4.

| **def** **calculeaza\_polinom**(x, v):  rez = 0  **for** i **in** range(len(v) - 1, -1, -1):  rez += v[i]  **if** i != 0:  rez \*= x   **return** rez  print(calculeaza\_polinom(3, [1, 2, 3, 4])) |
| --- |

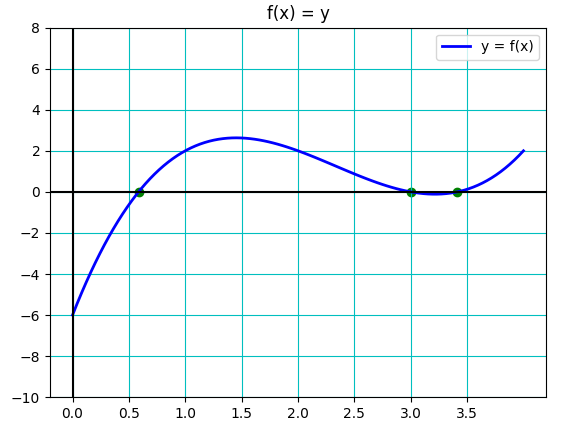
Consola

| 142 |
| --- |

6.

| **import** numpy **as** np **import** matplotlib.pyplot **as** plt **from** prettytable **import** PrettyTable **import** math  **def** **eroare**(x1, x2):  **return** abs((x2 - x1) / x2)  **def** **functie**(x):  **return** x\*x\*x - 7 \* x \* x + 14 \* x - 6  **def** **radacina**(func, a, b, tabel, e = 0.001):  cnt = 0  x\_vechi = -1    **while** **True**:  x = (a+b) / 2   y = func(x)  er = ""  **if** cnt != 0:  er = eroare(x\_vechi, x)  logaritm = (float)((b - a) / (2 \*\* (cnt)))   tabel.add\_row([cnt + 1, x, y, e, logaritm])   **if** cnt != 0 **and** er < e:  **return** {'x':x, 'it':cnt}   **if** func(a) \* func(x) < 0:  b = x  **else**:  a = x    cnt += 1  x\_vechi = x    puncte = [] tabele = []  st = 0 fin = 1 tabel = PrettyTable(['k', 'x\_k', 'f(x\_k)', 'rel err', '(b-a)/2^(k-1)']) pt = radacina(functie, st, fin, tabel, 10\*\*-6) puncte.append(pt['x']) tabele.append(tabel)  st = 2.5 fin = 3.05 tabel = PrettyTable(['k', 'x\_k', 'f(x\_k)', 'rel err', '(b-a)/2^(k-1)']) pt = radacina(functie, st, fin, tabel, 10\*\*-6) puncte.append(pt['x']) tabele.append(tabel)  st = 3.05 fin = 4 tabel = PrettyTable(['k', 'x\_k', 'f(x\_k)', 'rel err', '(b-a)/2^(k-1)']) pt = radacina(functie, st, fin, tabel, 0.000001) puncte.append(pt['x']) tabele.append(tabel)   print("Radacini:") print(puncte)  **for** i **in** range(3):  print(f"x = {puncte[i]}:")  print(tabele[i])  fig = plt.figure(1) ax = plt.axes() x = np.linspace(0, 4, 100) y = functie(x) ax.plot(x, y, linestyle='-', lw = 2, color = 'b', label = 'y = f(x)') ax.scatter(puncte, np.zeros(3), color="g") ax.grid(**True**, color = 'c') plt.xticks(np.arange(0,4,0.5)) plt.yticks(np.arange(-10,10,2)) ax.legend(loc='best') plt.title('f(x) = y') ax.axhline(y=0,color='k') ax.axvline(x=0,color='k') plt.show() |
| --- |

Grafic

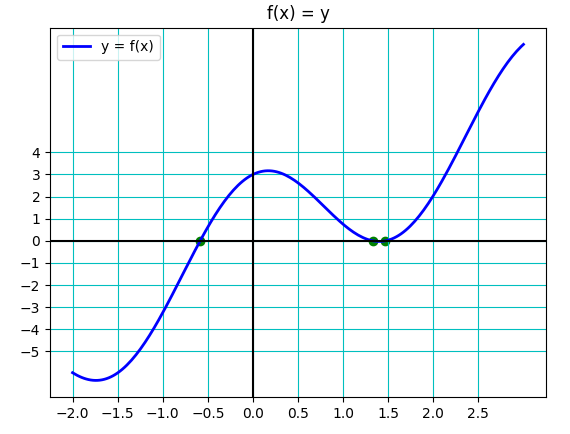


Output:

