Problema 4

a) Implementați metoda lui Newton pentru rădăcini multiple.

Metoda lui Newton pentru **radacini simple** are rata de convergenta cel putin patratica, insa pentru radacini multiple, aceeasi metoda are rata de convergenta liniara.

Pentru a imbunatati rata de convergenta, putem folosi urmatoarea relatie de recurenta:

$$x_{k+1} = x_k - m \cdot \frac{f(x_k)}{f'(x_k)}, m = \text{ordin de multiplicitate}$$

Cu rata de convergenta: $\frac{m-1}{m}$

Daca insa m este necunoscut, il putem aproxima prin:

$$m \approx \frac{\ln\left|\frac{f(x_{k-1})}{f(x_{k-2})}\right|}{\ln\left|\frac{x_{k-1}-x_k}{x_{k-2}-x_k}\right|}$$

b) Se consideră ecuația $\frac{1}{2}x^2 + x + 1 - e^x = 0$, pe [-1, 1]. Ce se întâmplă dacă se aplică metoda lui Newton cu x0 = 0?

Pentru $x_0 = 0 : f(x_0) = 0, f'(x_0) = 0$. Avem caz de nedeterminare $\frac{0}{0}$.

c) Remarcați convergența lentă, explicați fenomenul și găsiți un remediu.

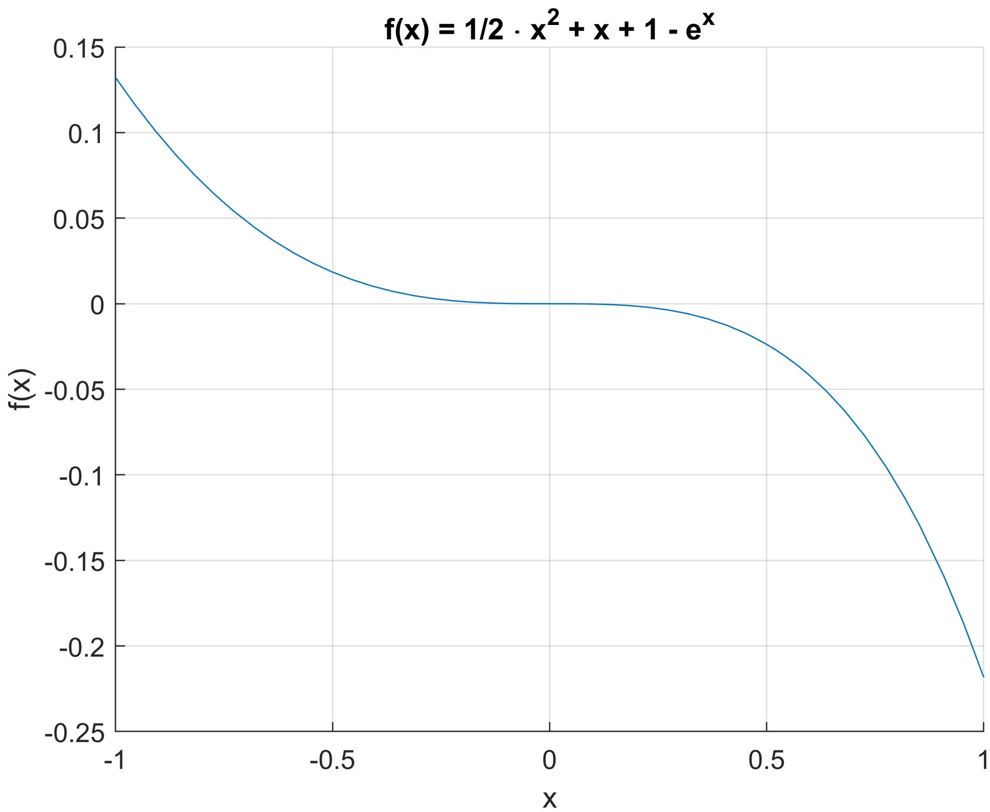
Insa daca nu cunoastem ordinul de multiplicitate, convergenta este foarte lenta. Am putea empiric sa incercam sa ghicim ordinul de multiplicitate, astfel ajungand la o solutie. Din grafic se observa ca pe [-1, 1] avem o singura radacina, aceea fiind chiar x0 = 0. Convergenta este lenta nu are remediu in cazul acesta: avem o singura radacina, multiplicitatea 1 pe $\mathbb{R}$, insa dat fiind faptul ca avem caz de nedeterminare chiar din solutie, aproximarea initiala trebuie sa fie cat mai apropiata de 0, de ex $1 * 10^{-6}$.

d) Când este mai mic numărul de iterații: când se cunoaște multiplicitatea sau se estimează?

