

Metody Komputerowe Metoda Elementów Skończonych

Element trójwymiarowy liniowy : pręt 3D - część 2

Element prętowy 3D można wykorzystać do obliczeń symulujących zachowanie się konstrukcji przestrzennych pod obciążeniem, które składają się z połączonych przegubowo elementów pracujących wyłącznie na ściskanie lub rozciąganie.

Przykład nr 1.

Wyznaczyć przemieszczenia węzłów oraz maksymalne naprężenie występujące w elementach dla kopuły o kształcie stożka (średnica $2L=20\text{m}$, wysokość $H=10\text{m}$) skonstruowanej z prętów wykonanych z materiału o module $E = 200\text{GPa}$, ciężarze objętościowym $\gamma = 78.5\text{kN/m}^3$ i stałych polach przekroju wynoszących $A = 0.003\text{m}^2$ w przypadku obciążenia ciężarem własnym.

Rozwiązanie:

UWAGA: ze względu na wariantowe obliczenia, w których znaczna część komend będzie się powtarzać, zaleca się umieszczanie poszczególnych poleceń w skrypcie systemu Matlab.

Krok 1 – dyskretyzacja zadania

Na początek, wprowadzimy dwie zmienne globalne (promień podstawy kopuły L i jej wysokość H), które będą widziane przez wszystkie polecenia Matlab (będą potrzebne w realizacji zadań do samodzielnego rozwiązania):

```
global H L
```

i nadamy im wartości wymiarów naszej kopuły:

```
H = 10  
L = 10
```

Należy wygenerować założoną liczbę punktów na powierzchni stożka, które będą węzłami (czyli „końcami”) elementów prętowych 3D. Generację punktów przeprowadza się dwuetapowo: najpierw na płaszczyźnie XY , a potem w przestrzeni XYZ nadając punktom obliczone dla przyjętego kształtu wysokości Z .

Etap pierwszy. Konstrukcja będzie oparta na podłożu w 36 punktach, wygenerowanych na płaszczyźnie XY na okręgu o promieniu $L=10\text{m}$ co 10 stopni kątowych (kąt od 0 stopni do 350 stopni, co 10 stopni):

```
alfa = 0:10:350
```

należy zamienić stopnie na radiany

```
alfa = alfa .* pi ./ 180
```

a potem wyliczyć współrzędne x i y punktów leżących na okręgu o promieniu L

```
x1 = L .* cos(alfa)
```

```
y1 = L .* sin(alfa)
```

Do dalszych obliczeń przydadzą się następujące informacje: ile punktów wygenerowaliśmy i jakie są ich odległości od środka układu współrzędnych ($X=0$, $Y=0$, $Z=0$) – odległości te powinny być zapisane w formie wektora (dla każdego punktu przypisana jest jego odległość):

```
n1 = max(size(alfa))  
L1 = ones(1,n1) .* L
```

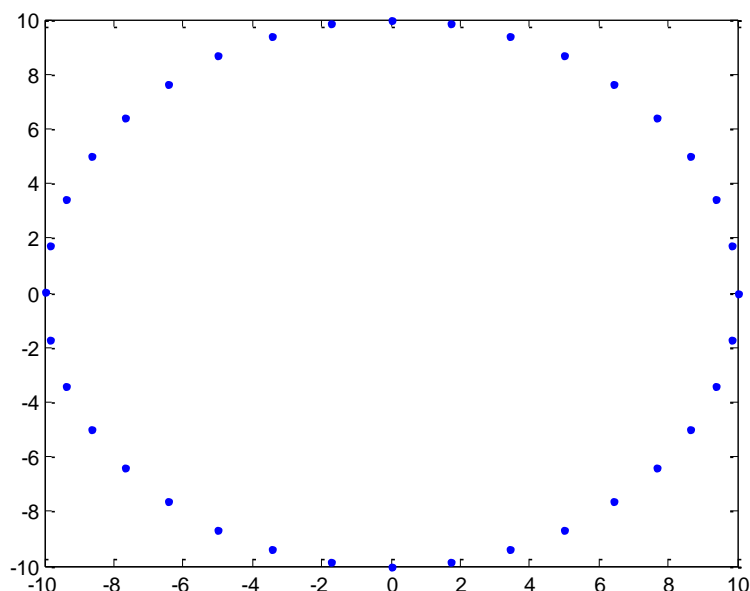
Poprawność generacji punktów można zweryfikować wyświetlając ich położenie na płaszczyźnie XY poleceniem `plot(x1,y1,'.')` (`hold on` - pozwoli na późniejsze dorysowywanie elementów na rysunku, a `pause` - umożliwi zatrzymanie obliczeń w celu obejrzenia rysunku i wznowienie ich po naciśnięciu dowolnego klawisza):

```
plot(x1,y1,'.')
```

```
hold on
```

```
pause
```

