# Rozwiązywanie układów równań nieliniowych

Opracował *dr inż. Robert JAKUBOWSKI, Politechnika Rzeszowska, KSiSL*

**Table of Contents**



Wykład poświęcony jest wykorzystaniu narzędzi Matlab_a do rozwiązywania układów równań nieliniowych.

Zaprezentuję dwa podejścia do rozwiązania tego typu zadań. Pierwsze będzie polegało na rozwiązaniu z wykorzystaniem biblioteki do obliczeń symbolicznych (symbolic toolbox) i drugie wykorzystujące metody numeryczne w rozwiązywaniu układów równań.

## Rozwiązywanie układówn równań z wykorzystaniem symbolic toolbox

Do prowadzenie obliczeń symbolicznych wymagany jest symbplic toolbox. Zmienne symboliczne muszą być zdefiniowane poprzez komendę ***syms***. Samo rozwiązywanie jest realizowane przez funkcję ***solve***. Funkcję solve można wykorzystać do rozwiązywania symbolicznego równań nieliniowych jednej zmiennej jak też układów rónań nieliniowych.

Rozpatrzmy układ równań nieliniowych dwóch zmiennych x i y przedstawiony poniżej (w równaniu stosuje się podwójny znak ==):

x^2 + x*y + y == 3

x^2 - 4*x + 3 == 0

Rozwiązanie uzyskamy wykonując poniższe polecenia.

```
syms x y
[S] = solve(x^2 + x*y + y == 3,x^2 - 4*x + 3 == 0)
```

```
S = struct with fields:
    x: [2×1 sym]
    y: [2×1 sym]
```

Otrzymujemy rozwiązanie w postaci struktury S.x i S.y, Aby zobaczyć wyniki należy wywołać strukturę w postaci S.x, S.y

```
S.x
```

```
ans =
```

$$\begin{pmatrix} 1 \\ 3 \end{pmatrix}$$

```
S.y
```

```
ans =
```

$$\begin{pmatrix} 1 \\ -\frac{3}{2} \end{pmatrix}$$

Widzilmy, że rozwiązanie układu równań stanowi dwa punkty: pierwszy P1=(1,1) i drugi P2=(3,-3/2)

Podobne rozwiązanie można otrzymać podstawiając taką ilość zmiennych wyjściowych, jaka jest ilość zmienneych wejściowych wtedy: rozwiązanie zobaczymy od razu, gdy linia komendy nie jest zakończona średnikiem

```
[Sx,Sy] = solve(x^2 + x*y + y == 3,x^2 - 4*x + 3 == 0)
```

```
Sx =
```

$$\begin{pmatrix} 1 \\ 3 \end{pmatrix}$$

```
Sy =
```

$$\begin{pmatrix} 1 \\ -\frac{3}{2} \end{pmatrix}$$

Rozwiązania symboliczne pozwalają na rowiązanie układów równań z parametrami, które nie są liczbami. Załżmy, że przedstawiony układ równań nieco zmodyfikujemy dodając parametr a i b i obecnie będziemy mieli układ w postaci:

x^2 + x*y + y*b == 3*a

x^2 - 4*x + 3 *b== 0

Ponieważ zostały dołożone dwie zmienne symboliczne, dlatego musimy je dodefiniować:

```
syms a b
```

Teraz użyjemy funkcji solve, podając na początku równania, a następnie zmienne według których będzie rozwiązywany układ równań:

```
[S] = solve(x^2 + x*y + y*b == 3*a, x^2 - 4*x + 3*b == 0, x, y)
```

```
S = struct with fields:
    x: [2×1 sym]
    y: [2×1 sym]
```

Zobaczmy wyniki wywołując:

```
S.x
```

```
ans =
```

$$\begin{pmatrix} \frac{12\,a+3\,a\,b+3\,b^2-\frac{b^2\,\sigma_1}{\sigma_3}-\frac{7\,b\,\sigma_1}{\sigma_3}}{3\,a+7\,b} \\ \frac{12\,a+3\,a\,b+3\,b^2-\frac{b^2\,\sigma_2}{\sigma_3}-\frac{7\,b\,\sigma_2}{\sigma_3}}{3\,a+7\,b} \end{pmatrix}$$

where

$$\sigma_1 = 6\,a-14\,b+3\,a\,b-3\,a\,\sqrt{4-3\,b}-7\,b\,\sqrt{4-3\,b}+3\,b^2$$

$$\sigma_2 = 6\,a-14\,b+3\,a\,b+3\,a\,\sqrt{4-3\,b}+7\,b\,\sqrt{4-3\,b}+3\,b^2$$

$$\sigma_3 = b^2+7\,b$$

```
S.y
```

```
ans =
```

$$\begin{pmatrix} \frac{6\,a-14\,b+3\,a\,b-\sigma_2-\sigma_1+3\,b^2}{b^2+7\,b} \\ \frac{6\,a-14\,b+3\,a\,b+\sigma_2+\sigma_1+3\,b^2}{b^2+7\,b} \end{pmatrix}$$

where

$$\sigma_1 = 7\,b\,\sqrt{4-3\,b}$$

$$\sigma_2 = 3\,a\,\sqrt{4-3\,b}$$

Tym razem rozwiązanie jest podane w postaci zależności funkcyjnej o złożonej strukturze.

Sprawdzmy wynik rozwiązania, gdy układ rozwiążemy względem x i a zamiast x i y

```
[S] = solve(x^2 + x*y + y*b == 3*a, x^2 - 4*x + 3*b == 0, x, a)
```

```
S = struct with fields:
    x: [2×1 sym]
    a: [2×1 sym]
```

Tym razem zmienna S ma dwie podzmienne x i a, stąd aby zobaczyć ich wartości wywołamy:

```
S.x
```

```
ans =
```

$$\begin{pmatrix} \frac{2y-\frac{3y\sqrt{\frac{16}{9}-\frac{4b}{3}}}{2}-6\sqrt{\frac{16}{9}-\frac{4b}{3}}+8}{y+4} \\ \frac{2y+\frac{3y\sqrt{\frac{16}{9}-\frac{4b}{3}}}{2}+6\sqrt{\frac{16}{9}-\frac{4b}{3}}+8}{y+4} \end{pmatrix}$$

```
S.a
```

```
ans =
```

$$\begin{pmatrix} \frac{2y}{3}-b+\frac{by}{3}-\sigma_1-\sigma_2+\frac{8}{3} \\ \frac{2y}{3}-b+\frac{by}{3}+\sigma_1+\sigma_2+\frac{8}{3} \end{pmatrix}$$

where

$$\sigma_1=\frac{y\sqrt{\frac{16}{9}-\frac{4b}{3}}}{2}$$

$$\sigma_2=2\sqrt{\frac{16}{9}-\frac{4b}{3}}$$

## Rozwiązywanie symboliczne z wykorzystaniem funkcji vpasolve

W przypadku, gdy rozwiązanie w zapisie symbolicznym (w postaci zleżności funkcjyjnej) nie jest możliwe funkcja ***solve*** nie może mieć zastosowania.

