



# METODY NUMERYCZNE:

## INSTRUKCJA 1

### Wprowadzenie

Rozwiązanie wielu problemów inżynierskich wymaga rozwiązania układów równań nieliniowych (wśród których choć jedno równanie jest równaniem nieliniowym). Dzisiejsze laboratorium będzie poświęcone metodzie Newtona-Raphsona pozwalającej rozwiązywać takie zagadnienia. W celu przypomnienia podstawowych zagadnień zaczniemy od problemu liniowego wypływającego ze statyki mechanizmu po uwolnieniu poszczególnych członów od więzów. Umiejętność rozwiązania zagadnienia liniowego jest nieodzownym elementem implementacji metody Newtona-Raphsona.

### 1 Problem liniowy

Rozważmy mechanizm pokazany na Rysunku 1 (strona 2) i uwolnijmy ten układ od więzów, uwydatniając siły w parach kinematycznych. Znana jest geometria układu oraz ciężary poszczególnych członów wynoszące  $G_{AB} = 25$ ,  $G_{BC} = 16$  oraz  $G_{CD} = 53$ .

Dla układu o zadanej na rysunku geometrii oraz ciężarach członów podanych powyżej równania równowagi wyglądają następująco:

$$\begin{aligned} R_{Ax} + R_{Bx} &= 0 \\ R_{Ay} + R_{By} &= G_{AB} \\ M_A - 1.5G_{AB} + 3R_{By} - 4R_{Bx} &= 0 \\ -R_{Bx} + R_{Cx} &= 0 \\ -R_{By} + R_{Cy} &= G_{BC} \\ -2G_{BC} - 3R_{Cx} + 4R_{Cy} &= 0 \\ -R_{Cx} + R_{Dx} &= 0 \\ -R_{Cy} + R_{Dy} &= G_{CD} \\ -G_{CD} + 7R_{Dx} + 2R_{Dy} &= 0 \end{aligned}$$

W postaci macierzowej układ równań zapisuje się następująco:

$$\begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -4 & 3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -3 & 4 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 7 & 2 \end{bmatrix} \begin{bmatrix} R_{Ax} \\ R_{Ay} \\ M_A \\ R_{Bx} \\ R_{By} \\ R_{Cx} \\ R_{Cy} \\ R_{Dx} \\ R_{Dy} \end{bmatrix} = \begin{bmatrix} 0 \\ 25 \\ 37.5 \\ 0 \\ 16 \\ 32 \\ 0 \\ 53 \\ 53 \end{bmatrix}$$

### Zadanie 1

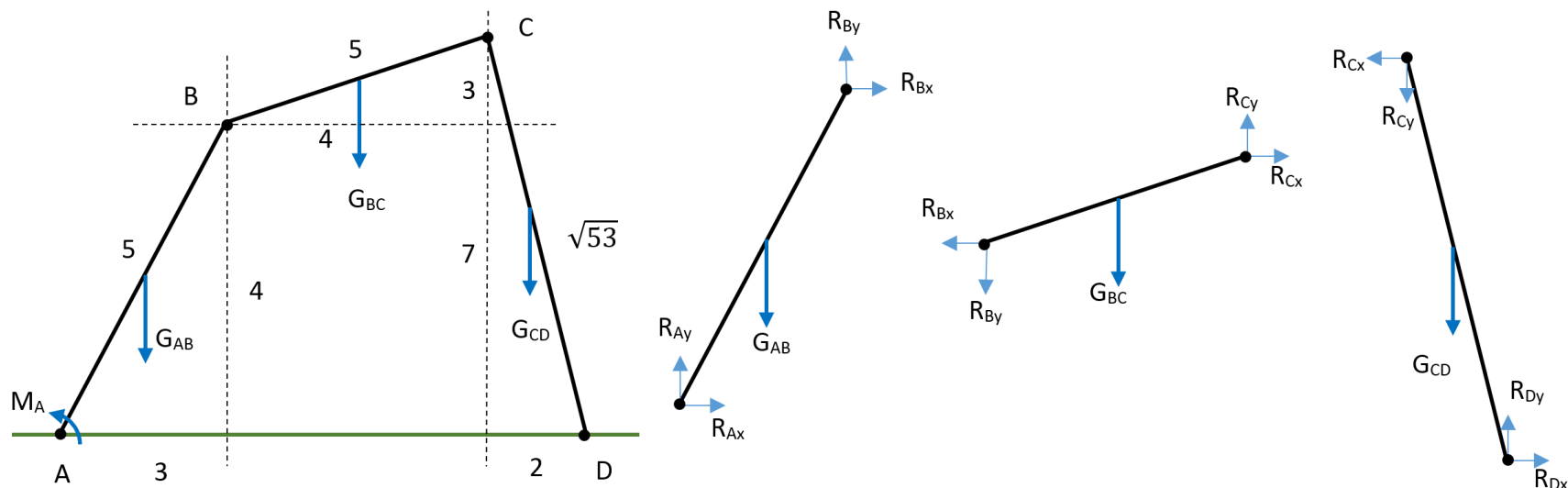
Napisz program w C, który obliczy siły i momenty przenoszone w parach kinematycznych. Do rozwiązania układu równań wykorzystaj metodę eliminacji Gaussa, której implementacja jest dostępna w pliku Gauss.h. (Wskazówka: Funkcja `void Gauss(int n, double **M, double *F, double *x)` przyjmuje podwójny wskaźnik do macierzy - z tego względu pamiętaj o zaalokowaniu dynamicznym dwuwymiarowej tablicy - tablica statyczna miałaby typ niezgodny z nagłówkiem funkcji). Sprawdź, czy otrzymujesz poprawne rozwiązanie wynoszące:

