Calcul Numeric – Tema #3

Ex. 1 Fie matricea simetrică $A = \begin{pmatrix} a_1 & a_2 & a_3 & \dots & a_n \\ & a_1 & a_2 & \dots & a_{n-1} \\ & & a_1 & \dots & a_{n-2} \\ SYM & & & \ddots & \\ & & & & a_1 \end{pmatrix}$

1. Să se definească în Python vectorul a şi matricea simetrică A; (Vezi Tema #2)
2. Să se verifice în Python că matricea A este pozitiv definită (Se va folosi criteriul Sylvester);
3. Să se construiască în Python procedura **GradConjugat**(A, b, ε) conform algoritmului Gradientului Conjugat. Procedura **GradConjugat** returnează soluţia sistemului $Ax = b$, A fiind o matrice simetrică şi pozitiv definită;
4. Să se afle soluţia sistemului $Ax = b$ şi să se afişeze soluţia. Condiţia de oprire este $\| \nabla f(x^k) \| < \varepsilon$ cu $\varepsilon = 10^{-10}$ sau algoritmul se opreşte după n paşi, unde n este dimensiunea matricei, mai exact după ce s-a calculat iteraţia $x^{(n)}$. Să se verifice soluţia obţinută.

V0 $n = 5, b_i = i^2, i = \overline{1,n}, a = (2n, 2n-2, ..., 2)^T$

V1 $n = 8, b_i = i^3, i = \overline{1,n}, a = (3n, 3n-3, ..., 3)^T$

V2 $n = 10, b_i = i^3 + 2, i = \overline{1,n}, a = (n, n-1, ..., 1)^T$

V3 $n = 5, b_i = i^4, i = \overline{1,n}, a = (2^n, 2^{n-1}, ..., 2^1)^T$

V4 $n = 6, b_i = i^4, i = \overline{1,n}, a = (4^n, 4^{n-1}, ..., 4^1)^T$

V5 $n = 6, b_i = i^2, i = \overline{1,n}, a = (3^n, 3^{n-1}, ..., 3^1)^T$

V6 $n = 5, b_i = i^2, i = \overline{1,n}, a = (2n, 2n-2, ..., 2)^T$

V7 $n = 8, b_i = i^3, i = \overline{1,n}, a = (3n, 3n-3, ..., 3)^T$

V8 $n = 10, b_i = i^3 + 2, i = \overline{1,n}, a = (n, n-1, ..., 1)^T$

V9 $n = 5, b_i = i^4, i = \overline{1,n}, a = (2^n, 2^{n-1}, ..., 2^1)^T$

V10 $n = 6, b_i = i^4, i = \overline{1,n}, a = (4^n, 4^{n-1}, ..., 4^1)^T$

V11 $n = 6, b_i = i^2, i = \overline{1,n}, a = (3^n, 3^{n-1}, ..., 3^1)^T$

V12 $n = 5, b_i = i^2, i = \overline{1,n}, a = (2n, 2n-2, ..., 2)^T$

V13 $n = 8, b_i = i^3, i = \overline{1,n}, a = (3n, 3n-3, ..., 3)^T$

V14 $n = 10, b_i = i^3 + 2, i = \overline{1,n}, a = (n, n-1, ..., 1)^T$

V15 $n = 5, b_i = i^4, i = \overline{1,n}, a = (2^n, 2^{n-1}, ..., 2^1)^T$

V16 $n = 6, b_i = i^4, i = \overline{1,n}, a = (4^n, 4^{n-1}, ..., 4^1)^T$

V17 $n = 8, b_i = i^3, i = \overline{1,n}, a = (3n, 3n-3, ..., 3)^T$

Ex. 2 Fie forma pătratică $f(x,y) = \frac{1}{2} < A\begin{pmatrix} x \\ y \end{pmatrix}, \begin{pmatrix} x \\ y \end{pmatrix} > - < \begin{pmatrix} b_1 \\ b_2 \end{pmatrix}, \begin{pmatrix} x \\ y \end{pmatrix} >$. Să se scrie matricea A şi vectorul b precizându-se natura matricei A şi natura punctului de extrem.