Rozwiążmy przedstawiony poniżej układ:

2xy-2y^2=1

y^3-cos(x)=-0.5

Spróbujmy rozwiązać to przez funkcję ***solve***.

```
syms x y
S=solve(2*x.*y-2*y.^2==1, y.^3-cos(x)==-0.5, x, y)
```

```
Warning: Unable to solve symbolically. Returning a numeric solution using vpasolve.
S = struct with fields:
    x: [1×1 sym]
    y: [1×1 sym]
```

Otrzymaliśmy informację, że zadanie nie zostało rozwiązane funkcją solve, a w zamian użyto funkcję ***vpasolve***. Rozwiązanie jest dostepne przez wywołanie struktury S.x S.y

```
S.x
```

```
ans = 80.634211717511172572960256063913
```

```
S.y
```

```
ans = 0.006201318841029554502755213620912 2
```

Wywołajmy rozwiązanie układu używajac funkcji numerycznej ***vpasolve***

```
S=vpasolve(2*x.*y-2*y.^2==1, y.^3-cos(x)==-0.5, x, y)
```

```
S = struct with fields:
    x: [1×1 sym]
    y: [1×1 sym]
```

```
S.x
```

```
ans = 80.634211717511172572960256063913
```

```
S.y
```

```
ans = 0.006201318841029554502755213620912 2
```

Otrzymane rozwiązanie pokazuje, że dla x=80.6342.. i y=0.0062.. występują warunki spełniające róność obydwu funkcji. Chcąc zobaczyć, czy dla innych warunków początkowych, można uzyskać inne rozwiązanie, można wskazć obszar poszukiwania rozwiązania wywołując funkcję w postaci podanej poniżej, gdzie wartości wpisane w nawiasie kwadratowym definiują punkt startowy poszukiwania rozwiązania dla x=-10 dla y=0:

```
S=vpasolve(2*x.*y-2*y.^2==1, y.^3-cos(x)==-0.5, x, y,[-10, 0]);
S.x
```

```
ans = −7.330759718379841787158019057681 9
```

```
S.y
```

```
ans = −0.068852435110532017315385178883978
```

Tym razem otrzymaliśmy rozwiązanie dla innych wartości x i y, bo zdefiniowliśmy rozpoczęcie poszukiwania rozwiazania x=-10 y=0. Możemy również wykorzystać przedziały do poszukiwania rozwiązania. Przyjżyjmy się jaki będze wynik rozwiązania, gdy przedział x będzie od -5 do 1, a y od -1 do 1

```
S=vpasolve(2*x.*y-2*y.^2==1, y.^3-cos(x)==-0.5, x, y,[-5, 1; -1, 1]);
S.x
```

```
ans = −1.414271110189339840727708167724 9
```

```
S.y
```

```
ans = −0.700756426702251535627748756847 78
```

Teraz znaleźliśmy kolejne rozwiązanie układu równań.

Żeby zobrazować wyniki rozwiązania najlepiej jest narysować obydwie funkcje w interesującym nas przedziale poszukiwania rozwiązania. Wtedy na podstawie analizy wykresu można wskazać interesujące nas obszary

poszukiwania dokładnego rozwiązania. Do tej pory otrzymaliśmy wyniki spełniające układ równań dla x od ok. -7 do ponad 80, y  było w pobliżu 0, dlatego narysujmy wykres obydwu funkcji w przedzaile dla x =-10 do 85 y=-1 do 1. Aby było to łatwiej zrobić zdefiniujmy równania r1 i r2, a do rysowania układu równań dwóch zmiennych wykorzystamy funkcję ***fimplicit***:

```
% Pierwsze równanie przypisujemy zmiennej r1, zwróćmy uwagę na podwójny
% znak == po stronie samego równania
r1=2*x.*y-2*y.^2==1
```

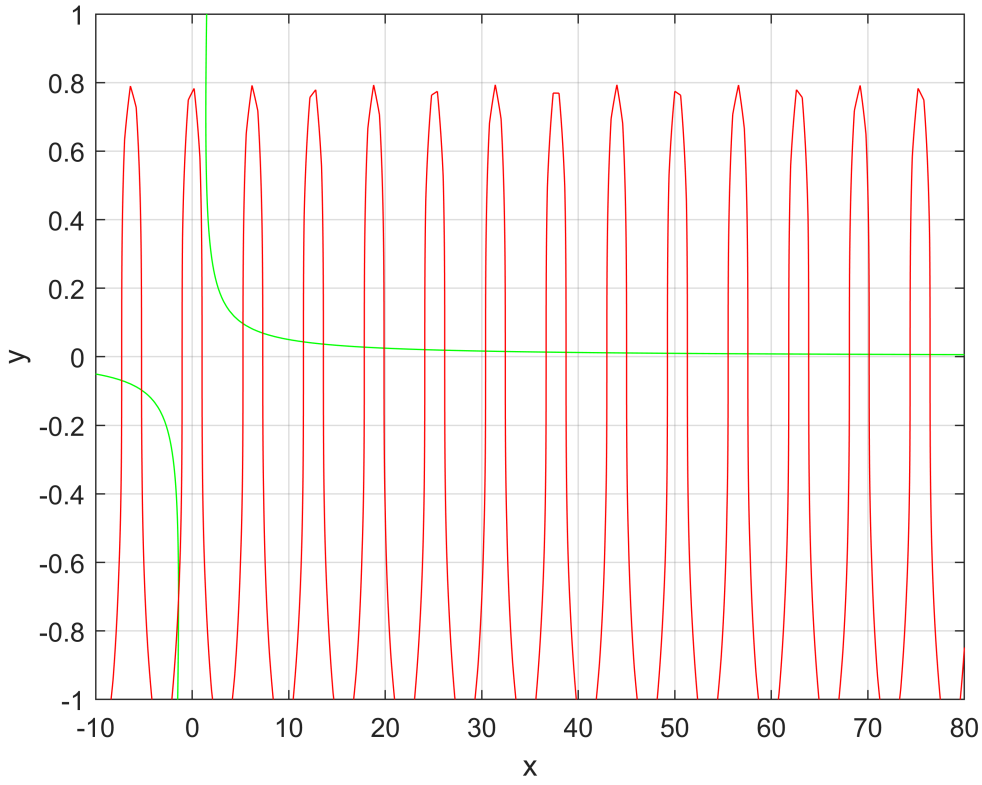
r1 = $2\,x\,y - 2\,y^2 = 1$

```
% drugie rónanie przypisujemy zmiennej r2
r2=y.^3-cos(x)==-0.5
```

r2 =

$$y^3 - \cos(x) = -\frac{1}{2}$$

```
%Rysujemy pierszą funkcję w kolorze zielonym 'g'
fimplicit(r1,[-10,80,-1,1],'g')
hold on
%Rysujemy drugą funkcję w kolorze czerwonym 'r'
fimplicit(r2,[-10,80,-1,1],'r')
%Dodajemy opis osi
ylabel('y')
xlabel('x')
grid on
hold off
```