$$\begin{bmatrix} R_{Ax} \\ R_{Ay} \\ M_A \\ R_{Bx} \\ R_{By} \\ R_{Cx} \\ R_{Cy} \\ R_{Dx} \\ R_{Dy} \end{bmatrix} = \begin{bmatrix} 8.117647 \\ 39.088235 \\ 47.294118 \\ -8.117647 \\ -14.088235 \\ -8.117647 \\ 1.911765 \\ -8.117647 \\ 54.911765 \end{bmatrix}$$

### 2 Problem nieliniowy

#### Metoda Newtona-Raphsona

Metoda Newtona Raphsona wypływa z rozwinięcia funkcji wielu zmiennych w szereg Taylora, ucięcia go po członie liniowym i zapostulowania, że nieznan przyrost argumentów ma być taki, aby funkcja miała w tym miejscu wartość



Rysunek 1: Wyjściowy układ mechaniczny (po lewej) oraz poszczególne człony uwolnione od więzów (po prawej).

zero. Zapiszmy takie rozwinięcie dla funkcji  $F(\vec{x})$ , gdzie  $\vec{x} = [x, y]$ , a  $\vec{h} = [h_x, h_y]$ .

$$F(\vec{x}_0 + \vec{h}) = F(\vec{x}_0) + \frac{\partial F}{\partial x} h_x + \frac{\partial F}{\partial y} h_y + \dots$$

W zapisie indeksowym napiszemy dla funkcji  $F_i$  (może tych funkcji być cały wektor dla  $i = 1, \dots, n$ )

$$F_i(\vec{x}_0 + \vec{h}) = F_i(\vec{x}_0) + \frac{\partial F_i}{\partial x_j} h_j + \dots$$

$\frac{\partial F_i}{\partial x_j}$  to nic innego jak macierz Jacobiego. Wiadomo, że jest to macierz kwadratowa, jako że rozwiązujemy zagadnienie mające tyle samo równań co niewiadomych. Przyporównujemy rozwinięcie do zera - pozwoli nam to wyznaczyć takie przesunięcie argumentów, że gdyby liniowe rozwinięcie funkcji wokół danego punktu było słuszne, to w jednej iteracji otrzymywalibyśmy dokładne rozwiązanie zadania. Otrzymujemy:

$$F_i(\vec{x}_0 + \vec{h}) = F_i(\vec{x}_0) + \frac{\partial F_i}{\partial x_j} h_j = 0$$

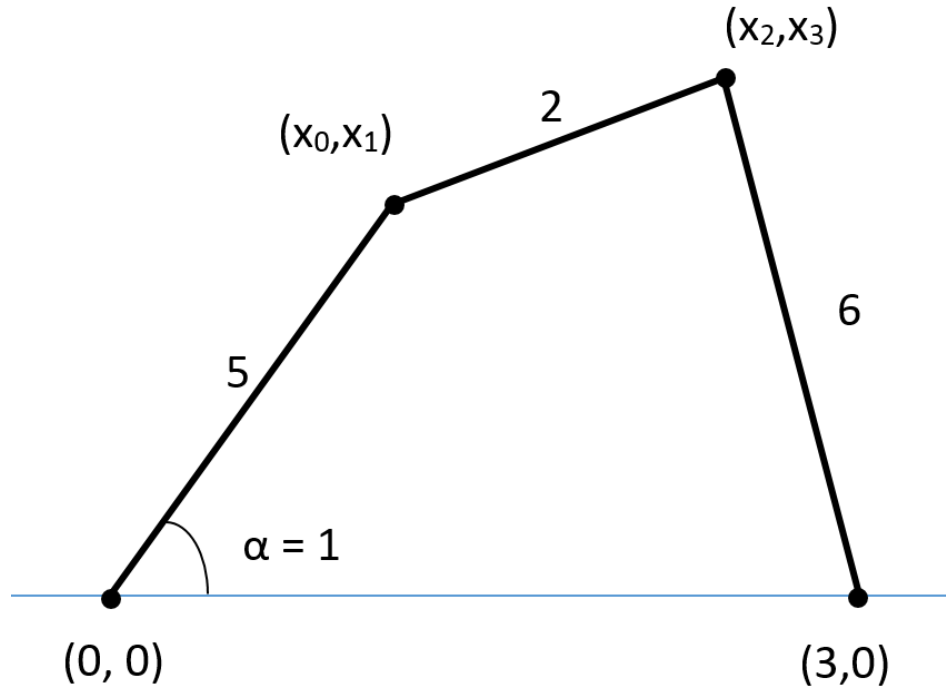
i tym samym

$$\frac{\partial F_i}{\partial x_j} h_j = -F_i(\vec{x}_0)$$

Proces iteracyjny dla metody Newtona-Raphsona ma następującą postać:

