2.3.1. Implementati metoda lui Newton si metoda aproximatiilor succesive.

Solutie: A se vedea functiile de la finalul documentului.

2.4.1. Rezolvati sistemul

$$\begin{cases} x^2 + y^2 &= 1 \\ x^3 - y &= 0 \end{cases}$$

Solutie: Se cauta radacinile functiei $f : \mathbb{R}^2 \to \mathbb{R}^2, f(x, y) = (x^2 + y^2 - 1, x^3 - y)$.

Alegem sa folosim metoda lui Newton:

$$\mathrm{x}^{(n+1)} = \mathrm{x}^{(n)} - (\mathrm{J}_f(\mathrm{x}^{(n)}))^{-1} f(\mathrm{x}^{(n)})$$

unde

$$\mathrm{J}_f = \begin{pmatrix} \frac{\partial f_1}{\partial x} & \frac{\partial f_1}{\partial y} \\ \frac{\partial f_2}{\partial x} & \frac{\partial f_2}{\partial y} \end{pmatrix} = \begin{pmatrix} 2x & 2y \\ 3x^2 & -1 \end{pmatrix}$$

```
f = @(x)[x(1)^2 + x(2)^2 - 1; x(1)^3 - x(2)];
df = @(x)[2 * x(1), 2 * x(2); 3 * x(1)^2, -1];
x0 = [1;0];
xs = NewtonRn(f, df, x0)
```

```
xs = 2×1
    0.8260
    0.5636
```

```
err =  mean(abs(f(xs))) % fsolve not working on my setup
```

```
err = 1.4279e-10
```

2.4.2. Rezolvati numeric sistemul

$$\begin{cases} 9x^2 + 36y^2 + 4z^2 - 36 &= 0 \\ x^2 - 2y^2 - 20z &= 0 \\ x^2 - y^2 + z^2 &= 0 \end{cases}$$

utilizand metoda lui Newton si metoda aproximatiilor succesive. *Indicatie.*Sunt 4 solutii. Valori bune de pornire: $[\pm 1, \pm 1, 0]^T$.

Solutie.

$$J_f = \begin{pmatrix} \frac{\partial f_1}{\partial x} & \frac{\partial f_1}{\partial y} & \frac{\partial f_1}{\partial z} \\ \frac{\partial f_2}{\partial x} & \frac{\partial f_2}{\partial y} & \frac{\partial f_2}{\partial z} \\ \frac{\partial f_3}{\partial x} & \frac{\partial f_3}{\partial y} & \frac{\partial f_3}{\partial z} \end{pmatrix} = \begin{pmatrix} 18x & 72y & 8z \\ 2x & -4y & -20 \\ 2x & -2y & 2z \end{pmatrix}$$

```
f = @(x)[9*x(1)^2+36*x(2)^2+ 4*x(3)^2-36;
           x(1)^2- 2*x(2)^2-20*x(3)     ;
           x(1)^2-   x(2)^2+   x(3)^2   ];
df = @(x)[18*x(1), 72*x(2), 8*x(3); 2*x(1), -4*x(2), -20; 2*x(1), -2*x(2), 2*x(3)];
x0 = [1; 1; 0];
```

Prin metoda lui Newton:

```
xsN = NewtonRn(f, df, x0)
```

```
xsN = 3×1
    0.8936
    0.8945
   -0.0401
```

Prin metoda aproximatiilor succesive:

```
xsA = SuccesiveApprox(f, df, x0, 1e-9, 50)
```

```
xsA = 3×1
    0.8936
    0.8945
   -0.0401
```

Helpers:

```
function [z, ni] = NewtonR(f, fd, x0, ea, er, Nmax)
% f - functia R->R
% fd - derivata
% x0 - valoarea de pornire
% ea, er - eroarea absolută, respectiv relativă
% Nmax - numărul maxim de iterații
    if (nargin < 6) Nmax = 50; end
    if (nargin < 5) er = 0; end
    if (nargin < 4) ea = 1E-3; end
    xv = x0;
    for k = 1 : Nmax
        xc = xv - f(xv) / fd(xv);
        if (abs(xc - xv) < ea + er * xc) % Succes
            z=xc; ni=k;
            return
        end
```

```
        xv = xc; % Iterația următoare
    end
    % Eșec
    error('S-a depășit numărul maxim de iterații.')
end
function [z,ni] = NewtonRn(f,fd,x0,ea,er,nmax)
% f - functia R^n->R^n
% fd - derivata (Jacobianul lui f)
% x0 - aproximația inițială
% ea - eroarea absolută (implicit 1E-3)
% er - eroarea relativă (implicit 0)
% Nmax - numărul maxim de iterații
    if (nargin < 6) nmax = 50; end
    if (nargin < 5) er = 0; end
    if (nargin < 4) ea = 1E-3; end

    xp = x0(:); % x precedent
    for k = 1 : nmax
        xc = xp - feval(fd, xp) \ feval(f, xp);
        if (norm(xc - xp, inf) < ea + er * norm(xc, inf)) % Succes
            z = xc; ni = k;
            return
        end
        xp = xc;
    end

    % Eșec
    error('S-a depășit numărul maxim de iterații.');
end

function [z, ni] = SuccesiveApprox(f, fd, x0, err, nmax)
% f - sistem de funcții liniare
% fd - derivatele lui f
% x0 - aproximația inițială
% err - toleranța
% nmax - numărul maxim de iterații
    i = 1;
    xp = x0; % xp - x anterior
    lambda = fd(x0);

    while i <= nmax
        xc = xp - lambda \ f(xp); % xc - x curent
        if (norm(xc - xp, inf) < err) % Succes
            z  = xc; ni = i;
            return;
        end
        xp = xc; i = i + 1;
    end
    % Eșec
    error('S-a depășit numărul maxim de iterații');
end
```