Widocznym jest, że dla x w przedziale -10 do 80 występuje wiele rozwiązań spełniających analizowany układ równań. Ograniczmy się do znalezienia wszystkich rozwiązań w przedziale od -10 do 0 aby zdefiniować punkty początkowe przerysujmy układ tak, aby powiększyć ten fragment:

```
fimplicit(r1,[-10,0,-1.2,1],'g')
hold on
fimplicit(r2,[-10,0,-1.2,1],'r')
%Dodajemy opis osi
ylabel('y')
xlabel('x')
grid on
hold off
```

Teraz widać, że punktami startowymi dla x mogą być kolejno -8, -6 -2, y dla każdego przypadku może mieć wartość 0. Wyniki będziemy kolekcjonować w wektorze S.

```
S(1)=vpasolve(r1,r2,x,y,[-8,0]);
S(2)=vpasolve(r1,r2,x,y,[-6,0]);
S(3)=vpasolve(r1,r2,x,y,[-2,0]);
S.x
```

```
ans = −7.330759718379841787158019057681 9
ans = −5.234923699448408733553514993868 6
ans = −1.414271110189339840727708167724 9
```

```
S.y
```

```
ans = −0.068852435110532017315385178883978
ans = −0.097321668154604972330460981374425
ans = −0.700756426702251535627748756847 78
```

Punkty możemy przedstawić na wykresie dorysowując je z użyciem funkcji ***plot***. Wcześniej musimy je przerobić z wyrażeń symbolicznych na liczny za pomocą funkcji ***eval***

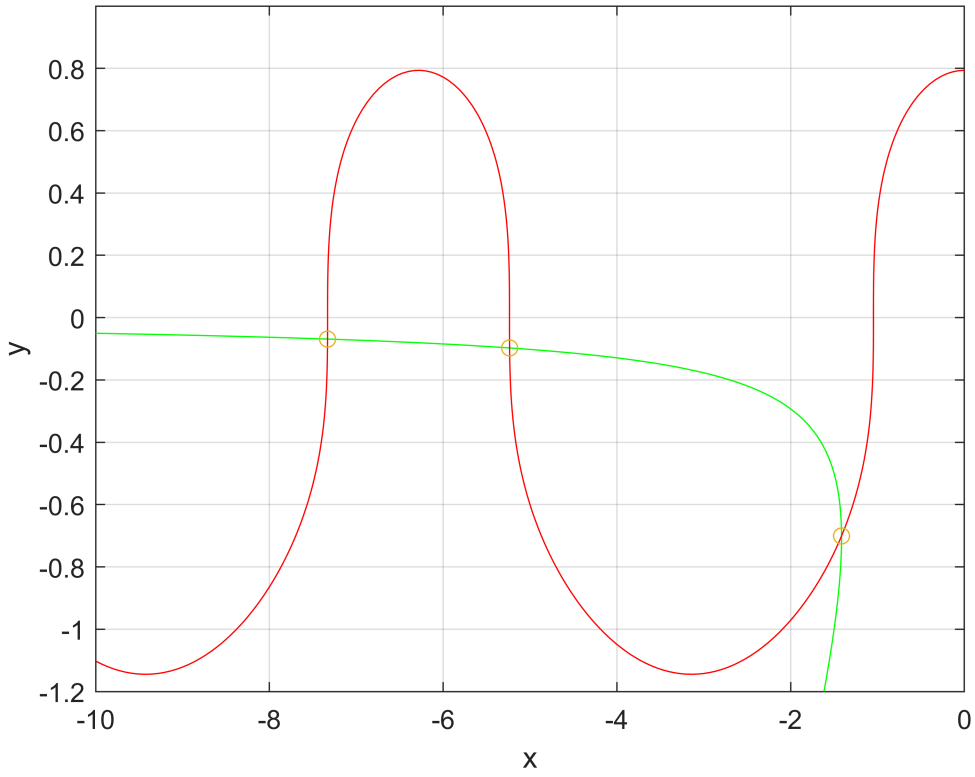
```
sx=[eval(S(1).x),eval(S(2).x),eval(S(3).x)]
```

```
sx = 1×3
   -7.3308   -5.2349   -1.4143
```

```
sy=[eval(S(1).y),eval(S(2).y),eval(S(3).y)]
```

```
sy = 1×3
   -0.0689   -0.0973   -0.7008
```

```
hold on
plot(sx, sy,'o')
hold off
```




UWAGA: Przedstawione tutaj podejście do zapisu równań w postaci zmiennych może zostać także wykorzystane w funkcji ***solve***. Wtedy najpierw definiujemy róania R1, R2 itd, i do funkcji solve zamiast równań możemy wstawić zmienne R1 i R2 itd. Przedstawiona metoda pozyskiwania wartości numerycznych z obliczeń symbolicznych poprzez funkcję ***eval*** jest także słuszna dla wyników uzyskanych metodą ***solve.***

## Rozwiązywanie z wykorzystaniem obliczeń numerycznych układów równań nieliniowych

Rozwiążmy przedstawiony poniżej układ inną metodą bez użycia obliczeń symbolicznych, a z wykorzystaniem zapisu w postaci funkcji. Tym razem nie trzeba definiować zmiennych x i y jako zmienne symboliczne. Spróbujmy rozwiązać układ, którym zajmowaliśmy się wcześniej.

2xy-2y^2=1

y^3-cos(x)=-0.5

Metodyka postępowania podczas rozwiązywania tego układu będzie zgodna z przedstawionymi poniżej krokami.

