Problema 3

(a) Determinați o metodă pentru rezolvarea ecuației neliniare folosind interpolarea inversă Lagrange de gradul II, în forma Newton.

O sa rezolvam ecuatia $f(x) = 0$ folosindu-ne de 3 iteratii x_{n-2}, x_{n-1}, x_n.

$$y_k = f(x_k), k = n-2, n-1, n \Rightarrow P(y) = \sum_{k=n-2}^{n} x_k l_k(y)$$

$$\text{unde } l_k(y) = \prod_{j=n-2, j\neq k}^{n} \frac{y - y_j}{y_k - y_j}; \text{aproximam apoi solutia prin } x_{n+1} = P(0)$$

Folosim forma Newton a lui P(y) si de diferente divizate:

$$P(y) = x_n + (y - y_n) \cdot \frac{x_n - x_{n-1}}{y_n - y_{n-1}} + (y - y_n)(y - y_{n-1}) \cdot \frac{\frac{x_n - x_{n-1}}{y_n - y_{n-1}} - \frac{x_{n-1} - x_{n-2}}{y_{n-1} - y_{n-2}}}{y_n - y_{n-2}}$$

$$x_{n+1} = P(0) = x_n - y_n \cdot \frac{x_n - x_{n-1}}{y_n - y_{n-1}} + y_n y_{n-1} \cdot \frac{\frac{x_n - x_{n-1}}{y_n - y_{n-1}} - \frac{x_{n-1} - x_{n-2}}{y_{n-1} - y_{n-2}}}{y_n - y_{n-2}}$$

(b) Determinați ordinul de convergență al metodei.

Presupunând că iterațiile se apropie de rădăcină, eroarea $R_k = x_k - x_{k-1}$ urmează relația de recurență:

$$R_{k+1} \approx C \cdot R_k R_{k-1} R_{k-2}, C \text{ constantă}$$

$$R_{k+1} \approx C \cdot R_k^p \Leftrightarrow p^{n+1} \approx C \cdot p^n p^{n-1} p^{n-2} \Rightarrow p^3 = p^2 + p + 1 \Leftrightarrow p^3 - p^2 - p - 1 = 0$$

Ordinul de convergenta este astfel: $p \approx 1.83928$

```
syms p

eq = p^3 - p^2 - p - 1 == 0;
sols = solve(eq, p);

disp(sols);
```

$$\begin{pmatrix} \mathrm{root}(z^3 - z^2 - z - 1, z, 1) \\ \mathrm{root}(z^3 - z^2 - z - 1, z, 2) \\ \mathrm{root}(z^3 - z^2 - z - 1, z, 3) \end{pmatrix}$$

```
sols_computed = vpa(sols, 10);
real_sols = sols_computed(imag(sols_computed) == 0);

fprintf("Ordinul de convergenta: %.16e\n", double(real_sols));
```

```
Ordinul de convergenta: 1.8392867552141612e+00
```

Problema 4

Considerăm formula de cuadratură Gauss-Cebîșev-Lobatto de speța I:

$$\int_{-1}^{1} \frac{1}{\sqrt{1-t^2}} \cdot f(t)\mathrm{dt} = \frac{\pi}{2(n+1)}[f(-1)+f(1)] + \frac{\pi}{n+1}\sum_{k=1}^{n} f\left(\cos\left(\frac{n+1-k}{n+1}\right)\right) + R(f)$$

(a) Demonstrati ca polinoamele Cebîșev de speța I verifică relația de ortogonalitate discretă

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

unde produsul scalar este dat de

$$(u, v) = \frac{1}{2}u(x_0)v(x_0) + \sum_{i=1}^{n-1} u(x_k)v(x_k) + \frac{1}{2}u(x_n)v(x_n)$$

iar x_k **sunt extremele polinomului** T_n, $x_k = \cos\frac{k\pi}{n}, k = 0,\ldots,n$

$$(u, v) = \sum_{k=0}^{n} \omega_k u(x_k)v(x_k), \omega_0 = \omega_n = \frac{1}{2}, \text{iar } \omega_k = 1, 1 \leq k \leq n-1$$

$$\Rightarrow (T_k, T_l) = \sum_{j=0}^{n} \omega_j T_k(x_j)T_l(x_j) = \sum_{j=0}^{n} \omega_j \cos\left(k\frac{j\pi}{n}\right)\cos\left(l\frac{j\pi}{n}\right) =$$

