Problema 1

Presupunem că ecuația $f(x) = 0, f \in C^3(I)$ are o rădăcină simplă α si $f''(\alpha) = 0$ si $f'''(\alpha) \neq 0$. Arătați că în acest caz metoda secantei converge pătratic.

Știind că α este o rădăcină simplă a ecuației, condițiile devin sunt:

1. $f \in C^3(I)$;
2. $f(\alpha) = 0$;
3. $f'(\alpha) \neq 0$;
4. $f''(\alpha) = 0$;
5. $f'''(\alpha) \neq 0$;

Metoda secantei: $x_{k+1} = x_k - f(x_k) \cdot \dfrac{x_k - x_{k-1}}{f(x_k) - f(x_{k-1})}$. Dorim să demonstrăm faptul că această metodă converge pătratic, ținând cont de condițiile impuse. Altfel spus, vrem să arătăm că eroarea $R_k = x_k - \alpha$ satisface:

$R_{k+1} = C \cdot R_k \cdot R_{k+1}, C - \text{constanta}$.

Ne putem folosi de dezvoltarea Taylor in jurul lui α:

$$f(x) = f(\alpha) + \frac{x-\alpha}{1!} \cdot f'(\alpha) + \frac{(x-\alpha)^2}{2!} \cdot f''(\alpha) + \frac{(x-\alpha)^3}{3!} \cdot f'''(\alpha) + O((x-\alpha)^3)$$
$$f(x_k) = \frac{x_k - \alpha}{1!} \cdot f'(\alpha) + \frac{(x-\alpha)^3}{3!} \cdot f'''(\alpha) + O\left((x_k-\alpha)^3\right) = R_k \cdot f'(\alpha) + R_k^3 \cdot f'''(\alpha) + O\left(R_k^3\right)$$
$$f(x_{k-1}) = R_{k-1} \cdot f'(\alpha) + R_{k-1}^3 \cdot f'''(\alpha) + O\left(R_{k-1}^3\right)$$

Astfel, iteratia devine:

$$x_{k+1} = x_k - \left(R_k \cdot f'(\alpha) + R_k^3 \cdot f'''(\alpha) + O\left(R_k^3\right)\right) \cdot \frac{R_k - R_{k-1}}{\left(R_k \cdot f'(\alpha) + R_k^3 \cdot f'''(\alpha) + O\left(R_k^3\right)\right) - \left(R_{k-1} \cdot f'(\alpha) + R_{k-1}^3 \cdot f'''(\alpha) + O\left(R_k^3\right)\right)}$$
$$x_{k+1} = x_k - \frac{R_k^2 \cdot f'(\alpha) + R_k^4 \cdot f'''(\alpha) + O\left(R_k^4\right) - R_k R_{k-1} \cdot f'(\alpha) - R_k^3 R_{k-1} \cdot f'''(\alpha) - O\left(R_k^3 \cdot R_{k-1}\right)}{\left(R_k \cdot f'(\alpha) + R_k^3 \cdot f'''(\alpha) + O\left(R_k^3\right)\right) - \left(R_{k-1} \cdot f'(\alpha) + R_{k-1}^3 \cdot f'''(\alpha) + O\left(R_{k-1}^3\right)\right)}$$
$$x_{k+1} = x_k - \frac{f'(\alpha) \cdot R_k \cdot (R_k - R_{k-1}) + O\left(R_k^3 R_{k-1}\right)}{f'(\alpha)(R_k - R_{k-1})\left(1 + O\left(R_k^2\right) + O\left(R_{k-1}^2\right)\right)}$$

Putem obtine astfel eroarea:

$$x_{k+1} - \alpha = R_{k+1} = R_k - \frac{f'(\alpha) R_k (R_k - R_{k-1}) + O\left(R_k^3 R_{k-1}\right)}{f'(\alpha)(R_k - R_{k-1})\left(1 + O\left(R_k^2\right) + O\left(R_{k-1}^2\right)\right)} =$$
$$= R_k - R_k\left(1 + O\left(R_k^2 + O\left(R_{k-1}^2\right)\right)\right) = C \cdot R_k R_{k-1} + O(R_k R_{k-1})$$

Rezulta astfel ca:

$R_{k+1} = C \cdot R_k R_{k-1} + O(R_k R_{k-1})$, ceea ce inseamna ca metoda secantei converge patratic avand acea lista de conditii.

