Exercício 1

a)

Considerando f(x) = 1/2*Xt*A*X* - *Xt*b

f´(x) = A*X - b

Para encontrar-se o mínimo, f´(x) = 0, portanto:

A*X - b = 0

A*x = b

b)

In [6]:
```
import numpy as np

A = np.array([[2, -1],
              [-1, 1]])

b = np.array([1, 1])

x = np.linalg.solve(A, b)
print("The solution is:")
print(x)
```

```
The solution is:
[2. 3.]
```

c) Os problemas (a) e (b) se relacionam na medida em que para se encontrar o vetor x que resulta no mínimo da função f(x), é necessário utilizar o sistema de equações lineares apresentado no problema (b)

Exercício 2

In [3]:
```
def Dichotomous_Search_Algorithm(a, b, func, tol=1, E = 0.001, max_iter=500):
    ak   = a
```

```python
    bk   = b
    iter = 0

    print(f"{'iter':>10} {'ak':>10} {'bk':>10} {'c1':>10} {'c2':>10} {'func(c1)':>15} {'func(c2)':>15} {'midpoint':>12} {'dist
    print("-" * 120)

    while ((bk - ak) > tol) and (iter < max_iter):
        c1    = (ak + bk) / 2 - E
        c2    = (ak + bk) / 2 + E
        iter += 1

        print(f"{iter:10.0f} {ak:10.4f} {bk:10.4f} {c1:10.4f} {c2:10.4f} {func(c1):15.6f} {func(c2):15.6f} {(ak + bk) / 2:12.4

        if func(c1) < func(c2): bk = c2
        else:ak = c1

    print(f"{iter+1:10.0f} {ak:10.4f} {bk:10.4f} {c1:10.4f} {c2:10.4f} {func(c1):15.6f} {func(c2):15.6f} {(ak + bk) / 2:12.4f}

    return (ak + bk) / 2
```

In [4]:

```python
def func1(x):
    return x**4 - 14*(x**3) + 60*(x**2) - 70*x

res_1 = Dichotomous_Search_Algorithm(0, 2, func1, tol=0.001, E = 0.000001)
print(f"\nMinimum found at x = {res_1}, f(x) = {func1(res_1)}")
```

iter	ak	bk	c1	c2	func(c1)	func(c2)	midpoint	distance
1	0.0000	2.0000	1.0000	1.0000	-23.000012	-22.999988	1.0000	-2.0000
2	0.0000	1.0000	0.5000	0.5000	-21.687490	-21.687530	0.5000	-1.0000
3	0.5000	1.0000	0.7500	0.7500	-24.339842	-24.339846	0.7500	-0.5000
4	0.7500	1.0000	0.8750	0.8750	-24.105229	-24.105218	0.8750	-0.2500
5	0.7500	0.8750	0.8125	0.8125	-24.339098	-24.339094	0.8125	-0.1250
6	0.7500	0.8125	0.7812	0.7813	-24.369597	-24.369597	0.7813	-0.0625
7	0.7500	0.7813	0.7656	0.7656	-24.362376	-24.362378	0.7656	-0.0313
8	0.7656	0.7813	0.7734	0.7734	-24.367885	-24.367886	0.7734	-0.0156
9	0.7734	0.7813	0.7773	0.7773	-24.369214	-24.369215	0.7773	-0.0078
10	0.7773	0.7813	0.7793	0.7793	-24.369524	-24.369524	0.7793	-0.0039
11	0.7793	0.7813	0.7803	0.7803	-24.369590	-24.369590	0.7803	-0.0020
12	0.7803	0.7813	0.7803	0.7803	-24.369590	-24.369590	0.7808	-0.0010