Po pierwsze układ sprowadzimy do układu równań, w których prawa strona będzie równa 0, czyli:

2xy-2y^2-1=0

y^3-cos(x)+0.5=0

Następnie przygotujemy funkcje dwóch zmiennych odpowiednio zwierające pierwsze i drugie równanie w postaci:

```
g=@(x,y) 2*x.*y-2.*y.^2-1;
h=@(x,y) y.^3-cos(x)+0.5;
```

Użyty w zapisie g=@(x,y) ..  oznacza, że wyrażenie g jest funkcją zmiennych x y. W funkcjach używamy operatorów mnożenia dzielenia i potęgowania z kropką, czyli .*, ./, .^

Do rozwiązania układu równań wykorzystamy funkcję Matlab-a ***fsolve***, ale aby ją użyć najpierw musimy odpowiednio przygotować układ równań. Musimy utworzyć nową strukturę, która zawierała będzie rozwiązywane funkcje. Dodatkowo różne zmienne jak x i y musimy w tej strukturze zastąpić jedną zmienną o odpowiednich indeksach. Tak więc wprowadzimy nową zmienną np. w i przypiszemy zmiennej x zmienną w(1) i zmiennej y zmienną w(2). Teraz nowa struktura zmienna nazwana uk_rown będzie zmieną jednej zmiennej w w postaci:

```
uk_rown=@(w) [g(w(1),w(2));h(w(1),w(2))];
```

Jak widać przygotowana została zmienna zawierająca obydwie wcześniej przygotowane funkcje g i h ułożone jedno równanie pod drugim (stąd oddzielenie średnikiem) a zmienne x i x zostały zastąpione w(1) i w(2)

Aby rozwiązać ten układ równań musimy podać przybliżone rozwiązanie, dlatego na początek narysujmy przebieg naszych funkcji na płaszczyźnie. Wykorzystamy do tego funkcję Matlab-a ***fimplicit***. Kolorem zielonym narysujemy funkcję g, a kolorem czerwonym h. Obydwie funkcje narysujemy w przestrzeni x  i y od -3 do 3.

```
fimplicit(g,[-3,3],'g')
hold on
fimplicit(h,[-3,3],'r')
ylabel('y')
xlabel('x')
grid on
```

Dla pokazanego układu funkcji występuje rozwiązanie w pobliżu x=-2, y=-1, dlatego ten punkt przyjmiemy jako punkt początkowy poszukiwania rozwiązania. Zrobimy to definiując wektor wp jako wektor kolumnowy zawierający jako -2 i -1:

```
wp=[-2;1];
```

Teraz z użyciem funkcji fsolve poszukujemy rozwiązania wstawiając przygotowany  układ równań jako pierwszą zmienną i rozwiązanie początkowe jako drugą zmienną. Otrzymane rozwiązanie przypiszemy jako poszukiwane rozwiązanie x0 i y0 i przedstawimy graficznie na wykresie

```
w0=fsolve(uk_rown,wp);
```

```
Equation solved.

fsolve completed because the vector of function values is near zero
as measured by the value of the function tolerance, and
the problem appears regular as measured by the gradient.

<stopping criteria details>
```

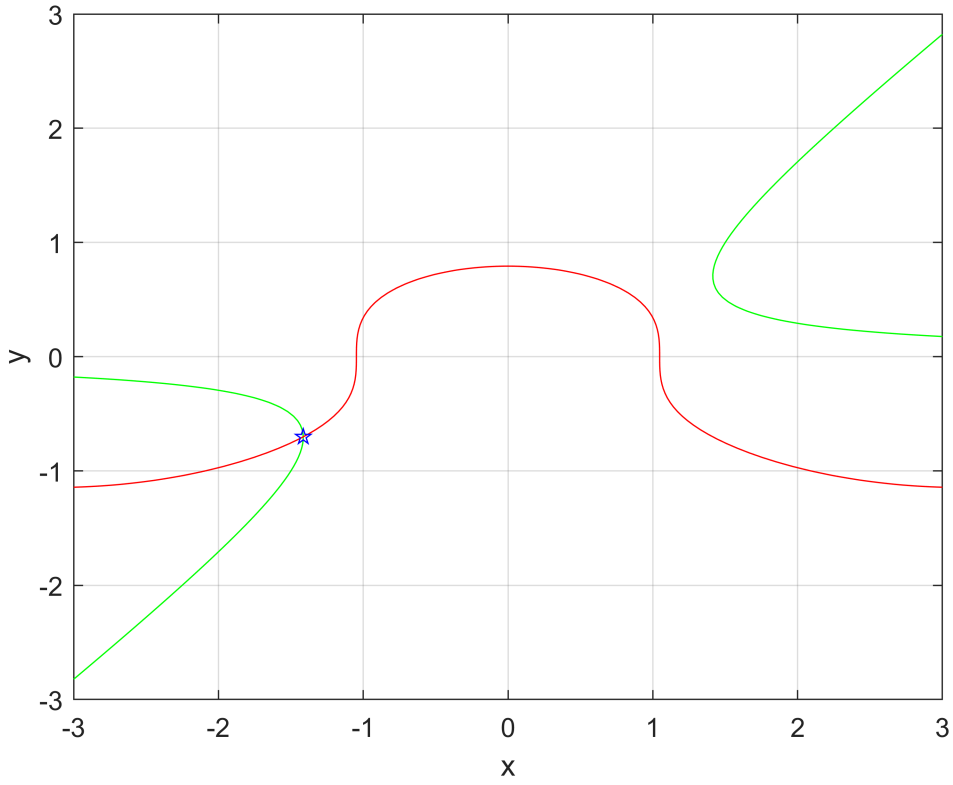
```
x0=w0(1)
```

```
x0 = -1.4143
```

```
y0=w0(2)
```

```
y0 = -0.7008
```

```
plot(x0,y0,'bp')
hold off
```




Prezentowane graficznie rozwiązanie pokazuje, że rozwązanie zostało wyznaczone poprawnie. Jest ono też zgodne z rozwiązaniem uzyskanym wcześniejszymi metodami. W ten sam sposób zmieniając punkt początkowy możemy poszukać pozostałych rozwiązań