Problema 2

Consideram formula de cuadatura Gauss-Cebisev #1:

$$\int_{-1}^{1} \frac{f(t)}{\sqrt{1-t^2}} \mathrm{dt} = \sum_{k=1}^{n} \frac{\pi}{n} f(t_k) + R(f)$$

a) Demonstrati ca polinoamele Cebisev#1 verifica relatia de ortogonalitate discreta

$$(T_k, T_l) = \begin{cases} 0 & k \neq l \\ n+1 & k = l = 0 \\ \dfrac{n+1}{2} & k = l \neq 0 \end{cases}$$

unde produsul scalar este dat de

$$(u, v) = \sum_{i=0}^{n} u(x_k) v(x_k)$$

iar x_k sunt radacinile polinomului T_{n+1}.

Radacinile polinomului T_{n+1} sunt $x_i = \cos\left(\frac{2i+1}{2(n+1)}\pi\right), i = 0, 1, \ldots, n$

Ne putem folosi astfel de forma trigonometrica a polinoamelor Cebîșev#1, relatia de ortogonalitate devenind:

$(T_k, T_l) = \sum_{i=0}^{n} \cos(k \cdot x_i) \cdot \cos(l \cdot x_i)$.

Stim ca $\cos(a) \cdot \cos(b) = \frac{1}{2}[\cos(a+b) + \cos(a-b)]$, de unde rezulta ca:

$$\cos(k \cdot x_i) \cdot \cos(l \cdot x_i) = \frac{1}{2}[\cos((k+l) \cdot x_i) + \cos((k-l) \cdot x_i)]$$

$$\sum_{i=0}^{n} \cos\left(m \cdot \frac{2i+1}{2n+2}\pi\right) = \begin{cases} 0 & m \neq 0 \\ n+1 & m = 0 \end{cases}$$

$k \neq l \Rightarrow k + l \neq 0 \text{ si } k - l \neq 0 \Rightarrow (T_k, T_l) = 0$

$k = l = 0 \Rightarrow (T_0, T_0) = \sum_{i=0}^{n} n + 1$

$k = l \neq 0 \Rightarrow k - l = 0 \text{ si } k + l = 2k \neq 0 \Rightarrow (T_k, T_k) = \frac{n+1}{2}$

Rezulta astfel ca polinoamele Cebîșev#1 verifica relația de ortogonalitate discretă.

b) Se consideră aproximanta discretă în sensul celor mai mici pătrate $f(x) \approx \phi(x) = \frac{c_0}{2}T_0(x) + \sum_{k=0}^{n} c_k T_k$, relativa la sistemul ortogonal $(T_k)_{k=\overline{0,n}}$ si produsul scalar de la punctul a). Arătați că

$$c_j = \frac{2}{n+1}\sum_{k=0}^{n} f(x_k)T_j(x_k)$$

În sensul celor mai mici pătrate, pentru $j = 0, 1, \ldots, n$ avem condiția:

$$(f - \phi, T_j) = 0$$

$$j = 0: \sum_{i=0}^{n}\left(f(x_i) - \frac{c_0}{2}T_0(x_i), T_0(x_i)\right) = 0 \Rightarrow \sum_{i=0}^{n} f(x_i) = \frac{c_0}{2}(T_0, T_0) = \frac{c_0}{2}(n+1) \Rightarrow c_0 = \frac{2}{n+1}\sum_{i=0}^{n} f(x_i)$$

$$j \geq 1: \sum_{i=0}^{n}(f(x_i) - c_jT_j(x_i), T_j(x_i)) = 0 \Rightarrow \sum_{i=0}^{n} f(x_i)T_j(x_i) = c_j(T_j, T_j) = c_j\frac{n+1}{2} \Rightarrow c_j = \frac{2}{n+1}\sum_{i=0}^{n} f(x_i)T_j(x_i)$$

Rezulta astfel ca: $c_j = \frac{2}{n+1}\sum_{i=0}^{n} f(x_i)T_j(x_i),\ j = 0, 1, \ldots, n$

c) Calculați $\int_{-1}^{1} T_k(x)\mathrm{dx}$

$$T_k = \cos(k \cdot \mathrm{arccosx}),\ x \in [-1, 1]$$

Prin schimbare de variabila:

$$x = \cos(\tau),\ \tau = \arccos(x),\ \tau \in [0, \pi]$$
$$\mathrm{dx} = -\sin(\tau)d\tau$$

Rezulta astfel:

$$\int_{-1}^{1} T_k(x)\mathrm{dx} = \int_{0}^{\pi}\cos(k\tau)(-\sin(\tau))d\tau = \int_{0}^{\pi}\cos(k\tau)\sin(\tau)d\tau$$

$$\cos(k\tau)\sin(\tau) = \frac{1}{2}[\sin((k+1)\tau) - \sin((k-1)\tau)]$$

$$\Rightarrow \int_{0}^{\pi}\cos(k\tau)\sin(\tau)d\tau = \frac{1}{2}\int_{0}^{\pi}\sin((k+1)\tau)d\tau - \frac{1}{2}\int_{0}^{\pi}\sin((k-1)\tau)d\tau$$

