Karol Kosoń

Zadanie 12

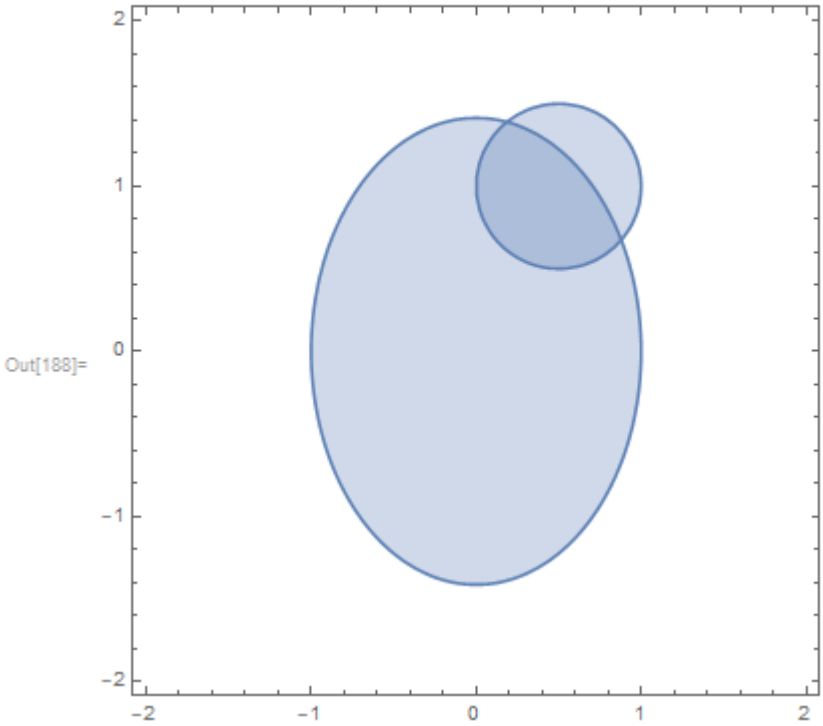
Zadanie 12

Proszę rozwiązać następujący układ równań

$$2x^2 + y^2 = 2, \qquad \left(x - \frac{1}{2}\right)^2 + (y-1)^2 = \frac{1}{4}.$$

Zadanie polega na rozwiązaniu równania. Czyli wyznaczenia przecięcia się dwóch okręgów (jednej elipsy) ze sobą.

```
In[188]:= Show[%187, %186]
          pokaż
```



Skorzystałem z analizy i wyznaczyłem sobie z pierwszego równania x

```
In[124]:= x = Sqrt[(2 - y^2) / 2]
          pierwiastek kwadratowy
```

$$\text{Out[124]=} \quad \frac{\sqrt{2 - y^2}}{\sqrt{2}}$$

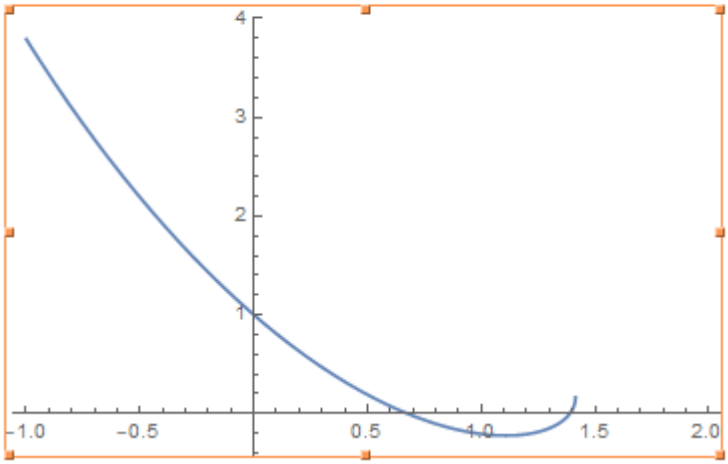
I wstawiłem do drugiego równania:

In[125]:=
```
g = (x - 1 / 2) ^2 + (y - 1) ^2 - 1 / 4
```
Out[125]= $-\frac{1}{4} + (-1+y)^2 + \left(-\frac{1}{2} + \frac{\sqrt{2-y^2}}{\sqrt{2}}\right)^2$

Wykres funkcji jaki uzyskałem:

In[126]:=
```
Plot[g, {y, -1., 2.}]
```
wykres

Out[126]=



Widzimy, że funkcja ma 2 miejsca zerowe. Jedno w okolicach 0.6 drugie w okolicy 1.4.