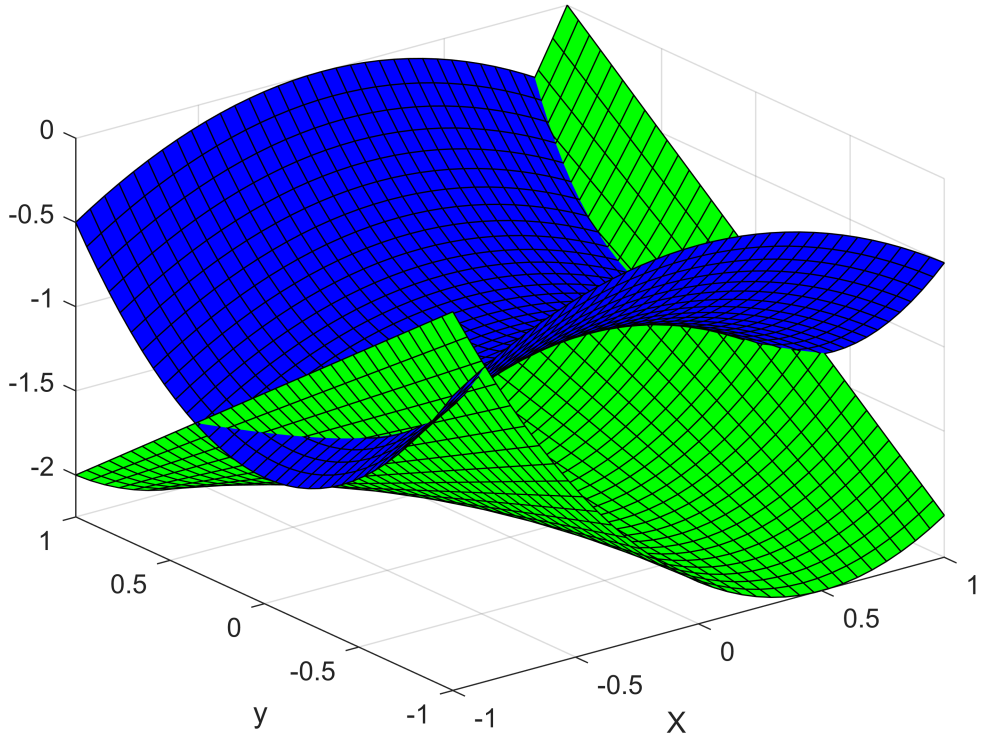
## Rozwiązywanie zagadnień przestrzennych

W kolejnym kroku przyjrzyjmy się jak przedstawioną metodę można wykorzystać do rozwiązania zagadnień przestrzennych. Załóżmy że mamy dwie funkcje dwóch zmiennych, które zapiszemy równaniami:

```
o1=@(x,y) -(x).^2/2+y.^2-1;
o2=@(x,y) x.^2+y.*x-2;
```

Wykorzystamy funkcję ***fsurf*** do narysowania wykresu w przestrzeni. Kolorem zielonym narysujemy funkcję o1 i niebieskim funkcję o2

```
fsurf(o1,[-1 1],'b')
hold on
fsurf(o2, [-1 1],'g')
xlabel('X')
ylabel('y')
hold off
```




Można zaobserwować, że w prezentowanym zakresie obydwie funkcje przecinają się. Spróbujmy ustalić rozwiązanie występujące w przestrzeni. Ograniczmy się do znalezienia rozwiązania w przedziale dla x i y od -1 do 0. Widać, że tutaj rozwiązania będą występowały dla płaszczyzn Z z przedziały -1 do -0.5.

Wybierzmy jakąś płaszczyznę startową i poszukajmy początkowego rozwiązania. Zacznijmy od przecięcia obydwu funkcji płaszczyzną Z=-0.5. Aby zobaczyć odcisk obydwu funkcji na płaszczyźnie Z=-0.5 wykorzystamy podobnie jak poprzednio funkcję ***fimplicit***. Aby obydwie funkcje zaprezentować na płaszczyźnie Z=-0.5 muszą one więc spełniać równania o1=-0.5 i o2=-0.5, czyli po przeniesieniu na drugą stronę dostaniemy nowe funkcje w postaci o11=o1+0.5 i o22=o2+0.5, a zatem utwórzmy nowe zmienne będące funkcjami x i y w postaci:

```
% Tworzenie nowych zmiennych z pierwotnie przyjętych funkcji z
% uwzględnieniem przesunięcia wynikającego z płaszczyzny na której chcemy
% poszukać rozwiązania
o11=@(x,y) o1(x,y)+0.5;
o22=@(x,y) o2(x,y)+0.5;
% Rysujemy wykres obu funkcji na wybranej płaszczyźnie
fimplicit(o11,[-1 0],'b')
hold on
fimplicit(o22,[-1 0],'g')
ylabel('y')
xlabel('x')
grid on
```

W pokazanym przypadku widzimy, że początkowe rozwiązanie można zdefiniować jako -1, -1 (oznaczymy p_p). Przezd przystąpieniem do rozwiązania wcześniej musimy przerobić nasz układ na jeden wspólny układ o wspólnych zmiennych np. jak wcześniej zmiennej w.

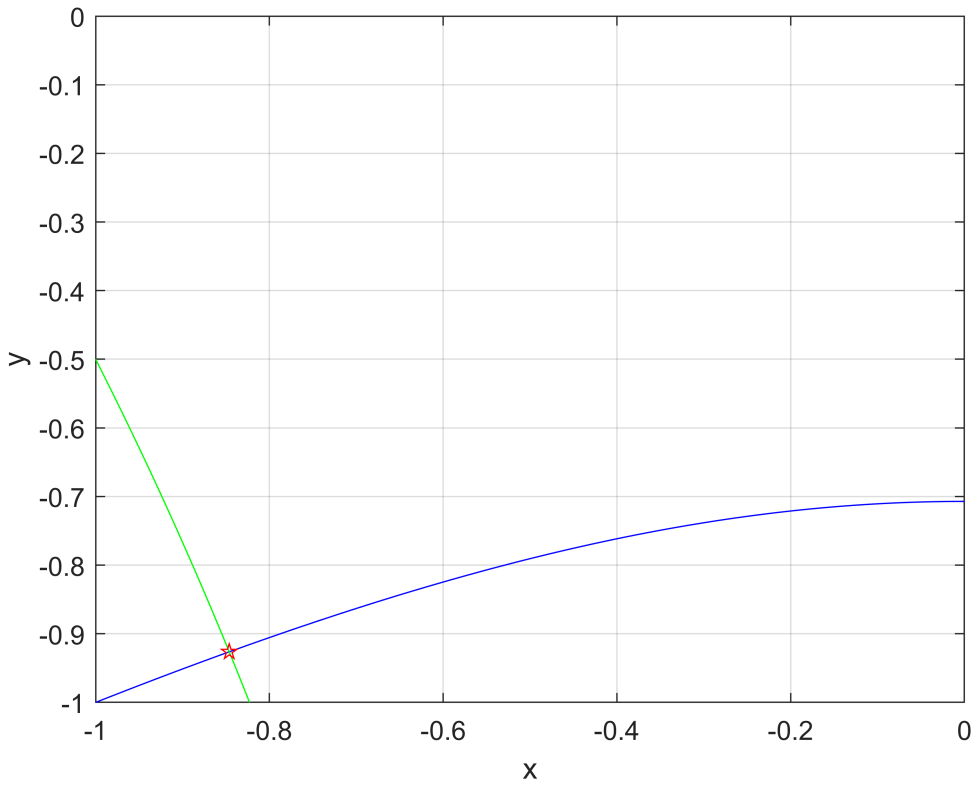
```
uk_rown=@(w) [o11(w(1),w(2));o22(w(1),w(2))];
p_p=[-1; -1];
w0=fsolve(uk_rown,p_p);
```

```
Equation solved.

fsolve completed because the vector of function values is near zero
as measured by the value of the function tolerance, and
the problem appears regular as measured by the gradient.

<stopping criteria details>
```

```
w0
```

```
w0 = 2×1
   -0.8462
   -0.9263
```

```
plot(w0(1),w0(2),'pr')
hold off
```




