podstawienia: $$ Ux=y\notag $$ Wykorzystując ponownie wsteczne podstawienie uzyskamy ostateczne rozwiązanie.

2.8.1 Metoda Doolitle'a

Pozostaje pytanie jak skutecznie znaleźć rozkład LU. Pierwszym podejściem jest zastosowanie **metody Doolitle'a**. Aby wyprowadzić tą metodę posłużmy się pełnym zapisem faktoryzacji LU: $$ A=LU\notag $$

$$ \begin{bmatrix} a_{11} & a_{12} & \ldots & a_{1n} \ a_{21} & a_{22} & \ldots & a_{2n} \ \vdots &\dots &\dots &\dots \ a_{n1} & a_{n2} & \ldots & a_{nn} \ \end{bmatrix} = \begin{bmatrix} 1 & 0 & \ldots & 0 \ l_{21} & 1 & \ldots & 0 \ \vdots &\dots &\dots &\dots \ l_{n1} & l_{n2} & \ldots & 1 \ \end{bmatrix} \begin{bmatrix} u_{11} & u_{12} & \ldots & u_{1n} \ 0 & u_{22} & \ldots & u_{2n} \ \vdots &\dots &\dots &\dots \ 0 & 0 & \ldots & u_{nn} \ \end{bmatrix} \notag $$ Zapis ten pozwoli nam wyprowadzić procedurę Doolitle'a bezpośrednio z definicji faktoryzacji. W metodzie tej kluczowa jest kolejność w jakiej będziemy wyznaczać współczynniki kolejno parami poszczególnych wierszy macierzy U i kolumn macierzy L.

1. Najpierw wyznaczymy współczynniki pierwszego wiersza macierzy U. Zgodnie ze schematem na rysunku 2.6 wartości współczynników macierzy A w pierwszym są równe:

$$ \begin{align} a_{11} = 1 u_{11} + 0 \cdot 0 + \ldots + 0 \cdot 0\ a_{12} = 1 u_{12} + 0 u_{22} + \ldots + 0 \cdot 0\ \ldots \ a_{1n} = 1 u_{1n} + 0 u_{2n} + \ldots + 0 u_{nn}\ \end{align} $$ Stąd błyskawicznie (z uwagi na większość zer), możemy wyznaczyć $u_{1i}=a_{1i}$ dla $i=1,2,\ldots,n$.

W kolejnym drugim kroku wyznaczymy współczynniki pierwszej kolumny macierzy L: $$ \begin{align} a_{21} &= l_{21} u_{11} + 1 \cdot 0 + \ldots + 0 \cdot 0 \longrightarrow l_{21} = \frac{a_{21}}{u_{11}}\ \ldots & \ a_{n1} &= l_{n1} u_{11} + l_{n2} \cdot 0 + \ldots + 1 \cdot 0 \longrightarrow l_{n1} = \frac{a_{n1}}{u_{11}} \ \end{align} $$ stąd łatwo wyprowadzić wyrażenia na wartości l_{i1} zakładając, że wartości u_11 została już obliczona w poprzednim kroku. W trzecim kroku algorytmu wrócimy do macierzy U, tym razem do drugiego wiersza, analogicznie wyprowadzając wzory na współczynniki z operacji mnożenia wiersza razy kolumna: $$ \begin{align} a_{22} = l_{21} u_{12} + 1 u_{22} + \ldots + 0 \cdot 0 \longrightarrow u_{22} = a_{22} - l_{21} u_{12} \ \ldots \ a_{2n} = l_{21} u_{1n} + 1 u_{2n} + \ldots + 0 u_{nn} \longrightarrow u_{2n} = a_{2n} - l_{21} u_{1n} \ \end{align} $$ i tak dalej.

image-20230106143609520

Rys. 2.6 Schemat z kolejnością obliczeń wierszy i kolumn macierzy L i U w faktoryzacji LU.

Wykorzystując ten schemat możemy określić algorytm faktoryzacji LU metodą Doolitle'a w następujący sposób:

1. pierwszy wiersz U kopiujemy z pierwszego wiersza A $u_{1i} = a_{1i}\text{ dla } i=1,2,\ldots,n$
2. pierwsza kolumna L obliczana jest za pomocą: $l_{i1} = a_{i1} / u_{11}\text{ dla } i=2,3,\ldots,n$
3. następnie dla każdej pary (i^{ty} wiersz U) oraz (i^{ta} kolumna L):

$$ \begin{align} \text{dla } i=2,3,\ldots,n& \ u_{ik} &= a_{ik} - \sum\limits_{j=1}^{i-1} \left(l_{ij}u_{jk}\right)\text{ for }k=i,i+1,\ldots,n \ l_{ki} &= \frac{\left(a_{ki} - \sum\limits_{j=1}^{i-1}l_{kj}u_{ji}\right)}{u_{ii}}\text{ for }k=i+1,i+2,\ldots,n \end{align} $$