| Radacini: [0.5857863426208496, 3.0000005722045895, 3.4142136573791504] x = 0.5857863426208496: +----+--------------------+-------------------------+---------+------------------------+ | k | x\_k | f(x\_k) | rel err | (b-a)/2^(k-1) | +----+--------------------+-------------------------+---------+------------------------+ | 1 | 0.5 | -0.625 | 1e-06 | 1.0 | | 2 | 0.75 | 0.984375 | 1e-06 | 0.25 | | 3 | 0.625 | 0.259765625 | 1e-06 | 0.0625 | | 4 | 0.5625 | -0.161865234375 | 1e-06 | 0.015625 | | 5 | 0.59375 | 0.054046630859375 | 1e-06 | 0.00390625 | | 6 | 0.578125 | -0.052623748779296875 | 1e-06 | 0.0009765625 | | 7 | 0.5859375 | 0.0010313987731933594 | 1e-06 | 0.000244140625 | | 8 | 0.58203125 | -0.025716006755828857 | 1e-06 | 6.103515625e-05 | | 9 | 0.583984375 | -0.012322284281253815 | 1e-06 | 1.52587890625e-05 | | 10 | 0.5849609375 | -0.005640440620481968 | 1e-06 | 3.814697265625e-06 | | 11 | 0.58544921875 | -0.002303270739503205 | 1e-06 | 9.5367431640625e-07 | | 12 | 0.585693359375 | -0.0006356234807753935 | 1e-06 | 2.384185791015625e-07 | | 13 | 0.5858154296875 | 0.000197965766346897 | 1e-06 | 5.960464477539063e-08 | | 14 | 0.58575439453125 | -0.00021880932649764873 | 1e-06 | 1.4901161193847656e-08 | | 15 | 0.585784912109375 | -1.0416897481491105e-05 | 1e-06 | 3.725290298461914e-09 | | 16 | 0.5858001708984375 | 9.3775655070516e-05 | 1e-06 | 9.313225746154785e-10 | | 17 | 0.5857925415039062 | 4.1679683954853886e-05 | 1e-06 | 2.3283064365386963e-10 | | 18 | 0.5857887268066406 | 1.563146952676675e-05 | 1e-06 | 5.820766091346741e-11 | | 19 | 0.5857868194580078 | 2.607305095381207e-06 | 1e-06 | 1.4551915228366852e-11 | | 20 | 0.5857858657836914 | -3.904791425313192e-06 | 1e-06 | 3.637978807091713e-12 | | 21 | 0.5857863426208496 | -6.487419721423748e-07 | 1e-06 | 9.094947017729282e-13 | +----+--------------------+-------------------------+---------+------------------------+ x = 3.0000005722045895: +----+--------------------+-------------------------+---------+------------------------+ | k | x\_k | f(x\_k) | rel err | (b-a)/2^(k-1) | +----+--------------------+-------------------------+---------+------------------------+ | 1 | 2.775 | 0.3148593749999975 | 1e-06 | 0.5499999999999998 | | 2 | 2.9124999999999996 | 0.10214257812500449 | 1e-06 | 0.13749999999999996 | | 3 | 2.9812499999999997 | 0.01944653320312284 | 1e-06 | 0.034375000000000044 | | 4 | 3.015625 | -0.015132904052734375 | 1e-06 | 0.008593750000000011 | | 5 | 2.9984374999999996 | 0.0015673789978052355 | 1e-06 | 0.0021484375000000167 | | 6 | 3.00703125 | -0.006932025432583089 | 1e-06 | 0.0005371093750000111 | | 7 | 3.0027343749999997 | -0.0027194009423254784 | 1e-06 | 0.00013427734375000278 | | 8 | 3.0005859374999995 | -0.0005852506533301494 | 1e-06 | 3.3569335937500694e-05 | | 9 | 