Powinien pojawić się następujący rysunek (nie zamykamy rysunku po obejrzeniu – przyda się do kolejnych weryfikacji):



Wygenerujemy teraz kolejny zestaw punktów na płaszczyźnie XY, który będzie leżał na okręgu o nieco mniejszym promieniu, np. $0.85L$ (będziemy stopniowo pokrywać powierzchnię stożka punktami leżącymi na coraz mniejszych okręgach). Niech punkty będą rozmieszczone co 10 stopni katowych ale w przesunięciu o 5 stopni w stosunku do punktów leżących na zewnętrznym okręgu (tzn. należy wygenerować kąty od 5 stopni do 355 stopni, co 10 stopni):

```
alfa = 5:10:355
```

jak poprzednio, uzyskane wartości należy zamienić na radiany

```
alfa = alfa .* pi ./ 180
```

a potem wyliczyć współrzędne x i y punktów leżących na okręgu o promieniu $0.85L$

```
x2 = 0.85 .* L .* cos(alfa)  
y2 = 0.85 .* L .* sin(alfa)
```

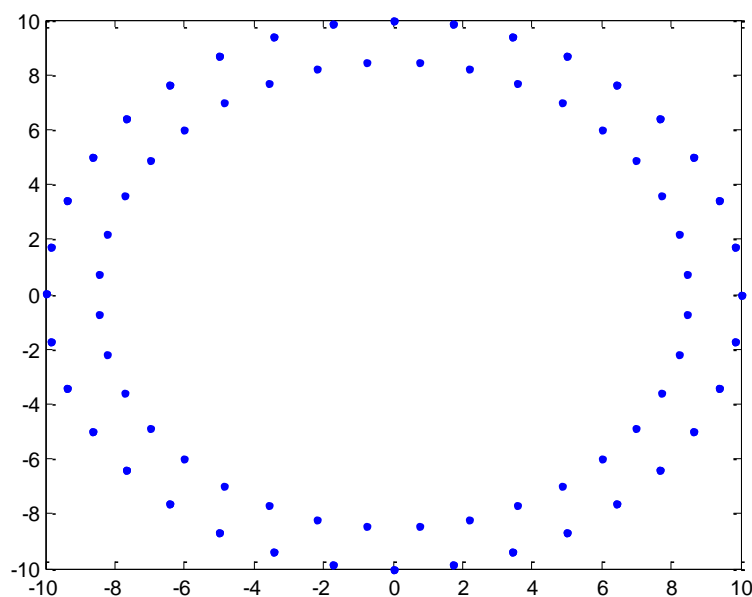
Do dalszych obliczeń znów przydadzą się następujące informacje: ile punktów wygenerowaliśmy i jakie są ich odległości od środka układu współrzędnych ($X=0$, $Y=0$, $Z=0$) – odległości te powinny być zapisane w formie wektora (dla każdego punktu przypisana jest jego odległość):

```
n2 = max(size(alfa))
L2 = ones(1,n2) .* 0.85 .* L
```

Poprawność generacji punktów można zweryfikować wyświetlając ich położenie na płaszczyźnie XY poleceniami:

```
plot(x2,y2,'.')
hold on
pause
```

Na stworzonym wcześniej rysunku powinny pojawić się nowe punkty:



Wygenerujemy kolejny zestaw punktów na płaszczyźnie XY, który będzie leżał na okręgu o znów nieco mniejszym promieniu, np. $0.7L$. Niech punkty będą rozmieszczone co 10 stopni kątowych tak jak punkty leżące na zewnętrznym, największym okręgu (tzn. od 0 stopni do 350 stopni, co 10 stopni):

```
alfa = 0:10:350
```

jak poprzednio, zamieniamy wartości na radiany

```
alfa = alfa .* pi ./ 180
```

I wyliczamy współrzędne x i y

```
x3 = 0.7 .* L .* cos(alfa)
y3 = 0.7 .* L .* sin(alfa)
```