V0 $f(x,y) = 4x^2 + 7y^2 - 0.8xy - 35.2x - 47.6y$

V1 $f(x,y) = x^2 + xy - 2x + y^2 - 3y$

V2 $f(x,y) = \frac{3}{2}x^2 + 2xy - x + \frac{3}{2}y^2 - 3y$

V3 $f(x,y) = 2x^2 + xy - x + 2y^2 - 2y$

V4 $f(x,y) = 2x^2 + 2xy + x + 2y^2 - 2y$

V5 $f(x,y) = 2x^2 + 3xy + x + 2y^2 - y$

V6 $f(x,y) = \frac{5}{2}x^2 + xy + x + \frac{5}{2}y^2 - y$

V7 $f(x,y) = \frac{5}{2}x^2 + 2xy + 3x + \frac{5}{2}y^2 - 3y$

V8 $f(x,y) = \frac{5}{2}x^2 + 3xy + 3x + \frac{5}{2}y^2 - 2y$

V9 $f(x,y) = \frac{5}{2}x^2 + 4xy + 3x + \frac{5}{2}y^2 - 3y$

V10 $f(x,y) = 3x^2 + xy + x + 3y^2 - y$

V11 $f(x,y) = 3x^2 + 2xy + 2x + 3y^2 - 2y$

V12 $f(x,y) = 3x^2 + 3xy + 2x + 3y^2 - 4y$

V13 $f(x,y) = 3x^2 + 4xy + 3x + 3y^2 - y$

V14 $f(x,y) = 3x^2 + 5xy + 3x + 3y^2 - y$

V15 $f(x,y) = \frac{7}{2}x^2 + xy + 3x + \frac{7}{2}y^2 - y$

V16 $f(x,y) = \frac{7}{2}x^2 + 2xy + 3x + \frac{7}{2}y^2 - y$

V17 $f(x,y) = \frac{7}{2}x^2 + 3xy + 3x + \frac{7}{2}y^2 - 2y$

Ex. 3 Pentru datele de la Ex. 2 şi domeniul $(x,y) \in [a,b] \times [c,d]$ să se implementeze în Python următoarele cerinţe:

- Să se construiască suprafaţa $z = f(x,y)$ pe domeniul dat;
- Să se afle conform Gradientului Conjugat $x^{(1)}, x^{(2)}$, dacă $x^{(0)} = (a,c)^T$.
- Să se reprezinte pe suprafaţă punctul de minim.
- Să se construiască într-o altă figură curbele de nivel care trec prin punctele $x^{(k)}, k = 0,1,2$, precum şi traseul deplasărilor până la punctul de minim.