Rozwiązanie na płaszczyźnie zostało odnalezione. Teraz przenieśmy go na rysunek trójwymiarowy, użyjmy funkcji *plot3*, gdzie x= w0(1) y=w0(2), a z= -0.5 Ponownie narysujmy rysunek 3D ale w mniejszej skali

```
fsurf(o1,[-1 -0.5 -1 0],'b')
```

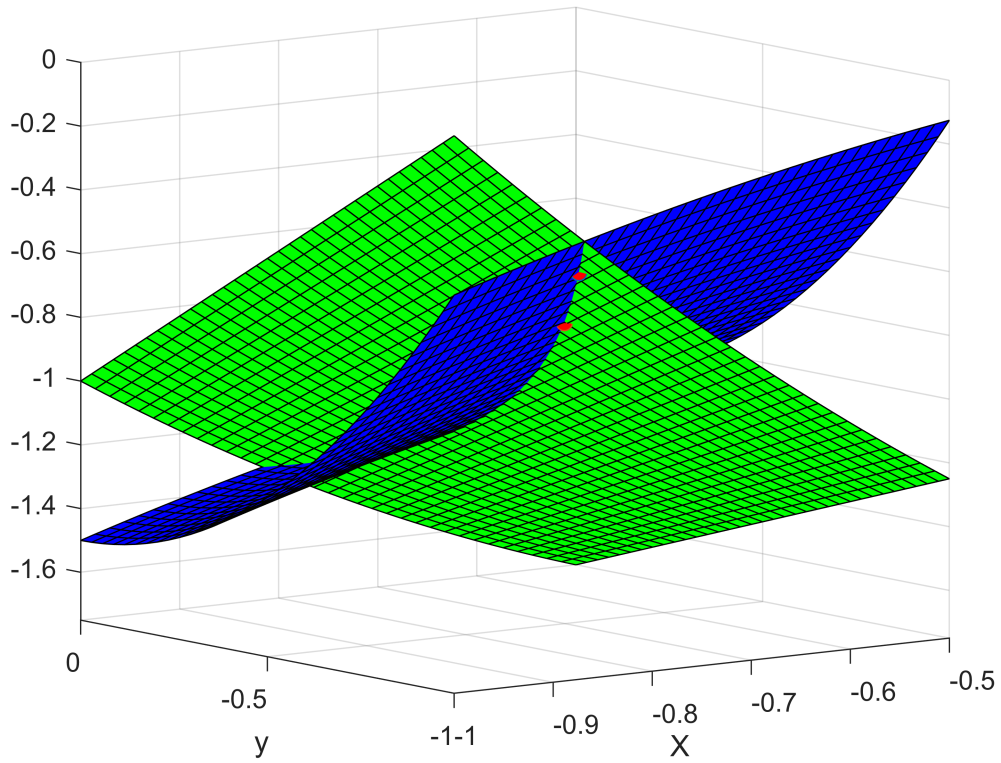
```
hold on
fsurf(o2, [-1 -0.5 -1 0],'g')
xlabel('X')
ylabel('y')
plot3(w0(1),w0(2),-0.5,"Marker","o","MarkerFaceColor",'r')
```

Poszukując kolejnych punktów przecięcia możemy obniżać pałaszczyznę o np. 0.2 od płaszczyzny Z=-0.5 i poszukiwać kolejnych punktów, zakładając że początkowe rozwiązanie jest wynikiem z ostatniego rozwiązania. Przypomnę że obniżenie płaszczyzny Z o 0.2 wiąże się z dodaniem do poprzedniego równania 0.2. Dlatego tworzymy nowe zmienne z wykorzystaniem wcześniejszych zmiennych następująco:

```
o11=@(x,y) o11(x,y)+0.2;
o22=@(x,y) o22(x,y)+0.2;
% Przepisujemy układ równań z nowymi zmiennymi
uk_rown=@(w) [o11(w(1),w(2));o22(w(1),w(2))];
% Nowy punkt startowy zawiera wynik z poprzedniego obliczenia
p_p=w0;
w0=fsolve(uk_rown,p_p,optimoptions('fsolve','Display','off'));
w0
```

```
w0 = 2×1
   -0.8107
   -0.7929
```

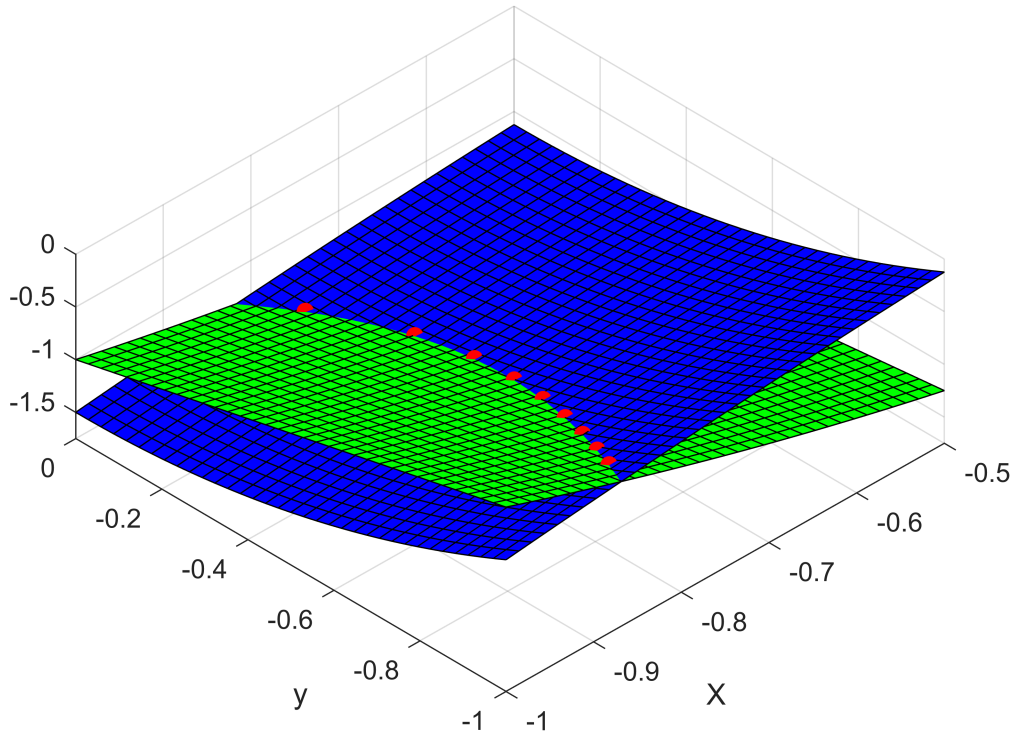
```
plot3(w0(1),w0(2),-0.5-0.2,"Marker","o","MarkerFaceColor",'r')
view([-37.02 9.35])
hold off
```