Jak poprzednio, przygotowujemy następujące informacje: ile punktów wygenerowaliśmy i jakie są ich odległości od środka układu współrzędnych ($X=0$, $Y=0$, $Z=0$):

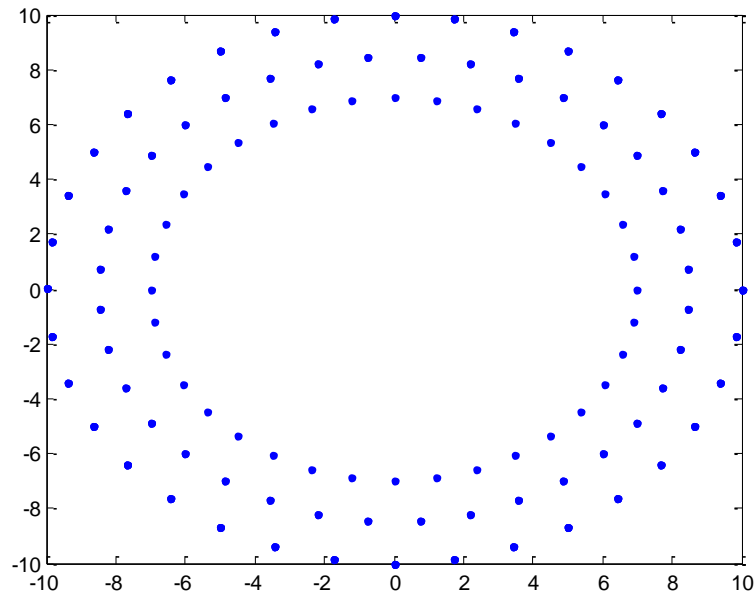
```
n3 = max(size(alfa))
```

```
L3 = ones(1,n3) .* 0.7 .* L
```

Poprawność generacji punktów zweryfikujemy wyświetlając ich położenie na płaszczyźnie XY poleceniem:

```
plot(x3,y3,'.')  
hold on  
pause
```

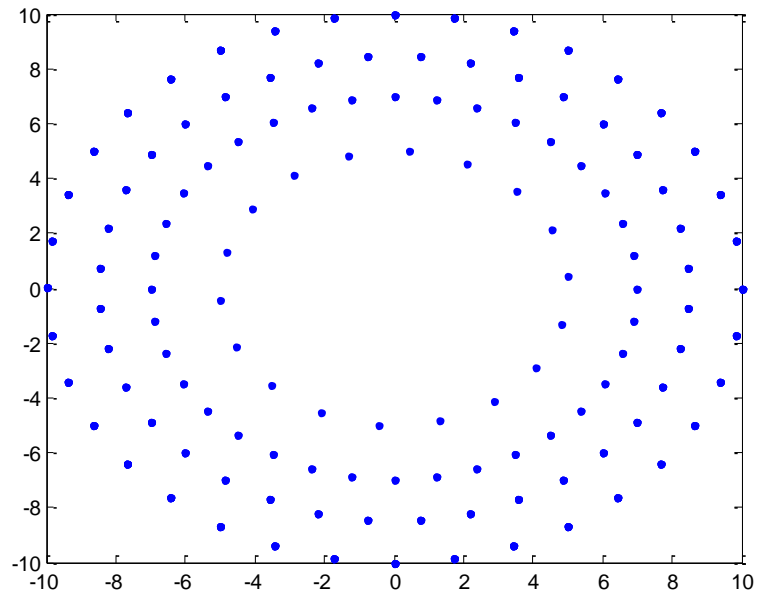
i powinny pojawić się dodatkowe punkty:



Operacje generacji punktów na coraz mniejszych okręgach powtarzamy wg wcześniejszego szablonu, przyjmując:

- okrąg o promieniu $0.5L$ oraz kątowny odstęp między punktami: 20 stopni (zbliżamy się do środka, robi się coraz mniej miejsca) z przesunięciem o 5 stopni względem poprzedniego zestawu punktów, tzn.:

```
alfa = 5:20:350  
alfa = alfa .* pi ./ 180  
x4 = 0.5 .* L .* cos(alfa)  
y4 = 0.5 .* L .* sin(alfa)  
n4 = max(size(alfa))  
L4 = ones(1,n4) .* 0.5 .* L  
plot(x4,y4,'.')  
hold on  
pause
```