V0 $[a,b] \times [c,d] = [-4,16] \times [-4,16]$

V1 $[a,b] \times [c,d] = [-3,4] \times [-2,5]$

V2 $[a,b] \times [c,d] = [-4,3] \times [-2,5]$

V3 $[a,b] \times [c,d] = [-3,4] \times [-3,4]$

V4 $[a,b] \times [c,d] = [-4,3] \times [-3,4]$

V5 $[a,b] \times [c,d] = [-4,3] \times [-3,4]$

V6 $[a, b] \times [c, d] = [-4, 3] \times [-3, 4]$

V7 $[a, b] \times [c, d] = [-4, 3] \times [-3, 4]$

V8 $[a, b] \times [c, d] = [-5, 2] \times [-2, 5]$

V9 $[a, b] \times [c, d] = [-6, 0] \times [0, 6]$

V10 $[a, b] \times [c, d] = [-4, 3] \times [-3, 4]$

V11 $[a, b] \times [c, d] = [-4, 3] \times [-3, 4]$

V12 $[a, b] \times [c, d] = [-4, 3] \times [-2, 5]$

V13 $[a, b] \times [c, d] = [-5, 2] \times [-3, 4]$

V14 $[a, b] \times [c, d] = [-6, 1] \times [-2, 5]$

V15 $[a, b] \times [c, d] = [-4, 3] \times [-3, 4]$

V16 $[a, b] \times [c, d] = [-4, 3] \times [-3, 4]$

V17 $[a, b] \times [c, d] = [-4, 3] \times [-3, 4]$

Ex. 4 a) Să se construiască în Python procedura **MetNewton**(X, Y, x) conform metodei Newton. Vectorii X, Y reprezintă nodurile de interpolare, respectiv valorile funcţiei f în nodurile de interpolare. Procedura **MetNewton**(X, Y, x) returnează valoarea polinomului de interpolare $y = P_n(x)$ conform metodei Newton.

b) Să se construiască în Python procedura **MetNDD**(X, Y, x) conform metodei Newton DD. Vectorii X, Y reprezintă nodurile de interpolare, respectiv valorile funcţiei f în nodurile de interpolare. Procedura **MetNDD**(X, Y, x) returnează valoarea polinomului de interpolare $y = P_n(x)$ conform metodei Newton cu DD.

c) Să se construiască în Python în aceeaşi figură, graficele funcţiei f pe intervalul $[a, b]$, punctele $(X_i, Y_i), i = \overline{1, n+1}$ şi polinomul $P_n(x)$ obţinut în baza metodei Newton. Reprezentaţi grafic într-o altă figură eroarea $|e_t(x)| = |f(x) - P_n(x)|$.

d) Să se construiască în Python în aceeaşi figură, graficele funcţiei f pe intervalul $[a, b]$, punctele $(X_i, Y_i), i = \overline{1, n+1}$ şi polinomul $P_n(x)$ obţinut în baza metodei Newton cu DD. Reprezentaţi grafic într-o altă figură eroarea $|e_t(x)| = |f(x) - P_n(x)|$.

Obs.: În cazul în care nu este dată funcţia nu se va reprezenta grafic şi nu se va calcula eroarea.

V0 $f(x) = sin3x, [a, b] = [1;\ 2,2], n = 4$ (se consideră discretizare echidistantă)

V1 $X = (2,\ 3,\ 5,\ 8,\ 12), Y = (10,\ 15,\ 25,\ 40,\ 60)$

V2 $X = (-2,\ 1,\ 3,\ 7), Y = (5,\ 7,\ 11,\ 34)$

V3 $f(x) = e^x, [a, b] = [-1, 1], n = 4$ (se consideră discretizare echidistantă)

V4 $X = (0.4,\ 0.5\ 0.7\ 0.8), f(x) = lnx$

V5 $X = (0,\ 1,\ 3,\ 6\ ), Y = (18,\ 10,\ -18, 90)$

V6 $f(x) = e^{3x}, [a, b] = [-1, 1], n = 5$ (se consideră discretizare Chebyshev)

V7 $f(x) = sin(2x), [a, b] = [-\pi, \pi], n = 7$ (se consideră discretizare Chebyshev)

V10 $f(x) = log_2(x), [a, b] = [-\pi, \pi], n = 7$ (se consideră discretizarea Chebyshev)

V11 $f(x) = cos(2x), [a, b] = [-2\pi, 2\pi], n = 10$ (se consideră discretizare Chebyshev)

V12 $f(x) = \dfrac{1}{1 + 16x^2}, [a, b] = [-2, 2], n = 10$ (se consideră discretizare Chebyshev)

V13 $X = (0.4,\ 0.5\ 0.7\ 0.8), f(x) = e^x$

V14 $X = (0,\ 1,\ 3,\ 6\ ), Y = (18,\ 10,\ -18, 90)$

V15 $f(x) = sin3x, [a, b] = [1;\ 2, 2], n = 4$ (se consideră discretizare echidistantă)

V16 $f(x) = e^{3x}, [a, b] = [-1, 1], n = 5$ (se consideră discretizare Chebyshev)

V17 $f(x) = cos(2x), [a, b] = [-2\pi, 2\pi], n = 10$ (se consideră discretizare Chebyshev)