Zadanie 14:

14. Rozwiąż układ równań

$$2x^2 + y^2 = 2 \tag{13a}$$

$$\left(x - \frac{1}{2}\right)^2 + (y-1)^2 = \frac{1}{4} \tag{13b}$$

Kod w języku Python:

```python
import math


def solver(xi, yi, eps):
    print("Starting from: (", xi, ",", yi, ")")
    i=0
    for i in range(0, 100):
        dx = (-f(xi, yi) * gy(xi, yi) + g(xi, yi) * fy(xi, yi)) / Jacob(xi, yi)
        dy = (-g(xi, yi) * fx(xi, yi) + f(xi, yi) * gx(xi, yi)) / Jacob(xi, yi)
        xip1 = xi + dx
        yip1 = yi + dy
        Errx = abs((xip1 - xi) / xi)
        Erry = abs((yip1 - yi) / yi)
        if Errx < eps and Erry < eps:
            break
        xi = xip1
        yi = yip1
    print("Znaleziono po ",i, "iteracjach")
    if i < 100:
        print ("Roots of equation: ", xi, ",", yi)
    else:
        print ("Not convergent in x0,y0")


def f(x, y):  # wartość f w punkcie x,y
    return (2*pow(x, 2) + pow(y, 2) - 2)


def g(x, y):  # wartość g w punkcie x,y
    return (pow(x - (1 / 2), 2)) + (pow(y - 1, 2) - 1 / 4)


def fx(x, y):  # pochodna f po x
    return 4 * x


def fy(x, y):  # pochodna f to y
    return 2 * y


def gx(x, y):  # pochodna g po x
    return 2 * x - 1


def gy(x, y):  # pochodna g po y
    return 2 * y - 2
```

```
def Jacob(x, y):
    return fx(x, y) * gy(x, y) - fy(x, y) * gx(x, y)


eps = 0.00001
print("Maximum ε = ", eps)
xi = 0.1
yi = 1.0

solver(xi, yi, eps)

xi = 1.0
yi = 0.1
solver(xi, yi, eps)
```

Wynik działania programu:

```
Maximum ε =  1e-05
Starting from: ( 0.1 , 1.0 )
Znaleziono po  4 iteracjach
Roots of equation:  0.18639765526839308 , 1.3894296391200378
Starting from: ( 1.0 , 0.1 )
Znaleziono po  4 iteracjach
Roots of equation:  0.8791208346050541 , 0.6740129068000752
```

Można zauważyć, że zadane równania opisują dwie elipsy (a dokładniej elipsę i okrąg) na płaszczyźnie. Rozwiązań tych równań może być 0 (elipsy się nie przecinają), 1 (brzegi elips stykają się w tym samym punkcie), 2 (jak w tym przypadku), 3 (spłaszczone elipsy, które się przecinają w dwóch miejscach i wewnętrznie w jednym) lub 4 (przykładowo dwie mozno spłaszczone elipsy, jedna wzdłuż OX, druga wzdłuż OY).
Miałem na uwadze, że metoda może nie być zbieżna dla pewnych zadanych punktów, dlatego w takim przypadku, jeśli nie otrzymamy wyniku po pewnej ilości kroków, obliczenia są przerywane i wypisywany jest stosowny komunikat.

Program wykorzystuje metodę Newtona przedstawioną na wykładzie 9.
Potrzebujemy pochodnych cząstkowych funkcji f(x,y) i g(x,y); wymagana jest znajomość analitycznych wzorów tych pochodnych (slajd 39).

Dla układu dwóch równań:

$$f_1(x, y) = 0$$
$$f_2(x, y) = 0$$

Kolejne kroki dx obliczamy korzystając z reguły Cramera:

$$\begin{cases} \Delta x = \dfrac{-f_1(x_1, y_1)\left.\dfrac{\partial f_2}{\partial y}\right|_{x_1, y_1} + f_2(x_1, y_1)\left.\dfrac{\partial f_1}{\partial y}\right|_{x_1, y_1}}{J(f_1(x_1, y_1), f_2(x_1, y_1))} \\ \Delta y = \dfrac{-f_2(x_1, y_1)\left.\dfrac{\partial f_1}{\partial x}\right|_{x_1, y_1} + f_1(x_1, y_1)\left.\dfrac{\partial f_2}{\partial x}\right|_{x_1, y_1}}{J(f_1(x_1, y_1), f_2(x_1, y_1))} \end{cases}$$

gdzie J – jakobian: $J(f_1, f_2) = \det\begin{bmatrix} \dfrac{\partial f_1}{\partial x} & \dfrac{\partial f_1}{\partial y} \\ \dfrac{\partial f_2}{\partial x} & \dfrac{\partial f_2}{\partial y} \end{bmatrix}$

Zatem nowe przybliżenie rozwiązań: $x_2 = x_1 + \Delta x$, $y_2 = y_1 + \Delta y$

gdzie J – jakobian: $J(f_1, f_2) = \det\begin{bmatrix} \dfrac{\partial f_1}{\partial x} & \dfrac{\partial f_1}{\partial y} \\ \dfrac{\partial f_2}{\partial x} & \dfrac{\partial f_2}{\partial y} \end{bmatrix}$

Zatem nowe przybliżenie rozwiązań: $x_2 = x_1 + \Delta x$, $y_2 = y_1 + \Delta y$

Jezeli znajdziemy rozwiązanie $x_i > \varepsilon$, gdzie $\varepsilon > 0$ jest pożądaną tolerancją, należy spróbować rozpocząć z innym warunkiem początkowym. Szansa na znalezienie numerycznego rozwiązania układu równań jest tym większa, im lepszy jest warunek początkowy.