Minimum found at x = 0.7807619379882811, f(x) = -24.369601107124794

In [5]:
```
def func2(x):
    return (1/4)*(x**4) - (5/3)*(x**3) - 6*(x**2) + 19*x - 7

res_2 = Dichotomous_Search_Algorithm(-4,4, func2, tol=0.001, E = 0.000001)
print(f"\nMinimum found at x = {res_2}, f(x) = {func2(res_2)}")
```

iter	ak	bk	c1	c2	func(c1)	func(c2)	midpoint	distance
1	-4.0000	4.0000	-0.0000	0.0000	-7.000019	-6.999981	0.0000	-8.0000
2	-4.0000	0.0000	-2.0000	-2.0000	-51.666674	-51.666644	-2.0000	-4.0000
3	-4.0000	-2.0000	-3.0000	-3.0000	-52.749996	-52.750030	-3.0000	-2.0000
4	-3.0000	-2.0000	-2.5000	-2.5000	-56.192709	-56.192705	-2.5000	-1.0000
5	-3.0000	-2.5000	-2.7500	-2.7500	-55.665688	-55.665701	-2.7500	-0.5000
6	-2.7500	-2.5000	-2.6250	-2.6250	-56.202087	-56.202091	-2.6250	-0.2500
7	-2.6250	-2.5000	-2.5625	-2.5625	-56.262488	-56.262488	-2.5625	-0.1250
8	-2.6250	-2.5625	-2.5938	-2.5937	-56.248948	-56.248950	-2.5937	-0.0625
9	-2.5938	-2.5625	-2.5781	-2.5781	-56.259834	-56.259835	-2.5781	-0.0313
10	-2.5781	-2.5625	-2.5703	-2.5703	-56.262184	-56.262185	-2.5703	-0.0156
11	-2.5703	-2.5625	-2.5664	-2.5664	-56.262591	-56.262591	-2.5664	-0.0078
12	-2.5664	-2.5625	-2.5645	-2.5645	-56.262603	-56.262603	-2.5645	-0.0039
13	-2.5664	-2.5645	-2.5654	-2.5654	-56.262613	-56.262613	-2.5654	-0.0020
14	-2.5654	-2.5645	-2.5654	-2.5654	-56.262613	-56.262613	-2.5649	-0.0010

Minimum found at x = -2.564940765014648, f(x) = -56.26261218274178

In [ ]:

```
def Fibonacci_Search_Algorithm(a, b, func, tol=1e-5):
    fib = [1, 1]
    n = 1
    while fib[n] < ((b - a) / tol):
        fib.append(fib[n] + fib[n-1])
        n += 1

    ak = a
    bk = b

    print(f"{'Iter':>6} {'ak':>12} {'bk':>12} {'b-a':>12} {'c1':>12} {'c2':>12} {'f(c1)':>15} {'f(c2)':>15}")
    print("-" * 99)

    for k in range(1, n):
        iter_count = k

        L_k = bk - ak
        c1 = ak + (fib[n - 2] / fib[n]) * L_k
        c2 = bk - (fib[n - 2] / fib[n]) * L_k

        f_c1 = func(c1)
        f_c2 = func(c2)

        if f_c1 < f_c2: bk = c2
        else: ak = c1

        print(f"{iter_count:6.0f} {ak:12.4f} {bk:12.4f} {(bk-ak):12.4f} {c1:12.4f} {c2:12.4f} {f_c1:15.6f} {f_c2:15.6f}")

    return (ak + bk) / 2
```