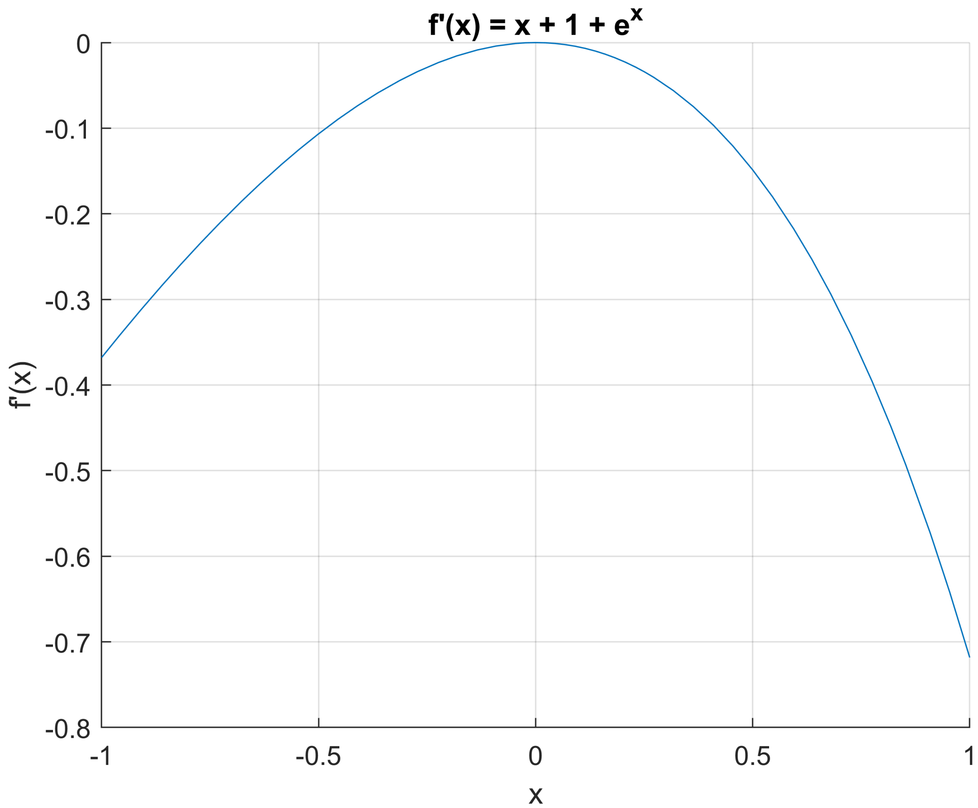
Numărul de iterații este același în ambele cazuri, având multiplicitatea 1.

```
f = @(x) 0.5 .* x .^ 2 + x + 1 - exp(x);
df = @(x) x + 1 - exp(x);

figure; hold on;
fplot(f, [-1, 1])
title('f(x) = 1/2 \cdot x^2 + x + 1 - e^x');
grid on;
xlabel('x');
ylabel('f(x)');
```




```
figure; hold on;
fplot(df, [-1, 1]);
title("f'(x) = x + 1 + e^x")
grid on;
xlabel('x');
ylabel("f'(x)")
```




```
[root0, iter0] = newton_multiple_roots(f, df, 0, 1);
```

```
Warning: Approximation did not converge in 10000 maximum iterations
```

```
fprintf("[x0 = 0] Root: %.16e | Iterations: %d\n", root0, iter0);
```

```
[x0 = 0] Root: NaN | Iterations: 9999
```

```
[root05, iter05] = newton_multiple_roots(f, df, 0.5, 1);
fprintf("[x0 = 0.5] Root: %.16e | Iterations: %d\n", root05, iter05);
```

```
[x0 = 0.5] Root: 9.3367371506939760e-06 | Iterations: 28
```

```
[root_near0, iter_near0] = newton_multiple_roots(f, df, 1e-7);
fprintf("[x0 = 1e-7] Root: %.16e | Iterations: %d\n", root_near0, iter_near0);
```

```
[x0 = 1e-7] Root: 9.9999999999999995e-08 | Iterations: 3
```

Problema 5

Deduceți o formulă de cuadratură de forma

$$\int_{-1}^{1} f(t)\mathrm{d}t = A_0 f(-1) + A_1 f(t_1) + A_2 f(t_2) + A_3 f(t_3) + A_4 f(1) + R_f$$

care să aibă grad maxim de exactitate.