2.9995117187499996 | 0.0004887579707499867 | 1e-06 | 8.392333984374306e-06 | | 10 | 3.0000488281249993 | -4.882335651501535e-05 | 1e-06 | 2.0980834960935765e-06 | | 11 | 2.9997802734374996 | 0.00021982311142210165 | 1e-06 | 5.245208740231773e-07 | | 12 | 2.9999145507812495 | 8.546382126439767e-05 | 1e-06 | 1.311302185056859e-07 | | 13 | 2.9999816894531244 | 1.8311217417021908e-05 | 1e-06 | 3.2782554626421476e-08 | | 14 | 3.0000152587890616 | -1.5258323394107265e-05 | 1e-06 | 8.195638656605369e-09 | | 15 | 2.999998474121093 | 1.5258835617260047e-06 | 1e-06 | 2.0489096641377897e-09 | | 16 | 3.0000068664550774 | -6.866360784840708e-06 | 1e-06 | 5.122274160276712e-10 | | 17 | 3.0000026702880853 | -2.670273815397195e-06 | 1e-06 | 1.280568540069178e-10 | | 18 | 3.0000005722045895 | -5.722039304600912e-07 | 1e-06 | 3.201421350172945e-11 | +----+--------------------+-------------------------+---------+------------------------+ x = 3.4142136573791504: +----+--------------------+-------------------------+---------+------------------------+ | k | x\_k | f(x\_k) | rel err | (b-a)/2^(k-1) | +----+--------------------+-------------------------+---------+------------------------+ | 1 | 3.525 | 0.17095312499999693 | 1e-06 | 0.9500000000000002 | | 2 | 3.2874999999999996 | -0.09842382812500716 | 1e-06 | 0.23750000000000004 | | 3 | 3.40625 | -0.009124755859375 | 1e-06 | 0.05937500000000007 | | 4 | 3.465625 | 0.06893887329101744 | 1e-06 | 0.014843749999999989 | | 5 | 3.4359375 | 0.026991725921632792 | 1e-06 | 0.003710937500000011 | | 6 | 3.42109375 | 0.00821446371077883 | 1e-06 | 0.0009277343750000028 | | 7 | 3.413671875 | -0.0006336749196051983 | 1e-06 | 0.00023193359374999722 | | 8 | 3.4173828124999996 | 0.0037456088736647075 | 1e-06 | 5.7983398437499306e-05 | | 9 | 3.41552734375 | 0.0015447897603735328 | 1e-06 | 1.449584960937396e-05 | | 10 | 3.414599609375 | 0.0004527655117030349 | 1e-06 | 3.6239624023439235e-06 | | 11 | 3.4141357421875 | -9.115238169243867e-05 | 1e-06 | 9.059906005861977e-07 | | 12 | 3.41436767578125 | 0.00018063210814034392 | 1e-06 | 2.2649765014665785e-07 | | 13 | 3.414251708984375 | 4.469625368130892e-05 | 1e-06 | 5.662441253666446e-08 | | 14 | 3.414193725585937 | -2.3238965795258082e-05 | 1e-06 | 1.4156103134166115e-08 | | 15 | 3.4142227172851562 | 1.0725918421883307e-05 | 1e-06 | 3.5390257835550814e-09 | | 16 | 3.4142082214355467 | -6.257205065196558e-06 | 1e-06 | 8.847564458955466e-10 | | 17 | 3.4142154693603515 | 2.2341863399333306e-06 | 1e-06 | 2.2118911147388665e-10 | | 18 | 3.414211845397949 | -2.0115519490104816e-06 | 1e-06 | 5.529727786847166e-11 | | 19 | 3.4142136573791504 | 1.1130654797852912e-07 | 1e-06 | 1.3824319467964949e-11 | +----+--------------------+-------------------------+---------+------------------------+ |
| --- |