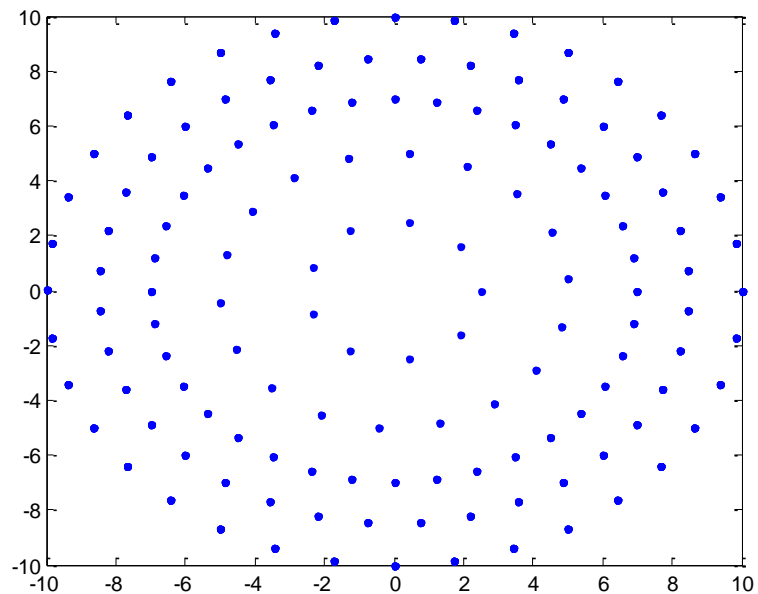


- okrąg o promieniu $0.25L$ oraz kątowy odstęp między punktami: 40 stopni (zbliżamy się do środka, robi się coraz mniej miejsca), tzn.:

```

alfa = 0:40:350
alfa = alfa .* pi ./ 180
x5 = 0.25 .* L .* cos(alfa)
y5 = 0.25 .* L .* sin(alfa)
n5 = max(size(alfa))
L5 = ones(1,n5) .* 0.25 .* L
plot(x5,y5,'.')
hold on
pause

```



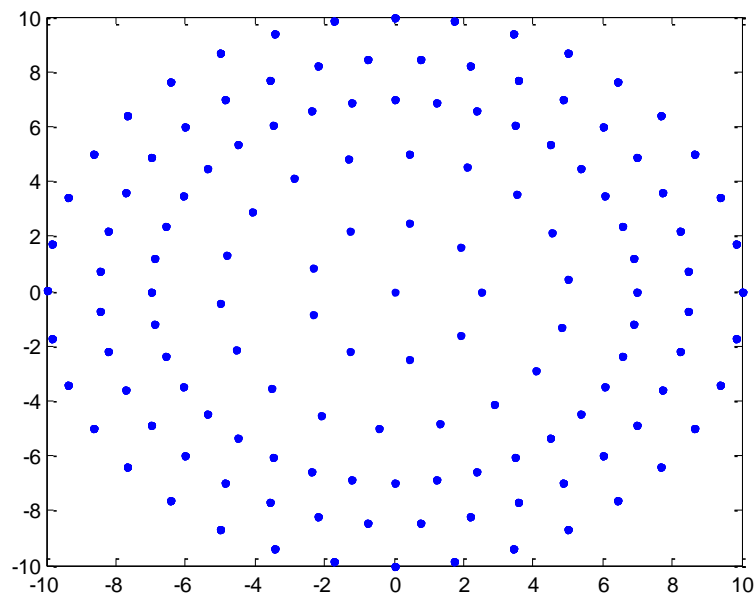
- okrąg o promieniu 0, tzn. punkt w środku wszystkich okręgów (wierzchołek stożka):

```

x6 = 0
y6 = 0
n6 = 1

```

```
L6 = 0
plot(x6,y6,'.')
hold on
pause
```



Po obejrzeniu rysunku można go zamknąć. Teraz zbierzemy wszystkie zestawy punktów w ujednolicone wektory wierszowe: najpierw zbierzemy współrzędne x i y:

```
x = [x1 x2 x3 x4 x5 x6]
y = [y1 y2 y3 y4 y5 y6]
```

potem zbierzemy w jeden wektor wszystkie odległości wygenerowanych punktów od środka układu współrzędnych (czyli środka rzutu kopuły na płaszczyznę XY):

```
r = [L1 L2 L3 L4 L5 L6]
```