Intervalul și ponderea se potrivesc cu valorile pentru polinoamele ortogonale Legendre.

```
syms t k integer
assumeAlso(k >= 0)

f = t^k;
intf = int(f, t, -1, 1);
disp(intf);
```

$$\frac{(-1)^k+1}{k+1}$$

```
for i = 0:7
    I = vpa(subs(intf, k, i), 6);
    fprintf("k = %2d | I = %.6e\n", i, I);
end
```

```
k =  0 | I = 2.000000e+00
k =  1 | I = 0.000000e+00
k =  2 | I = 6.666667e-01
k =  3 | I = 0.000000e+00
k =  4 | I = 4.000000e-01
k =  5 | I = 0.000000e+00
k =  6 | I = 2.857143e-01
k =  7 | I = 0.000000e+00
```

$$\int_{-1}^{1} t^k \mathrm{dt} = \frac{(-1)^k+1}{k+1}$$

$$\begin{bmatrix} f(t) & f(-1) & f(t_1) & f(t_2) & f(t_3) & f(1) & \int_{-1}^{1} f(t)\mathrm{dt} \\ 1 & 1 & 1 & 1 & 1 & 1 & 2 \\ t & -1 & t_1 & t_2 & t_3 & 1 & 0 \\ t^2 & 1 & t_1^2 & t_2^2 & t_3^2 & 1 & \frac{2}{3} \\ t^3 & -1 & t_1^3 & t_2^3 & t_3^3 & 1 & 0 \\ t^4 & 1 & t_1^4 & t_2^4 & t_3^4 & 1 & \frac{2}{5} \\ t^5 & -1 & t_1^5 & t_2^5 & t_3^5 & 1 & 0 \\ t^6 & 1 & t_1^6 & t_2^6 & t_3^6 & 1 & \frac{2}{7} \\ t^7 & -1 & t_1^7 & t_2^7 & t_3^7 & 1 & 0 \end{bmatrix}$$

Rezulta sistemul:

$$f(t) = 1 : A_0 + A_1 + A_2 + A_3 + A_4 = 2$$
$$f(t) = t : -A_0 + t_1A_1 + t_2A_2 + t_3A_3 + A_4 = 0$$
$$f(t) = t^2 : A_0 + t_1^2A_1 + t_2^2A_2 + t_3^2A_3 + A_4 = \frac{2}{3}$$
$$f(t) = t^3 : -A_0 + t_1^3A_1 + t_2^3A_2 + t_3^3A_3 + A_4 = 0$$
$$f(t) = t^4 : A_0 + t_1^4A_1 + t_2^4A_2 + t_3^4A_3 + A_4 = \frac{2}{5}$$
$$f(t) = t^5 : -A_0 + t_1^5A_1 + t_2^5A_2 + t_3^5A_3 + A_4 = 0$$
$$f(t) = t^6 : A_0 + t_1^6A_1 + t_2^6A_2 + t_3^6A_3 + A_4 = \frac{2}{7}$$
$$f(t) = t^7 : -A_0 + t_1^7A_1 + t_2^7A_2 + t_3^7A_3 + A_4 = 0$$

```
syms A0 A1 A2 A3 A4 t1 t2 t3

eq1 = A0 + A1 + A2 + A3 + A4 == 2;
eq2 = -A0 + t1 * A1 + t2 * A2 + t3 * A3 + A4 == 0;
eq3 = A0 + t1^2 * A1 + t2^2 * A2 + t3^2 * A3 + A4 == 2/3;
eq4 = -A0 + t1^3 * A1 + t2^3 * A2 + t3^3 * A3 + A4 == 0;
eq5 = A0 + t1^4 * A1 + t2^4 * A2 + t3^4 * A3 + A4 == 2/5;
eq6 = -A0 + t1^5 * A1 + t2^5 * A2 + t3^5 * A3 + A4 == 0;
eq7 = A0 + t1^6 * A1 + t2^6 * A2 + t3^6 * A3 + A4 == 2/7;
eq8 = -A0 + t1^7 * A1 + t2^7 * A2 + t3^7 * A3 + A4 == 0;

solution = solve([eq1, eq2, eq3, eq4, eq5, eq6, eq7, eq8], [A0, A1, A2, A3, A4, t1, t2, t3]);
fields = fieldnames(solution);
for i = 1:length(fields)
    fprintf('%s = ', fields{i});
    disp(vpa(solution.(fields{i}), 6));
    fprintf('\n');
end
```

A0 =

$$\begin{pmatrix} 0.1 \\ 0.1 \\ 0.1 \\ 0.1 \\ 0.1 \\ 0.1 \end{pmatrix}$$

A1 =

$$\begin{pmatrix} 0.544444 \\ 0.544444 \\ 0.544444 \\ 0.544444 \\ 0.711111 \\ 0.711111 \end{pmatrix}$$