In [22]:

```
res_1 = Fibonacci_Search_Algorithm(0, 2, func1, tol=0.001)
print(f"\nMinimum found at x = {res_1}, f(x) = {func1(res_1)}")
```

```
 Iter          ak          bk         b-a          c1          c2         f(c1)         f(c2)
----------------------------------------------------------------------------------------------------
    1      0.0000      1.2361      1.2361      0.7639      1.2361    -24.360680    -18.958158
    2      0.4721      1.2361      0.7639      0.4721      0.7639    -21.098514    -24.360680
    3      0.4721      0.9443      0.4721      0.7639      0.9443    -24.360680    -23.592461
    4      0.6525      0.9443      0.2918      0.6525      0.7639    -23.837435    -24.360680
    5      0.6525      0.8328      0.1803      0.7639      0.8328    -24.360680    -24.287887
    6      0.7214      0.8328      0.1115      0.7214      0.7639    -24.257948    -24.360680
    7      0.7639      0.8328      0.0689      0.7639      0.7902    -24.360680    -24.366907
    8      0.7639      0.8065      0.0426      0.7902      0.8065    -24.366907    -24.349526
    9      0.7639      0.7902      0.0263      0.7802      0.7902    -24.369587    -24.366907
   10      0.7740      0.7902      0.0163      0.7740      0.7802    -24.368128    -24.369587
   11      0.7740      0.7840      0.0101      0.7802      0.7840    -24.369587    -24.369296
   12      0.7778      0.7840      0.0062      0.7778      0.7802    -24.369312    -24.369587
   13      0.7778      0.7817      0.0038      0.7802      0.7817    -24.369587    -24.369583
   14      0.7793      0.7817      0.0024      0.7793      0.7802    -24.369523    -24.369587
   15      0.7802      0.7817      0.0015      0.7802      0.7808    -24.369587    -24.369601
   16      0.7802      0.7811      0.0009      0.7808      0.7811    -24.369601    -24.369600

Minimum found at x = 0.7806464204687811, f(x) = -24.369599824480176
```

In [31]:

```python
res_2 = Fibonacci_Search_Algorithm(-4, 4, func2, tol=0.001)
print(f"\nMinimum found at x = {res_2}, f(x) = {func2(res_2)}")
```

Iter	ak	bk	b-a	c1	c2	f(c1)	f(c2)
1	-4.0000	0.9443	4.9443	-0.9443	0.9443	-28.689037	4.386763
2	-4.0000	-0.9443	3.0557	-2.1115	-0.9443	-53.209169	-28.689038
3	-4.0000	-2.1115	1.8885	-2.8328	-2.1115	-54.984856	-53.209170
4	-3.2786	-2.1115	1.1672	-3.2786	-2.8328	-46.163736	-54.984856
5	-2.8328	-2.1115	0.7214	-2.8328	-2.5573	-54.984856	-56.261557
6	-2.8328	-2.3870	0.4458	-2.5573	-2.3870	-56.261557	-55.755797
7	-2.6625	-2.3870	0.2755	-2.6625	-2.5573	-56.100681	-56.261557
8	-2.6625	-2.4922	0.1703	-2.5573	-2.4922	-56.261557	-56.175255
9	-2.5975	-2.4922	0.1052	-2.5975	-2.5573	-56.245120	-56.261557
10	-2.5975	-2.5324	0.0650	-2.5573	-2.5324	-56.261557	-56.244790
11	-2.5975	-2.5573	0.0402	-2.5726	-2.5573	-56.261700	-56.261557
12	-2.5821	-2.5573	0.0248	-2.5821	-2.5726	-56.257835	-56.261700
13	-2.5726	-2.5573	0.0154	-2.5726	-2.5668	-56.261700	-56.262575
14	-2.5726	-2.5631	0.0095	-2.5668	-2.5631	-56.262575	-56.262540
15	-2.5690	-2.5631	0.0059	-2.5690	-2.5668	-56.262377	-56.262575
16	-2.5668	-2.5631	0.0036	-2.5668	-2.5654	-56.262575	-56.262613
17	-2.5668	-2.5645	0.0022	-2.5654	-2.5645	-56.262613	-56.262605
18	-2.5659	-2.5645	0.0014	-2.5659	-2.5654	-56.262606	-56.262613
19	-2.5659	-2.5651	0.0009	-2.5654	-2.5651	-56.262613	-56.262613

```
Minimum found at x = -2.565487292094689, f(x) = -56.262612738046975
```

Os métodos acima apresentam vantagens sobre a diferenciação de funções, pois nem sempre a função ou a sua derivada é conhecida. Além disso, através deles é sempre possível dizer o quão distante você está do ponto mínimo. Por outro lado, a convergência desses métodos dependerá do passo a ser dado e da tolerância estipulada, o que pode demorar muito para ser atingida ou cair em uma região de oscilação indefinida, sem chegar na convergência efetiva do mínimo da função.