i zsumujemy liczby punktów:

```
n = n1 + n2 + n3 + n4 + n5 + n6
```

Teraz „podniesiemy” punkty z płaszczyzny XY nadając im współrzędne Z, które obliczymy z równania liniowego opisującego tworzącą stożka. Najpierw wygenerujemy wektor z wysokością naszej kopuły h ($H=10\text{m}$, wektor h umożliwi obliczenie wszystkich współrzędnych Z za pomocą jednego polecenia):

```
h = ones(1,n) .* H
```

a potem zastosujemy równanie liniowe $z = h - r$, w którym h jest wysokością kopuły a r jest odległością punktu od jej środka (proszę zauważyć, że dla $r = 0 \rightarrow z=10$ – wierzchołek stożka, a dla $r=10 \rightarrow z = 0$ – zewnętrzny obrys podstawy stożka):

```
z = h - r
```

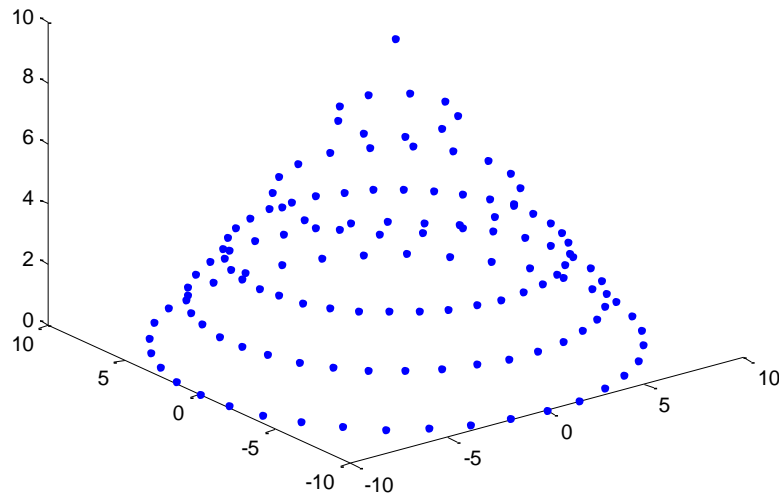
Układ punktów w przestrzeni 3D można obejrzeć dzięki poleceniom (`close(gcf)` zamyka automatycznie poprzedni rysunek):

```
close(gcf)
plot3(x,y,z,'.')

```

pause

powinien pojawić się następujący rysunek:



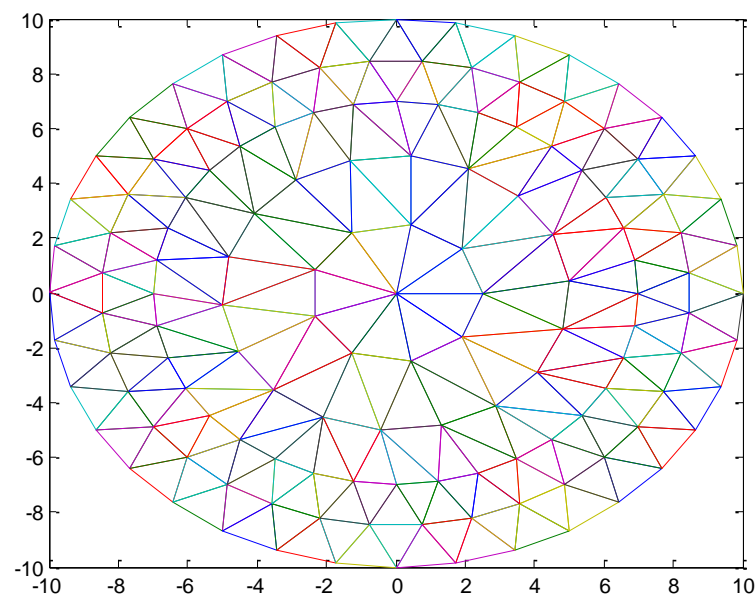
Po zamknięciu rysunku przystąpimy do generacji elementów - wykorzystamy w tym celu technikę triangulacji Delaunay'a, która udostępniona jest w systemie Matlab w poleceniu:

```
triangulacja = delaunay(x,y)
```

Efekty triangulacji można obejrzeć wywołując polecenia:

```
close(gcf)  
trimesh(triangulacja,x,y)  
pause
```

i powinien pojawić się rysunek:



Wynikiem triangulacji jest macierz z numerami węzłów przypisanych do trójkątnej siatki elementów - w oknie komend Matlab'a powinniśmy zobaczyć macierz:

```
triangulacja =
```