1. Wybierz przybliżenie startowe  $x^1$ .
2. Przypisz  $k = 1$ .
3. Wyznacz wektor  $\vec{h}^k$ , rozwiązując układ równań  $\frac{\partial F_i(\vec{x}^k)}{\partial x_j} h_j^k = -F_i(\vec{x}^k)$ .
4. Zaktualizuj przybliżenie rozwiązania:  $\vec{x}^{k+1} = \vec{x}^k + \vec{h}^k$ .
5. Przypisz  $k = k + 1$ .
6. Wróć do punktu 3. i powtarzaj aż do osiągnięcia zbieżności.

## Zadanie 2



Zajmijmy się teraz czworobokiem przegubowym pokazanym powyżej i rozważmy zadanie o położeniach (patrz: TMM I). Zadanie o położeniach zawsze prowadzi do układu równań nieliniowych. Do jego rozwiązania wykorzystamy metodę Newtona-Raphsona. Układ rozważymy we współrzędnych naturalnych (nieznanymi wielkościami będą współrzędne punktów  $(x_0, x_1)$  i  $(x_2, x_3)$ , a równania więzów będą wynikać z odchylenia członu kierującego o kąt  $\alpha$  od poziomu oraz długości dwóch pozostałych członów). Tym samym równania członów są postaci:

$$\begin{aligned} x_0 &= 5 \cdot \cos\alpha \\ x_1 &= 5 \cdot \sin\alpha \\ (x_2 - x_0)^2 + (x_3 - x_1)^2 &= 4 \\ (3 - x_2)^2 + (x_3 - 0)^2 &= 36 \end{aligned}$$

Po rozwinięciu i zapisaniu całego układu w postaci funkcji wektorowej wek-

torowego argumentu otrzymamy następujące sformułowanie naszego układu równań:  $\vec{F}(\vec{x}) = \vec{0}$ , gdzie

$$\vec{F}(\vec{x}) = \begin{bmatrix} x_0 - 5 \cdot \cos\alpha \\ x_1 - 5 \cdot \sin\alpha \\ x_2^2 - 2x_0x_2 + x_0^2 + x_3^2 - 2x_1x_3 + x_1^2 - 4 \\ -6x_2 + x_2^2 + x_3^2 - 27 \end{bmatrix}$$

Wyprowadziwszy powyższe równania możemy analitycznie policzyć macierz Jacobiego:

$$J = \frac{\partial \vec{F}}{\partial \vec{x}} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ -2x_2 + 2x_0 & -2x_3 + 2x_1 & -2x_0 + 2x_2 & -2x_1 + 2x_3 \\ 0 & 0 & -6 + 2x_2 & 2x_3 \end{bmatrix}$$

## Zadania do wykonania

1. Napisz program, który rozwiąże zadanie o położeniach przy wykorzystaniu metody Newtona-Raphsona. W tym celu stwórz następujące funkcje:

- `void Constraints(double *x, double *F);`
- `void JacobiMatrix(double **J, double *x);`
- `void NewtonRaphson(double *x);`

2. Zmodyfikuj program tak, aby nie wymagał analitycznego obliczenia macierzy Jacobiego, ale potrafił numerycznie obliczyć tę macierz. W tym celu stwórz dodatkową funkcję `void JacobiMatrixFD(double **J, double *x);` przybliżającą poprawną macierz Jacobiego macierzą obliczoną z użyciem metody różnic skończonych (ang. *finite difference*). Można tego dokonać z użyciem algorytmu zapisanego w poniższym pseudokodzie (metoda różnic skończonych 2-ego rzędu):

(a) Wybierz małą wartość, np.  $\epsilon = 1e - 8$ , stwórz wektor  $\vec{x}'$  i  $\vec{x}''$ .

(b) Pętla po wszystkich czterech kolumnach:

- Przypisz do  $\vec{x}'$  i  $\vec{x}''$  bieżącą wartość  $\vec{x}$ .
- Zwiększ (zaburz)  $i$ -tą składową  $\vec{x}'$  o  $\epsilon$ , a tę samą składową  $\vec{x}''$  zmniejsz o  $\epsilon$ .
- Wyznacz wektory wartości funkcji  $\vec{F}(\vec{x}')$  oraz  $\vec{F}(\vec{x}'')$ .
- Do  $i$ -tej kolumny macierzy  $J$  wpisz wartości  $\frac{\vec{F}(\vec{x}') - \vec{F}(\vec{x}'')}{2\epsilon}$ .



W ramach testów sprawdź, czy macierz Jacobiego dla punktu startowego obliczona metodą dokładną i numeryczną ma te same wartości. Dla punktu startowego  $\vec{x} = [0, 5, 3, 6]$  macierz Jacobiego ma wartości

$$J = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ -6 & -2 & 6 & 2 \\ 0 & 0 & 0 & 12 \end{bmatrix}$$