7

| **import** math **import** numpy **as** np **import** matplotlib.pyplot **as** plt  **def** **eroare**(x1, x2):  **if** x2 == 0:  **return** 1  **return** abs((x2 - x1) / x2)  **def** **functie**(x):  **return** 2 \* x + 3 \* math.cos(2 \* x)  **def** **Secanta**(f, x0, x1, eps = 0.00001):  v = [x0, x1]  k = 1   **while** eroare(v[k-1], v[k]) >= eps:  k += 1  xk = (v[k-2] \* f(v[k-1]) - v[k-1] \* f(v[k-2])) / (f(v[k-1]) - f(v[k-2]))  v.append(xk)    **return** {'x': v[k], 'it': k - 1}  **def** **PozitieFalsa**(f, a, b, eps = 0.00005):  k = 0  va = [a]  vb = [b]  x = [(a\*f(b) - b \* f(a)) / (f(b) - f(a))]   k = 0   **while** k == 0 **or** eroare(x[k-1], x[k]) >= eps:  k += 1  **if** f(x[k-1]) == 0:  x.append(x[k-1])  **break**   **elif** f(va[k-1])\*f(x[k-1]) < 0:  va.append(va[k-1])  vb.append(x[k-1])  x.append((va[k]\*f(vb[k]) - vb[k] \* f(va[k])) / (f(vb[k]) - f(va[k])))  **else**:  va.append(x[k-1])  vb.append(vb[k-1])  x.append((va[k]\*f(vb[k]) - vb[k] \* f(va[k])) / (f(vb[k]) - f(va[k])))   **return** {'x': x[k], 'nr\_it': k}      puncte1 = [] puncte2 = []  st = -2 fin = 0 pt1 = Secanta(functie, st, fin, 0.000001) pt2 = PozitieFalsa(functie, st, fin, 0.000001) puncte1.append(pt1) puncte2.append(pt2)  st = 0 fin = 1.4 pt1 = Secanta(functie, st, fin, 0.000001) pt2 = PozitieFalsa(functie, st, fin, 0.000001) puncte1.append(pt1) puncte2.append(pt2)  st = 1.4 fin = 3 pt1 = Secanta(functie, st, fin, 0.000001) pt2 = PozitieFalsa(functie, st, fin, 0.000001) puncte1.append(pt1) puncte2.append(pt2)  print("Metoda Secantei:") print(puncte1)  print("Metoda Pozitiei False:") print(puncte2)  desen\_puncte = [] **for** i **in** range(len(puncte1)):  desen\_puncte.append(puncte1[i]['x'])  fig = plt.figure(1) ax = plt.axes() x = np.linspace(-2, 3, 200) y = np.vectorize(functie)(x) ax.plot(x, y, linestyle='-', lw = 2, color = 'b', label = 'y = f(x)') ax.scatter(desen\_puncte, np.zeros(3), color="g") ax.grid(**True**, color = 'c') plt.xticks(np.arange(-2,3,0.5)) plt.yticks(np.arange(-5, 5, 1)) ax.legend(loc='best') plt.title('f(x) = y') ax.axhline(y=0,color='k') ax.axvline(x=0,color='k') plt.show() |
| --- |



| Metoda Secantei: [{'x': -0.5850604750013056, 'it': 6}, {'x': 1.3315894416848741, 'it': 10}, {'x': 1.4690501969855176, 'it': 12}] Metoda Pozitiei **False**: [{'x': -0.5850604658662881, 'nr\_it': 5}, {'x': 1.3315918795349821, 'nr\_it': 26}, {'x': 1.4690417695067062, 'nr\_it': 66}] |
| --- |

9

| **import** numpy **as** np  **def** **citeste**():  dim = 0  A = **None**  B = **None**  **with** open("in.txt", 'r') **as** file:  dim = int(file.readline())  A = np.zeros((dim, dim))  **for** i **in** range(dim):  A[i] = [float(x) **for** x **in** file.readline().split()]   B = np.zeros(dim)  B = [float(x) **for** x **in** file.readline().split()]   **return** A, B  **def** **GaussJordan**(A, b, eps = 10 \*\* -16):  A\_extins = np.c\_[A, b]  n = A.shape[0]   **for** k **in** range(n):  p = k  **while** p < n **and** abs(A\_extins[p][k]) < eps:  p += 1   **if** abs(A\_extins[p][k]) < eps:  print("Sistemul nu este determinat")  **return** **None**   **if** k != p:  A\_extins[[k, p]] = A\_extins[[p, k]]   **for** l **in** range(n):  **if** l != k :  mlk = A\_extins[l][k] / A\_extins[k][k]  A\_extins[l] = A\_extins[l] - A\_extins[k] \* mlk    **if** abs(A\_extins[n-1][n-1]) < eps:  print("Sistemul nu este determinat")  **return** **None**   **return** A\_extins  **def** **Diagonala**(diag):  n = diag.shape[0]   rez = np.zeros(n)  **for** i **in** range(n):  rez[i] = diag[i][n] / diag[i][i]  **return** rez   A, B = citeste()  diag = GaussJordan(A, B) print(diag)  res = Diagonala(diag) print(res) |
| --- |

Rezultat

| [[ 1. 0. 0. 1.26666667]  [ 0. -3. 0. 0.2 ]  [ 0. 0. 5. 1. ]] [ 1.26666667 -0.06666667 0.2 ] |
| --- |