136	132	133
92	55	56
44	45	81
111	112	128
107	108	126
108	72	73
58	94	57
108	71	72
31	66	30
96	120	95
96	97	121
58	95	94
. . .		
124	104	105
103	67	104
31	32	67
67	32	68
35	70	34
33	69	68
34	69	33

w wierszu umieszczone są trzy numery węzłów tworzących trójkąt w siatce, a wierszy jest tyle ile wygenerowanych zostało trójkątów. My potrzebujemy odrębnych definicji poszczególnych elementów (czyli poszczególnych boków trójkąta, a nie całych trójkątów) i dlatego musimy „pociąć” te trójkąty na poszczególne boki, a następnie usunąć powstałe duplikaty. Służy do tego specjalnie przygotowane polecenie:

```
elementy = ZamienTriangulacjeNaElementy(triangulacja,n)
```

które nie tylko wyodrębni numery węzłów definiujących poszczególne elementy, ale jeszcze posortuje te numery od mniejszych do większych:

```
elementy =
    1    37
    1     2
    2    37
    2    38
    2     3
    3     4
    . . .
   128   136
   131   136
   132   136
   134   136
   135   136
```

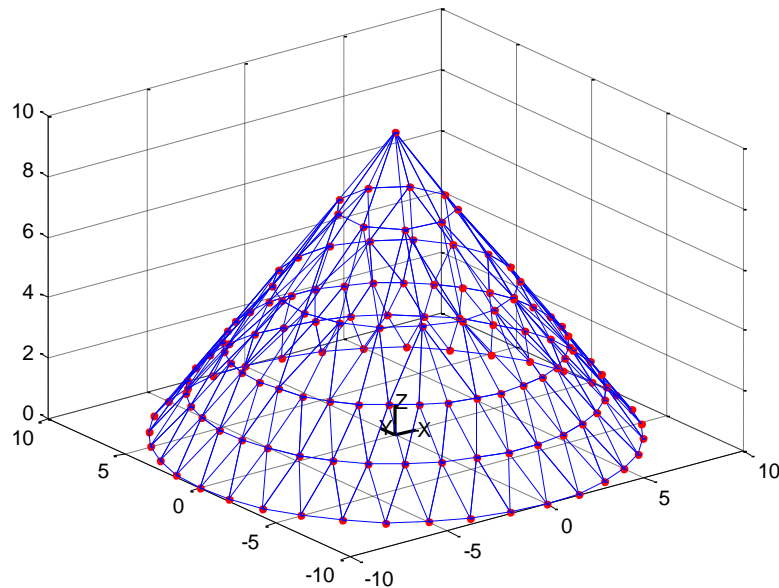
Przyda nam się informacja o liczbie finalnie wygenerowanych elementów:

```
ne = max(size(elementy))
```

Teraz możemy zobaczyć naszą siatkę elementów w przestrzeni 3D wykonując specjalnie przygotowane w tym celu polecenie:

```
close(gcf)
showmesh3D(elementy,x,y,z,'r','b')
pause
```


Funkcja `showmesh3D` rysuje węzły i siatkę osobnymi kolorami, dlatego po argumentach: `elementy, x, y, z` trzeba podać jeszcze symbole kolorów: dla węzłów 'r' (red) – czerwony i elementów 'b' (blue) – niebieski.



Krok 2 i 3 – utworzenie macierzy sztywności dla każdego elementu oraz składanie macierzy sztywności elementów w jedną globalną macierz dla całego układu

W odróżnieniu od poprzednio realizowanych zadań, kroki 2 i 3 zostaną przeprowadzone jednocześnie ze względu na znaczną liczbę elementów w układzie. Wprowadzamy dane materiałowe i geometryczne naszego zadania: moduł Younga E i pole przekroju poprzecznego elementów A :

```
E=200e6
A=0.003
```