Obydwa punkty są widoczne na przestrzennym rysunku. Gdybyśmy chcieli wyznaczyć więcej punktów spełniajacych w przestrzeni rozwiązanie układu równań, to proces należałoby zautomatyzować używając iteracji. Wykorzystamy pętlę ***while.*** W pokaznym przypadku wykonamy funkcję while dopuki y-owa składowa jest w przedziale -1 do 0. Obliczenia wykonamy obniżając płaszczyznę poszukiwania rozwiązania Z z krokiem co 0.1

```
% ustalamy początkową płaszczyznę z dla której poszukujemy rozwiązania
% wz=-0.5
wz=-0.5;
% Pierwsze równanie na określonej płaszczyźnie
o11=@(x,y) o1(x,y)-wz(1);
% Drugie równanie na określonej płaszczyźnie
o22=@(x,y) o2(x,y)-wz(1);
uk_rown=@(w) [o11(w(1),w(2));o22(w(1),w(2))];
% Punkt początkowy rozwiązanie
p_p=[-1; -1];
% Rozwiązanie
w0=fsolve(uk_rown,p_p,optimoptions('fsolve','Display','off'));
% Przypisanie zmiennej x pierwszego elementu zmiennej w0 i zmiennej wy
% drugiego elementu zmiennej w0
wx=w0(1);
wy=w0(2);
i=1;
% rozpoczęcie pętli iteracyjnej while
while wy(i)<0 && wy(i)>-1
    % zmienna iteracyjna i
    i=i+1;
    % Określenie nowej płaszczyzny z, która jest kolekcjonowana jako wektor
    % w zmiennej wz
    wz(i)=wz(1)-(i-1).*0.1;
    % Pierwsze i drugie równanie dla nowej płaszczyzny
    o11=@(x,y) o1(x,y)-wz(i);
    o22=@(x,y) o2(x,y)-wz(i);
    uk_rown=@(w) [o11(w(1),w(2));o22(w(1),w(2))];
    %punkt początkowy rozwiązanie wzięty jako rozwiązanie z poprzedniego
    %kroku
    p_p=[wx(i-1);wy(i-1)];
    % Wyznaczenie rozwiązania układu równań
    w0=fsolve(uk_rown,p_p,optimoptions('fsolve','Display','off'));
    %Przypisanie zmiennym wx i wy kolejnych rozwiązań
    wx(i)=w0(1);
    wy(i)=w0(2);
end
% Usunięcie ostatnich elementów z wektora wx,wy,wz. Obliczenia zakończyły
% się po przekroczeniu określonego zakresu obliczeń o jeden punkt ponad
% limit
wx=wx(1:i-1);
wy=wy(1:i-1);
wz=wz(1:i-1);
% Rysowanie wykresu w przestrzeni
fsurf(o1,[-1 -0.5 -1 0],'b')
hold on
```

```
fsurf(o2, [-1 -0.5 -1 0],'g')
xlabel('X')
ylabel('y')

%Rysowanie wyniku rozwiązania w przestrzeni z wykorzystaniem funkcji plot3
plot3(wx,wy,wz,'Color','r',"MarkerFaceColor","r",'LineStyle',"none","Marker","o")
hold off
view([-44.47 62.50])
```




Graficznie przedstawiono rozwiązanie kropkami na wykresie. Równocześnie wartości x,y,z odpowiadajace kolejnym punktom przedstawiono poniżej

```
wx
```

```
wx = 1×9
   -0.8462   -0.8283   -0.8107   -0.7939   -0.7784   -0.7654   -0.7570   -0.7585 ···
```

```
wy
```

```
wy = 1×9
   -0.9263   -0.8620   -0.7929   -0.7177   -0.6348   -0.5412   -0.4319   -0.2961 ···
```

```
wz
```

```
wz = 1×9
   -0.5000   -0.6000   -0.7000   -0.8000   -0.9000   -1.0000   -1.1000   -1.2000 ···
```

## Rozwiązanie zadania z wykorzystaniem układu trzech równań z trzema niewiadomymi

To zadanie można rozwiązać tworząc układ trzech równań z trzema niewiadomymi. Dołóżmy dodatkową zmienną z, która będzie definiować przemieszczanie się względem osi Z. Z równania o1 o zmiennych x i y tworzymy równanie o11 o zmiennych x, y, z odejmując z od o1. Podobnie postępujemy z równaniem o2. Na końcu tworzymy trzecie równanie o zmiennych x, y, z, które ma postać o33= z-a (a jest wartością przemieszczenia względem osi Z). Rozwiązanie realizujemy iteracyjnie zmieniajac wartość a (zmiana z) od -0.5 z krokiem co -0.1 do -1.3.

```
o1=@(x,y) -(x).^2/2+y.^2-1;
o2=@(x,y) x.^2+y.*x-2;
o11=@(x,y,z) o1(x,y)-z;
o22=@(x,y,z) o2(x,y)-z;
a=-0.5:-0.1:-1.3;
for i=1:length(a)
    o33=@(x,y,z) z-a(i);
    uk_rown=@(w) [o11(w(1),w(2),w(3));o22(w(1),w(2),w(3));o33(w(1),w(2),w(3))];
    w0=fsolve(uk_rown,[-1;-1;-1.3],optimoptions('fsolve','Display','off'));
    wx(i)=w0(1);
    wy(i)=w0(2);
    wz(i)=w0(3);
end
wx
```

```
wx = 1×9
   -0.8462   -0.8283   -0.8107   -0.7939   -0.7784   -0.7654   -0.7570   -0.7585 ···
```

```
wy
```

```
wy = 1×9
   -0.9263   -0.8620   -0.7929   -0.7177   -0.6348   -0.5412   -0.4319   -0.2961 ···
```

```
wz
```

```
wz = 1×9
   -0.5000   -0.6000   -0.7000   -0.8000   -0.9000   -1.0000   -1.1000   -1.2000 ···
```

Otrzymany wynik jest zgodny z uzyskanym wcześniej.