Do policzenia pierwszego miejsca zerowego wykorzystałem metodę Newtona (pochodną funkcji policzyłem w mathematice).

In[143]:=
```
gPrim = D[g, y]
```
oblicz poc

Out[143]= $2\,(-1+y) - \frac{\sqrt{2}\,y\left(-\frac{1}{2} + \frac{\sqrt{2-y^2}}{\sqrt{2}}\right)}{\sqrt{2-y^2}}$

Zbieżność uzyskałem po ok 6 iteracjach. Wykorzystując ten wynik, wstawiając go do pierwszego równania i wyliczając tym razem miejsce zerowe dla y, uzyskałem kolejny wynik. Są to współrzędne przecięcia się dwóch okręgów jakimi są te równania. Mamy pierwsze rozwiązanie.

Drugie rozwiązanie nie byłem w stanie otrzymać za pomocą mojego programu. Metody stosowane prowadziły do błędów (metoda newtona, metoda siecznych i metoda odwrotnej interpolacji). Punkt zerowy x2 leży za blisko końca dziedziny funkcji i chcąc wyliczyć miejsce zerowe wypadam z dziedziny i program się zawiesza. Drugie punkty przecięcia uzyskałem dzięki Mathematice.

```
In[127]:= NSolve[g]
          rozwiąż numerycznie
Out[127]= {{y → 1.38943}, {y → 0.674013}}

In[137]:= y = 0.6740130461040731`
Out[137]= 0.674013

In[139]:= h = 2 x^2 + y^2 - 2
Out[139]= -1.54571 + 2 x^2

          NSolve[h]
          rozwiąż numerycznie
Out[140]= {{x → -0.879121}, {x → 0.879121}}
```

```
In[154]:= h = 2 x^2 + y^2 - 2
Out[154]= -0.0694891 + 2 x^2

In[155]:= NSolve[h]
          rozwiąż numerycznie
Out[155]= {{x → -0.186399}, {x → 0.186399}}
```

Tutaj jest cały układ równań rozwiązany przez mathematikę:

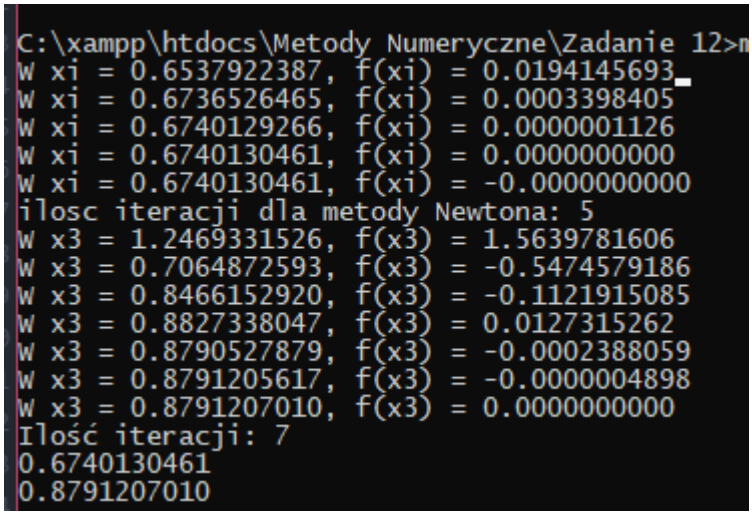
```
In[144]:= k = 2 x^2 + y^2 - 2
Out[144]= -2 + 2 x^2 + y^2

In[145]:= l = (x - 1 / 2) ^2 + (y - 1) ^2 - 1 / 4
Out[145]= -1/4 + (-1/2 + x)^2 + (-1 + y)^2

In[147]:= NSolve[{k == 0, l == 0}, {x, y}]
          rozwiąż numerycznie
Out[147]= {{x → -1.53276 + 1.93731 i, y → 2.96828 + 2.00077 i},
           {x → -1.53276 - 1.93731 i, y → 2.96828 - 2.00077 i},
           {x → 0.186399, y → 1.38943}, {x → 0.879121, y → 0.674013}}
```

Czyli jeden z wyników osiągniętych przez mój program zgadza się z wynikami otrzymanymi w Mathematice.