Pozostałe stałe geometryczne (L_e , Th_x , Th_y i Th_z) wyznaczamy ze współrzędnych węzłów wykorzystując pętlę iteracyjną `for-end` ze względu na znaczną liczbę elementów:

```
for ie=1:ne
    w1 = elementy(ie,1)
    w2 = elementy(ie,2)
    [Le(ie), Thx(ie), Thy(ie), Thz(ie)] =
    DlugoscKatElementPretowy3D(x(w1),y(w1),z(w1),x(w2),y(w2),z(w2))
end
```

powyższy zapis oznacza:

dla pewnej zmiennej ie , która przyjmuje wartości od 1 do ne (całkowita liczba elementów) wykonaj:

przepisz numer pierwszego węzła elementu nr ie do zmiennej $w1$

przepisz numer drugiego węzła elementu nr ie do zmiennej $w2$

oblicz dla elementu nr ie : L_e , Th_x , Th_y i Th_z funkcją `DlugoscKatElementPretowy3D`

przyjmując współrzędne x , y , z pierwszego węzła $w1$ oraz współrzędne x , y i z węzła drugiego $w2$

koniec pętli

W ten sposób kilkoma linijkami komend wygenerujemy długości L_e oraz kąty θ_{Th} dla kilkuset elementów znajdujących się w układzie. Całkowita liczba stopni swobody w układzie N jest równa trzykrotności liczby węzłów n (każdy węzeł ma 3 stopnie swobody):

```
N = n .* 3
```

zatem globalna macierz sztywności powinna mieć wymiary $N \times N$ (na końcu stawiamy średnik, bo to duża macierz i jej wyświetlenie na ekranie zajmie dużo czasu):

```
K = zeros(N,N);
```

Tak jak podczas generacji cech geometrycznych prętów (L_e , θ_{Th}), macierze poszczególnych elementów oraz macierz globalną poskładamy w sposób automatyczny stosując pętlę **for-end** (na końcu stawiamy średniki, bo to duże macierze i ich wyświetlanie w pętli na ekranie zajmie bardzo dużo czasu):

```
for ie=1:ne
    k = SztynoscElementPretowy3D(E,A,Le(ie),Thx(ie),Thy(ie),Thz(ie));
    K = ZlozSztynoscPretow3D(K,k,elementy(ie,1),elementy(ie,2));
end
```

co oznacza:

dla pewnej zmiennej ie , która przyjmuje wartości od 1 do ne (liczba elementów)

wykonaj:

wyznacz macierz elementu k funkcją `SztynoscElementPretowy3D` (dla elementu nr ie)

złóż macierz globalną K z macierzy k funkcją `ZlozSztynoscPretow3D` (dla elementu nr ie)

koniec pętli

Uzyskamy w ten sposób kompletną macierz globalną układu.

Krok 4 – uwzględnienie warunków brzegowych

Ze względu na rozmiary układu równań $[K]\{u\}=\{f\}$ nie można go rozpisać w niniejszym skrypcie. Biorąc jednak pod uwagę fakt, iż analizowana konstrukcja oparta jest na podporach zewnętrzną krawędzią, która została wygenerowana pierwszymi n_1 punktami, można wykreślić z macierzy globalnej K pierwsze $n_1 \times 3$ wierszy oraz pierwsze $n_1 \times 3$ kolumn (trzy stopnie swobody w każdym punkcie!). Zatem, do dalszych obliczeń należy skopiować do macierzy k resztę wierszy i kolumn z K poczynając od wiersza i kolumny o numerze $p=n_1 \times 3+1$ a kończąc na wierszu i kolumnie o numerze N :

```
p = n1*3+1
k = K(p:N,p:N)
```

Z kolei ciężar własny każdego pręta wygenerujemy poleceniem (ciężar objętościowy * pole * długość; Matlab wiedząc, że zmienna L_e jest wektorem przeprowadzi obliczenia wektorowo i wygeneruje w wyniku mnożeń wektor ciężarów Q_e):

```
Qe = 78.5 .* A .* Le;
```

Globalny wektor obciążeń węzłowych ma długość równą liczbie stopni swobody w całym układzie:

```
F = zeros(N,1);
```

Obciążenie węzłowe na końcu każdego z prętów jest równe połowie ciężaru prętów połączonych w danym węźle:

```

for ie=1:ne
    w1 = elementy(ie,1);
    w2 = elementy(ie,2);
    F(w1.*3) = F(w1.*3) - Qe(ie) ./ 2;
    F(w2.*3) = F(w2.*3) - Qe(ie) ./ 2;
end

```

powyższy zapis oznacza:

dla pewnej zmiennej ie , która przyjmuje wartości od 1 do ne (całkowita liczba elementów) **wykonaj:**

przepisz numer pierwszego węzła elementu nr ie do zmiennej $w1$

przepisz numer drugiego węzła elementu nr ie do zmiennej $w2$

*dodaj do wiersza o numerze $3*w1$ globalnego wektora obciążeń F połowę ciężaru elementu ie*

*dodaj do wiersza o numerze $3*w2$ globalnego wektora obciążeń F połowę ciężaru elementu ie*

koniec pętli

UWAGA! Kierunek działania pola grawitacyjnego jest przeciwny do zwrotu osi Y, dlatego ciężary uwzględniamy ze znakiem ujemnym.