Știm de altfel că

$$\int_0^{\pi} \sin(m\tau)d\tau = \left[-\frac{1}{m}\cos(m\tau)\right]|_0^{\pi} = -\frac{1}{m}(\cos(m\tau) - \cos(0)) = -\frac{1}{m}((-1)^m - 1) = \begin{cases} 0 & m - \text{par} \\ \frac{2}{m} & m - \text{impar} \end{cases}$$

$$k - \text{par} \Rightarrow k \pm 1 - \text{impar} \Rightarrow \int_0^{\pi} \sin((k+1)\tau)d\tau = \frac{2}{k+1}, \int_0^{\pi} \sin((k-1)\tau)d\tau = \frac{2}{k-1}$$

$$k - \text{impar} \Rightarrow k \pm 1 - \text{par} \Rightarrow \int_0^{\pi} \sin((k+1)\tau)d\tau = \int_0^{\pi} \sin((k-1)\tau)d\tau = 0$$

$$\Rightarrow \int_{-1}^{1} T_k(x)\mathrm{dx} = \begin{cases} \frac{2}{1-k^2} & k - \text{par} \\ 0 & k \text{ impar} \end{cases}$$

d) Integrând termen cu termen aproximația de la punctul (b) și folosind rezultatul de la (c) stabiliți formula de cuadratură

$$\int_{-1}^{1} f(x)\mathrm{dx} \approx \sum_{k=0,\text{par}} \frac{2c_k}{1-k^2}$$

Folosind proprietatea de liniaritate a integralei:

$$\int_{-1}^{1} f(x)\mathrm{dx} \approx \int_{-1}^{1} \phi(x)\mathrm{dx} = \int_{-1}^{1} \frac{c_0}{2} T_0(x)\mathrm{dx} + \sum_{k=1}^{n} c_k \cdot \int_{-1}^{1} T_k(x)\mathrm{dx}$$

Stim ca $\int_{-1}^{1} T_k(x)\mathrm{dx} = \begin{cases} \frac{2}{1-k^2} & k - \text{par} \\ 0 & k - \text{impar} \end{cases}$, rezulta astfel:

$$\int_{-1}^{1} \phi(x)\mathrm{dx} = \frac{c_0}{2} \cdot \frac{2}{1-0^2} + \sum_{k=2,\text{par}}^{n} c_k \cdot \frac{2}{1-k^2} = c_0 + \sum_{k=2,\text{par}}^{n} \frac{2c_k}{1-k^2} = \sum_{k=0,\text{par}}^{n} \frac{2c_k}{1-k^2}$$

Rezulta astfel ca: $\int_{-1}^{1} f(x)\mathrm{dx} \approx \sum_{k=0,\text{par}} \frac{2c_k}{1-k^2}$

e) Implementați metoda de la punctul (d) în MATLAB (pentru o precizie dată) și calculați $\int_{-1}^{1} e^x \cos(x^2)\mathrm{dx}$.

```
f = @(x) exp(x) .* sin (x .^ 2);
result = cheb_quad(f);

fprintf("Result: %.16e\n", result);
```

```
Result: 2.0681711476113787e+00
```

```
function result = cheb_quad(f, tol, n_nodes)
    %% CHEB_QUAD - computes the quadrature using the above formula
```

```
    %
    %   Inputs:
    %
    %   f           - function handle;
    %   tol         - acceptable tolerance;
    %   n_nodes     - the number of nodes
    %
    %   Outputs:
    %
    %   result  - computed quadrature
    %   iter    - the number of iterations until convergence is achieved
    %
    if nargin < 2
        tol = 1e-6;
    end
    if nargin < 3
        n_nodes = 10;
    end

    prev = 0;
    while true
        x = cos((2 * (0:n_nodes) + 1) * pi / (2*n_nodes + 2));
        T = zeros(n_nodes+1, n_nodes+1);
        T(:, 1) = 1;
        T(:, 2) = x';

        for k = 2:n_nodes
            T(:, k+1) = 2 * x' .* T(:, k) - T(:, k-1);
        end

        fx = f(x)';
        cj = zeros(n_nodes+1, 1);
        for j = 0:n_nodes
            cj(j+1) = (2 / (n_nodes+1)) * sum(fx .* T(:, j+1));
        end

        result = 0;
        for k = 0:2:n_nodes
            result = result + (2 * cj(k+1)) / (1 - k^2);
        end

        if abs(result - prev) < tol
            return;
        end

        prev = result;
        n_nodes = n_nodes * 2;
    end
end
```