Wynik mojego programu:



KOD PROGRAMU:

```
#include <math.h>
#include <iostream>
#include <cstdio>
#include <iomanip>
#include <cmath>
#include <cstdlib>
#include <ctime>

#define epsilon 0.0000001

using namespace std;

typedef double( * function )( double );

double fx(double x){
   double a = sqrt(2.0-x*x) / sqrt(2.0);
   double b = (x - 1.0);

   return (a - 0.5)*(a - 0.5) + b*b - 1.0/4.0;
}
```

```
double yx(double x, double y){
   return 2.0*x*x + y*y - 2.0;
}

double fxPrim(double x){
double up = sqrt(2.0)*x*(-0.5 + (sqrt(2.0 - x*x) / sqrt(2.0)));
double down = sqrt(2.0 - x*x);

return 2.0*(-1.0 + x) - (up / down);
}

double FunctionValue(double x, function pDzialanie){
   return pDzialanie(x);
}



double secantMethodFirstFunction(double x1, double x2){
   double f1, f2, f3, x3, tmpX3;

   int i = 0;

   while(1)
 {
     f1 = fx(x1);
     f2 = fx(x2);
     x3 = (f1 * x2 - f2 * x1)/( f1 - f2);
     f3 = fx(x3);

   if(f1 == f2 )
```

```
    {
      cout << "Złe punkty startowe, f1 = f2\n";
      break;
    }

    if((abs(x3 - tmpX3) < epsilon) && fx(x3) < epsilon) break;
    cout << "W x3 = "<< x3 << ", f(x3) = " << f3 << endl;
    x2 = x1;
    x1 = x3;
    tmpX3 = x3;

    i++;
  }

    cout << "Ilość iteracji: " << i << endl;

    return x3;
}

double secantMethodSecondFunctionFunction(double x1, double x2, double y){
    double f1, f2, f3, x3, tmpX3;

    int i = 0;

    while(1)
  {
      f1 = yx(x1, y);
      f2 = yx(x2, y);
      x3 = (f1 * x2 - f2 * x1)/( f1 - f2);
      f3 = yx(x3, y);
```

```
    if(f1 == f2 )
    {
      cout << "Złe punkty startowe, f1 = f2\n";
      break;
    }

    if((abs(x3 - tmpX3) < epsilon) && yx(x3, y) < epsilon) break;
    cout << "W x3 = "<< x3 << ", f(x3) = " << f3 << endl;
    x2 = x1;
    x1 = x3;
    tmpX3 = x3;

    i++;
  }

    cout << "Ilość iteracji: " << i << endl;

    return x3;
}

double newtonMethod(double x){
    double x0 = x;
    double xi;
    int i = 0;
    while(1){
        i++;
        xi = x0 - fx(x0)/fxPrim(x0);
        cout << "W xi = "<< xi << ", f(xi) = " << fx(xi) << endl;
        if(abs(x0 - xi) < epsilon) break;
        x0 = xi;
    }
```

```
    cout << "ilosc iteracji dla metody Newtona: " << i << endl;
    return xi;
}

int main(){

    cout << setprecision(10)     // 8 cyfr po przecinku
        << fixed;
    double a = 0.2;
    double b = 0.5;
    double c = 1.2;
    // double y0 = secantMethodFirstFunction(a,b);
    double y0 = newtonMethod(b);
    double x0 = secantMethodSecondFunctionFunction(a,b,y0);
    cout << y0 <<endl;
    cout << x0 << endl;



    return 0;
}
```