Po przygotowaniu globalnego wektora obciążeń F , kopiujemy z niego tylko ten fragment, który odpowiada nieznanym przemieszczeniom w układzie (tzn. od wiersza p do wiersza N):

```
f = F(p:N);
```

i komplet danych mamy już przygotowany.

Krok 5 – rozwiązanie równań

Skonstruowany układ równań rozwiążemy poleceniem:

```
u=k\f
```

wyznaczając nieznane przemieszczenia u .

Krok 6 – obróbka wyników (postprocessing)

Mając przemieszczenia wszystkich węzłów, możemy obliczyć reakcje w podporach. Najpierw zbierzmy przemieszczenia w jeden wektor (dodajemy do wyników zerowe przemieszczenia podpór w pierwszych $n1$ punktach):

```

u0 = zeros(n1.*3,1);
U = [u0; u];

```

a potem wyliczmy komplet sił w układzie:

```
F=K*U
```

Obejrzymy teraz wyniki obliczeń rysując oryginalną siatkę elementów (kolorem szarym wygenerowanym intensywnościami RGB [Red Blue Green] - [0.5 0.5 0.5])

```

close(gcf)
showmesh3D(elementy,x,y,z,[0.5 0.5 0.5],[0.5 0.5 0.5]);

```

oraz siatkę zdeformowaną przemieszczeniami węzłów (na czerwono) stosując mnożnik do przemieszczeń (1000), aby deformacje były zauważalne:

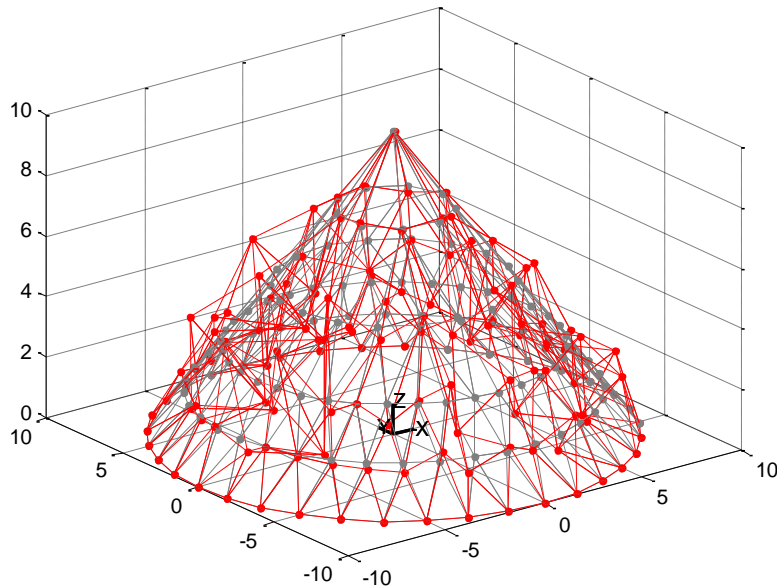
```
mnoznik = 1000;
```

```

X = x + U(1:3:N)' .* mnoznik;
Y = y + U(2:3:N)' .* mnoznik;
Z = z + U(3:3:N)' .* mnoznik;
showmesh3D(elementy,X,Y,Z,'r','r');

```

Powinien pojawić się rysunek:



Maksymalne przemieszczenie w układzie wyznaczymy poleceniem:

```
max(abs(U))
```

a maksymalną wartość siły poleceniem:

```
max(abs(F))
```

Zadania do samodzielnego rozwiązania:

Zadanie nr 1.

Powtórzyć obliczenia z przykładu dla kopuły w kształcie półkuli o równaniu:

```
z = sqrt(h.^2 - r.^2)
```

Zadanie nr 2.

Powtórzyć obliczenia z przykładu dla kopuły w kształcie paraboloidy o równaniu:

```
z = h - 1./L .* r.^2
```

Zadanie nr 3.

Powtórzyć obliczenia z przykładu dla kopuły w kształcie katenoidy o równaniu:

```
z = katena(r, 6.18759227774256)
```

W którym zadaniu maksymalne przemieszczenia są najmniejsze? Uzasadnić odpowiedź.