Implementacja tego kodu w postaci funkcji MATLABa jest następująca.

```
function [L, U] = doolitle( A )
    n = size( A,1 );
    L = eye( n );  % inicjujemy macierz jednsotkowa, poniewaz zawsze na diagonali sa jedynki
    U = zeros( n ); % pusta (na razie) macierz gornotrojkatna
    U( 1, : ) = A( 1,: ) % kopiujemy pierwszy wiersz
    L( 2 : n,1 ) = A( 2 : n,1 ) / U( 1,1 ); % obliczamy pierwsza kolumne

    % wykonujemy parami obliczenia kolejno wierszy U i kolumn L
    for i = 2:n
        for k = i:n
            s = 0;
            for j=1:i-1
                s = s + L( i,j )*U( j,k );
            end
            U( i,k ) = A( i,k ) - s;
        end
        for k = i+1:n
            s = 0;
            for j=1:i-1
                s = s + L( k,j )*U( j,i );
            end
            L( k,i ) = (A( k,i ) - s) / U( i,i );
        end
    end
end
```

Przykład 2.7

Wykorzystując przykładową, powyższą funkcję MATLABa oraz wcześniej zdefiniowane funkcje do wstecznego podstawienia znajdź rozwiązanie układu równań: $$ \begin{bmatrix} 1 &-1 &1 &1\ 4 &3 &-1& 2\ 3& 2& 2& 5\ 8 &9 &5& 8 \end{bmatrix}\notag x = \begin{bmatrix} 4 \6 \15 \1 \end{bmatrix} $$ Rozwiązanie:

```
function L04_lu
A = [1 -1 1 1
    4 3 -1 2
    3 2 2 5
    8 9 5 8];
b = [4 6 15 1]'
[L, U] = doolitle( A )
A - L*U  % powinna byc macierz zerowa
y = wsteczne_dolnotrojkatne( L,b )
x = wsteczne_gornotrojkatne( U, y )

% sprawdzam rozwiazanie - norma powinna być zero
norm( A*x-b )
end

L =
    1.0000         0         0         0
    4.0000    1.0000         0         0
    3.0000    0.7143    1.0000         0
    8.0000    2.4286    3.5556    1.0000

U =
    1.0000   -1.0000    1.0000    1.0000
         0    7.0000   -5.0000   -2.0000
         0         0    2.5714    3.4286
         0         0         0   -7.3333

ans =
     0     0     0     0
     0     0     0     0
     0     0     0     0
```

2.8.2 Metoda eliminacji Gaussa

Drugim, najbardziej użytecznym z praktycznego punktu widzenia sposobem jest wykorzystanie eliminacji Gaussa. Znaczenie tego podejścia jest bardzo istotne, gdyż pozwala na selekcję elementu głównego w kolejnych krokach metody. Metoda Doolitle'a nie pozwala na to. Dzięki selekcji jesteśmy zabezpieczeni przed dzieleniem przez zero na diagonali oraz redukujemy błędy zaokrągleń.

Algorytm faktoryzacji LU z wykorzystaniem eliminacji Gaussa jest bardzo prosty do implementacji. Pomijając szczegóły związane z wyprowadzeniem (wyrazilibyśmy operacje wierszowe za pomocą operatorów macierzowych oraz metodą Gaussa-Jordana wyprowadziliśmy macierze odwrotne tych operatów) przedstawmy algorytm.

Faktoryzacja LU metodą eliminacji Gaussa przebiega zgodnie z procesem eliminacji, ale w trakcie procesu współczynniki l_{ij}, które używaliśmy do wymnożenia macierzy diagonalnych podczas zerowania elementów zapamiętujemy w odpowiednich miejscach i,j macierzy L. Macierz wynikowa eliminacji Gaussa staje się wynikową macierzą U.

Zatem, współczynnik l_{21} z rysunku 2.7 wstawiamy w miejsce $2,1$ macierzy docelowej L (patrz rysunek 2.8).

image-20230106153458673

Rys. 2.7 Ilustracja operacji odejmowania pierwszego wiersza macierzy w trakcie eliminacja Gaussa od wiersza drugiego, z zaznaczonym współczynnikiem l_{21}, który wykorzystywany jest do wstawienia do macierzy L

image-20230106153620984

Rys. 2.8 Ilustracja z zaznaczonym elementem l_{21} macierzy L

Implementacja faktoryzacji LU z wykorzystaniem eliminacji Gaussa w środowisku MATLAB (bez selekcji elementu głównego) została przedstawiona poniżej.

```
function [L, U] = lu_gaussian(A)
    n = size( A, 1 );
    L = eye( n );
    for j = 1:n-1
        for i = j+1:n
            f = A( i,j ) / A( j,j );
            A(i, : ) = A(i, : ) - f*A(j,: );
            L(i, j) = f; % tutaj zapamietujemy wspolczynnik
        end
    end
    U = A;
end
```

Przykład 2.8

Wykorzystaj implementację faktoryzacji LU na przykładowej macierzy losowej o rozmiarach 4×4. Sprawdź wynik faktoryzacji obliczając normę $||LU-A||$

Rozwiązanie:

```
function l05_lu_gaussian
    A = rand( 4 )
    [L, U] = lu_gaussian( A )
    norm( L * U - A, 2 )
end

>> L05_lu_gaussian
A =
    0.8147    0.6324    0.9575    0.9572
    0.9058    0.0975    0.9649    0.4854
    0.1270    0.2785    0.1576    0.8003
    0.9134    0.5469    0.9706    0.1419

L =
    1.0000         0         0         0
    1.1118    1.0000         0         0
    0.1559   -0.2972    1.0000         0
    1.1211    0.2676    3.5869    1.0000

U =
    0.8147    0.6324    0.9575    0.9572
         0   -0.6055   -0.0996   -0.5788
         0         0   -0.0212    0.4791
         0         0         0   -2.4948

ans =
   2.2204e-16
```

normy