$$= \frac{1}{2}\sum_{j=0}^{n} \omega_j \left[\cos\left(\frac{(k+l)\pi}{n}\right) + \cos\left(\frac{(k+l)\pi}{n}\right)\right]$$

$$k \neq l : k+l \neq 0 \text{ si } k-l \neq 0; \sum_{j=0}^{n} \omega_j \cos\left(\frac{m \cdot j \cdot \pi}{n}\right) = 0, m \in N^*, m \leq 2n-1$$

$$k = l = 0 : T_0(x_j) = 1 \Rightarrow (T_0, T_0) = \sum_{j=0}^{n} \omega_j = \frac{1}{2} + (n-1) + \frac{1}{2} = n$$

$$k = l \neq 0 : T_k(x_j)^2 = \cos^2\left(\frac{kj\pi}{n}\right) = \frac{1}{2}\left[1 + \cos\left(\frac{2\mathrm{kj}\pi}{n}\right)\right] \Rightarrow (T_k, T_k) = \sum_{j=0}^{n} \omega_j T_k(x_j)^2 = \frac{1}{2}\sum_{j=0}^{n} \omega_j = \frac{n}{2}$$

Rezulta astfel ca: $(T_k, T_l) = \begin{cases} 0 & k \neq l \\ n & k = l = 0 \\ \frac{n}{2} & k = l \neq 0 \end{cases}$

b) Se consideră aproximanta discretă în sensul celor mai mici pătrate

$f(x) \approx \phi(x) = \frac{c_0}{2}T_0(x) + \sum_{k=0}^{n} c_k T_k(x)$, **relativă la sistemul ortogonal** $(T_k)_{k=\overline{0,n}}$ **și produsul scalar de la punctul (a). Arătați că:**

$c_j = \dfrac{2}{n}\sum_{k=0}^{n} f(x_k)T_j(x_k)$.

In sensul celor mai mici patrate, pentru j = 0,...,n, avem conditia: $(f - \phi, T_j) = 0$

$$j = 0 : \sum_{k=0}^{n}\left(f(x_k) - \frac{c_0}{2}T_0(x_k), T_0(x_k)\right) = 0 \Rightarrow \sum_{k=0}^{n} f(x_k) = \frac{c_0}{2}(T_0, T_0) = \frac{c_0}{2}n \Rightarrow c_0 = \frac{2}{n}\sum_{k=0}^{n} f(x_k)$$

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

Rezulta astfel ca: $c_j = \dfrac{2}{n}\sum_{k=0}^{n} f(x_k)T_j(x_k)$

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

$T_k = \cos(k \cdot \arccos(x)), x \in [-1, 1]$

Prin schimbare de variabila:

$x = \cos(\theta), \mathrm{dx} = -\sin(\theta)d\theta$. Rezulta astfel:

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

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

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

$$k - \text{par} \Rightarrow k \pm 1 - \text{impar} \Rightarrow \int_{0}^{\pi}\sin((k+1)\tau)d\tau = \frac{2}{k+1}, \int_{0}^{\pi}\sin((k-1)\tau)d\tau = \frac{2}{k-1}$$

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

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

d) Integrand termen cu termen aproximatia de la punctul (b) si folosind rezultatul de la (c) stabiliti formula de cuadratura

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

Folosind proprietatea de liniaritate a integralei:

$$\int_{-1}^{1} f(x)\mathrm{dx} \approx \int_{-1}^{1}\phi(x)\mathrm{dx} = \int_{-1}^{1}\frac{c_0}{2}T_0(x)\mathrm{dx} + \sum_{k=0}^{n} c_k\int_{-1}^{1} T_k(x)\mathrm{dx}$$

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

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

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

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

```
Result: 4.1092920035319285e+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
        n = n_nodes;
        x = cos((0:n) * pi / n);
        fx = f(x);

        T = zeros(n+1, n+1);
        T(:, 1) = 1;
        if n >= 1
            T(:, 2) = x';
        end
        for k = 2:n
            T(:, k+1) = 2 * x' .* T(:, k) - T(:, k-1);
        end

        cj = zeros(n+1, 1);
        w = ones(n+1, 1);
```

```
        w(1) = 0.5; w(end) = 0.5;
        for j = 0:n
            prod = fx(:) .* T(:, j+1);
            cj(j+1) = (2 / n) * sum(w .* prod);
        end


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

        if abs(result - prev) < tol
            break;
        end
        prev = result;
        n_nodes = n_nodes * 2;
    end
end
```