A2 =

$$\begin{pmatrix} 0.544444 \\ 0.544444 \\ 0.711111 \\ 0.711111 \\ 0.544444 \\ 0.544444 \end{pmatrix}$$

A3 =

$$\begin{pmatrix} 0.711111 \\ 0.711111 \\ 0.544444 \\ 0.544444 \\ 0.544444 \\ 0.544444 \end{pmatrix}$$

A4 =

$$\begin{pmatrix} 0.1 \\ 0.1 \\ 0.1 \\ 0.1 \\ 0.1 \\ 0.1 \end{pmatrix}$$

t1 =

$$\begin{pmatrix} 0.654654 \\ -0.654654 \\ 0.654654 \\ -0.654654 \\ 0 \\ 0 \end{pmatrix}$$

t2 =

$$\begin{pmatrix} -0.654654 \\ 0.654654 \\ 0 \\ 0 \\ 0.654654 \\ -0.654654 \end{pmatrix}$$

t3 =

$$\begin{pmatrix} 0 \\ 0 \\ -0.654654 \\ 0.654654 \\ -0.654654 \\ 0.654654 \end{pmatrix}$$

Rezulta solutiile:

$$A0 = \begin{pmatrix} 0.1 \\ 0.1 \\ 0.1 \\ 0.1 \\ 0.1 \\ 0.1 \end{pmatrix}$$

$$A1 = \begin{pmatrix} 0.544444 \\ 0.544444 \\ 0.544444 \\ 0.544444 \\ 0.711111 \\ 0.711111 \end{pmatrix}$$

$$A2 = \begin{pmatrix} 0.544444 \\ 0.544444 \\ 0.711111 \\ 0.711111 \\ 0.544444 \\ 0.544444 \end{pmatrix}$$

$$A3 = \begin{pmatrix} 0.711111 \\ 0.711111 \\ 0.544444 \\ 0.544444 \\ 0.544444 \\ 0.544444 \end{pmatrix}$$

$$A4 = \begin{pmatrix} 0.1 \\ 0.1 \\ 0.1 \\ 0.1 \\ 0.1 \\ 0.1 \end{pmatrix}$$

$$t1 = \begin{pmatrix} 0.654654 \\ -0.654654 \\ 0.654654 \\ -0.654654 \\ 0 \\ 0 \end{pmatrix}$$

$$t2 = \begin{pmatrix} -0.654654 \\ 0.654654 \\ 0 \\ 0 \\ 0.654654 \\ -0.654654 \end{pmatrix}$$

$$t3 = \begin{pmatrix} 0 \\ 0 \\ -0.654654 \\ 0.654654 \\ -0.654654 \\ 0.654654 \end{pmatrix}$$

Alegem prima solutie, adica:

$$\int_{-1}^{1} f(t)\mathrm{dt} = 0.1 + 0.544444 \cdot f(0.654654) + 0.544444 \cdot f(-0.654654) + 0.711111 \cdot f(0) + 0.1 + R(f)$$

$$R(f) = \frac{f^{(8)}(\xi)}{8!} \cdot \int_{-1}^{1} \omega(t)\mathrm{dt}, \text{unde } \omega(t) = (t+1)(t-t_1)(t-t_2)(t-t_3)(t-1), \xi \in (-1, 1)$$

$$\int_{-1}^{1} \omega(t)\mathrm{dt} = \frac{4t_1 + 4t_2 + 4t_3 + 20t_1t_2t_3}{15}, \text{iar cu valorile aproximate} \Rightarrow \omega(t) = 0 \Rightarrow R(f) = 0$$

```
syms t t1 t2 t3

omega = (t+1)*(t-t1)*(t-t2)*(t-t3)*(t-1);
int_omega = int(omega, t, -1, 1);
fprintf("I_omega = ");
```

```
I_omega =
```

```
disp(int_omega);
```

$$\frac{4\,t_1}{15} + \frac{4\,t_2}{15} + \frac{4\,t_3}{15} + \frac{4\,t_1\,t_2\,t_3}{3}$$

```
fprintf("\n");

omega_known = (t+1)*(t-0.69622)*(t+0.69622)*(t-0)*(t-1);
int_omega_known = int(omega_known, t, -1, 1);
fprintf("I_omega = ");
```

```
I_omega =
```

```
disp(vpa(int_omega_known, 6));
```

0.0

```
fprintf("\n");
```

Rezulta formula de cuadratura:

$$\int_{-1}^{1} f(t)\mathrm{dt} = 0.1 + 0.544444 \cdot f(0.654654) + 0.544444 \cdot f(-0.654654) + 0.711111 \cdot f(0) + 0.1$$

Problema 6

Șirul $x_n = 10^{-n^2}$ converge către 0 când $n \to \infty$. Arătați că converge superliniar. Ce se poate spune despre ordinul de convergență?