Do rozwiązania zadania możemy też wykorzystać metodę z uwzględnieniem obliczeń symbolicznych. Stąd też zdefiniujemy trzy zmienne symboliczne: x, y, z oraz trzy równania. Pierwsze dwa będzie to przyrównanie równań wyjściowych do wartości z i trzecie, gdzie z=a. Zmienna a będzie odpowiadać za zmianę wzdłóż osi Z. Układ równań przedstawia się poniżej oraz zpis w Matlabie przedstawiaa poniższy kod. Zwróćmy uwagę, że w funkcji ***vpasolve*** zdefiniowano te same obszary poszukiwania rozwiązania (zapis w nawiasie kwadratowym). Wynikało to z wcześniejszych obliczeń, gdzie stwierdzono, że w analizowanym przedziale zmiany x, y i z występuje rozwiązanie i jest ono tylko jedno.

```
syms x y z
a=-0.5:-0.1:-1.3;
o1= -(x).^2/2+y.^2-1==z;
o2= x.^2+y.*x-2==z;
for i=1:length(a)
    o3= z==a(i);
    [S0(i)]=vpasolve(o1,o2,o3,x,y,z,[-1,0;-1,0;-1.5,-0.5]);
end
```

```
S0.x
```

```
ans = −0.846237448110746892241053055122 94
ans = −0.828274702019017997762731732592 73
ans = −0.810699605772335141682531488249 76
ans = −0.793869283724202958061980962519 97
ans = −0.778391010604733521655854427656 25
ans = −0.765366864730179543456919968060 8
ans = −0.757005437920716334212591040307 05
ans = −0.758537325961265989192875727523 51
ans = −0.787665441458953463612489450214 33
```

```
S0.y
```

```
ans = −0.926314692365663883552233510043 4
ans = −0.861985783526820428517557331183 51
ans = −0.792853659510826630256463413939 96
ans = −0.717714581028133723275009897090 92
ans = −0.634780499617883021563670695030 92
ans = −0.541196100146196984399723205366 39
ans = −0.431889588344946972579611483387 99
ans = −0.296124023743825865597844616730 42
ans = −0.101036744971144151166466401802 44
```

```
S0.z
```

```
ans = −0.5
ans = −0.6
ans = −0.7
ans = −0.8
ans = −0.9
ans = −1.0
ans = −1.1
ans = −1.2
ans = −1.3
```

Po porównaniu wyników można stwierdzić, że są one takie same. Oczywiści prezentowane tutaj wyniki należałoby zamienić na wartości liczbowe, aby można je było wykorzystać w dalszych obliczeniach.

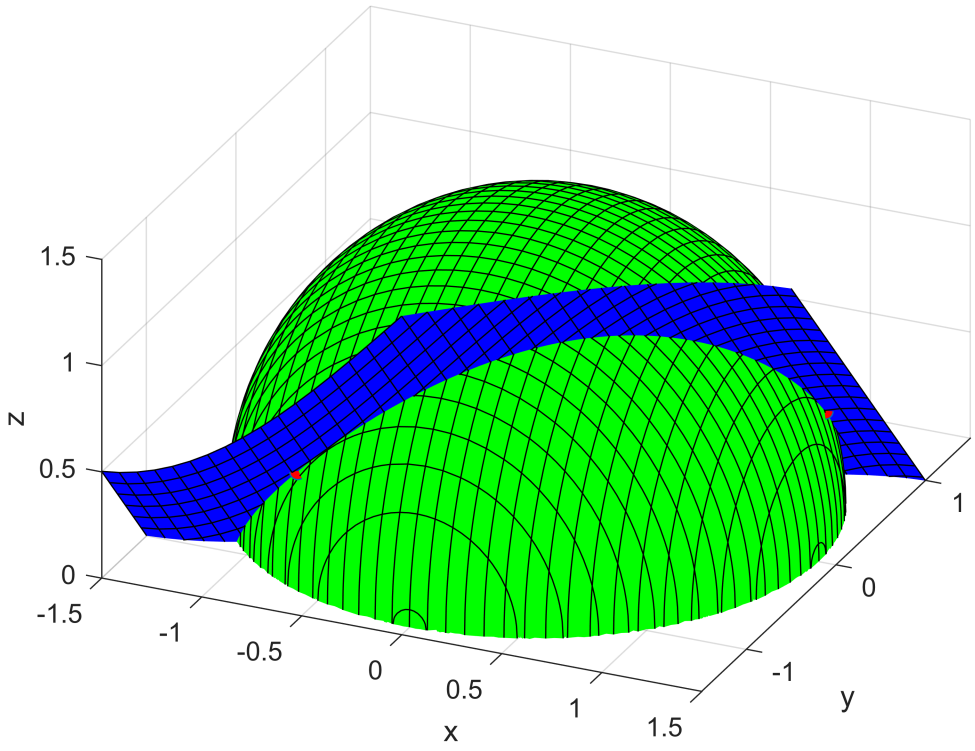
## Zadania do samodzielnego rozwiązania

Zadanie nie podlega ocenie przez prowadzących, jest formą samosprawdzenia wiedzy.

Mamy dwa obiekty w przestrzeni. Jedn opisuje równanie z=(2-x^2-y^2)^0.5, a drugi z=sin(x)-y. Obydwa obiekty przedstawiono na wykresie. Wykorzystując jedną z metod opisanych powyżej znaleźć punkty przecięcia tych

obiektów na płaszczyźnie z=0.5 i pokazać rozwiązanie na wykresie przedstawionym na tej płaszczyźnie (Z=0.5 - użyj funkcji fimplicit do rysowania wykresu na płaszczyźnie)

```
z1= @(x,y)(2-x.^2-y.^2).^0.5;
z2= @(x,y) sin(x)-y;
fsurf(z1,[-2,2],'g')
hold on
fsurf(z2,[-2,2],'b')
xlabel('x')
ylabel('y')
zlabel('z')
axis([-1.5 1.5 -1.5 1.5 0 1.5])
view([24.11 41.05])
x0= [-0.6843 1.2449];
y0= [-1.1321 0.4474];
z0= [0.5 0.5];
plot3(x0,y0,z0,"MarkerFaceColor","r","Marker",'o',"LineStyle","none")
hold off
```




ODPOWIEDŹ: Jako wynik obliczeń powinino wyjść dwa punkty:  P1=(-0.6843, -1.1321) i P2=(1.2449, 0.4474). Pokazano je czerwonymi kropkami na wykresie przestrzennym