Pentru o metoda iterativa care converge la o valoare L, un sir (x_k) are ordinul de convergenta $q \geq 1$ si rata de convergenta μ daca:

$\lim_{k\to\infty} \frac{|x_{k+1} - L|}{|x_k - L|^q} = \mu$. Sirul converge superliniar daca $\mu \in (1,2)$.

Sirul nostru $x_n = 10^{-n^2}$ are ordinul de convergenta q daca $\exists\, C > 0$ a.i.:

$$x_{n+1} \approx C \cdot x_n^q \Leftrightarrow \ln(x_{n+1}) \approx \ln(C) + q \cdot \ln(x_n)$$

$$x_n = 10^{-n^2} \Leftrightarrow \ln(x_n) = -n^2 \cdot \ln(10)$$

$$x_{n+1} = 10^{-(n+1)^2} \Leftrightarrow \ln(x_{n+1}) = -(n+1)^2 \cdot \ln(10)$$

$$\frac{\ln(x_{n+1})}{\ln(x_n)} = \frac{(n+1)^2}{n^2} = \frac{n^2\left(1+\frac{1}{n}\right)^2}{n^2} = \left(1+\frac{1}{n}\right)^2 > 1$$

Astfel: $\ln(x_{n+1}) = \ln(1) + \left(1+\frac{1}{n}\right)^2 \ln(x_n) \Rightarrow x_{n+1} = 1 \cdot x_n^{\left(1+\frac{1}{n}\right)^2}$. Ordinul de convergenta este $q = \left(1+\frac{1}{n}\right)^2, n > 0$.

Rata de convergenta este > 1:

$$\lim_{n\to\infty} \frac{x_{n+1}}{x_n} = \lim_{n\to\infty} \frac{10^{-n^2\cdot\left(1+\frac{1}{n}\right)^2}}{10^{-n^2}} = \lim_{n\to\infty} 10^{-2n-1} = \lim_{n\to\infty} \frac{1}{10^{2n+1}} = \frac{1}{10^{\infty}} = 0$$

Limita cand n tinde la infinit este 0, de unde rezulta faptul ca rata de convergenta este superliniara.

```
function [root, iter] = newton_multiple_roots(f, df, x0, m, tol, max_iter)
    %% NEWTOK_MULTIPLE_ROOTS - Newton-Raphson method adapted for multiple roots
    %
    %  Inputs:
    %
    %  f        - function to approximate;
    %  df       - first derivative of the function to approximate;
    %  x0       - initial guess;
    %  tol      - acceptable error between iterations;
    %  max_iter - maximum number of iterations
    %
    %  Outputs
    %
    %  root     - the approximated root;
    %  iter     - the number of iterations took to approximate the root;
    %
    %  Error: if the number of iterations is exceeded.
    known_m = true;

    if nargin < 3
        x0 = 0;
    end
    if nargin < 4
        known_m = false;
```

```
    end
    if nargin < 5
        tol = 1e-6;
    end
    if nargin < 6
        max_iter = 10000;
    end

    % Keeping track of 3 iterations: x_{k-2}, x_{k-1}, x_k
    x = zeros(1, max_iter + 2);
    fx = zeros(1, max_iter + 2);
    x(1) = x0;
    fx(1) = f(x0);

    % First step, with classic Newton-Raphson
    x(2) = x(1) - fx(1) / df(x(1));
    fx(2) = f(x(2));

    % Second step, again with classic Newton-Raphson
    x(3) = x(2) - fx(2) / df(x(2));
    fx(3) = f(x(3));

    for i = 4:max_iter+1
        % Estimating the multiplicity of the roots, if it is not known
        if ~known_m
            num = log(abs(fx(i-1) / fx(i-2)));
            den = log(abs((x(i-1) - x(i-2)) / (x(i-2) - x(i-3))));
            if isnan(num) || isnan(den) || isinf(num) || isinf(den) || den == 0
                m = 1;
            else
                m = num / den;
            end
        end

        % Computing the iteration
        x(i) = x(i - 1) - m * fx(i-1) / df(x(i-1));
        fx(i) = f(x(i));

        % Checking if the approximation is acceptable
        if abs(x(i) - x(i-1)) < tol
            root = x(i);
            iter = i - 1;
            return;
        end
    end

    root = x(max_iter - 1);
    iter = max_iter - 1;
    warning('Approximation did not converge in %d maximum iterations', max_iter);
end
```