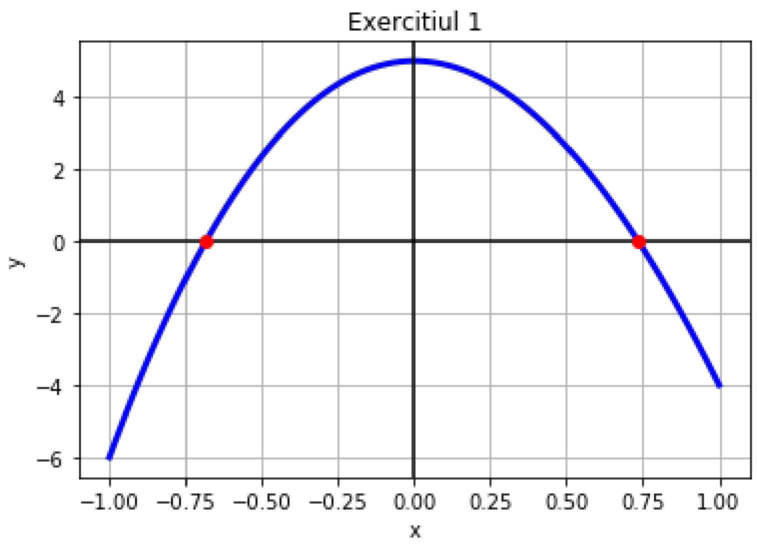
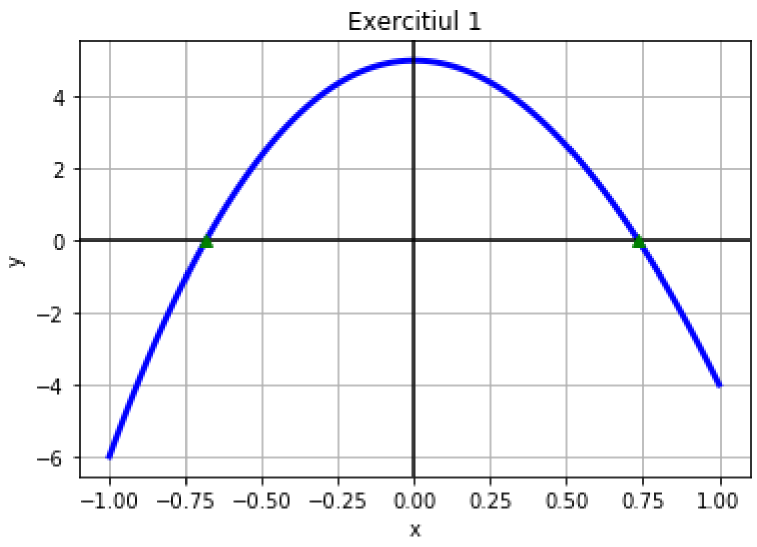
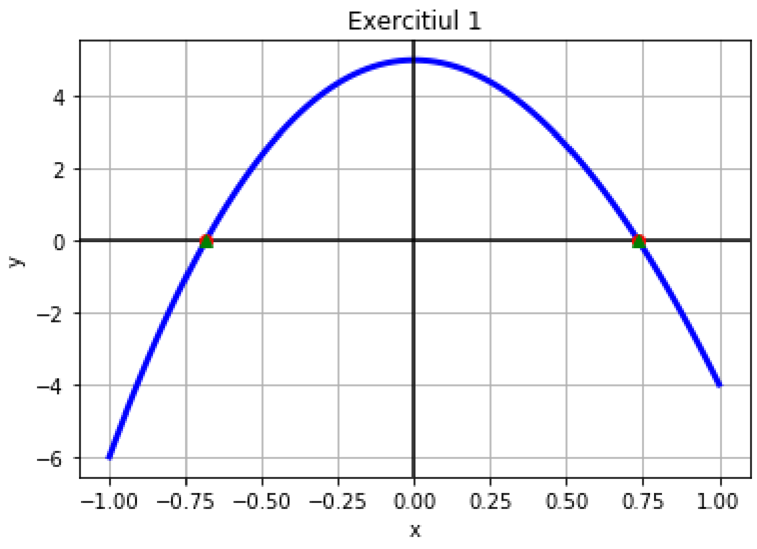
In [1]:

```python
# -*- coding: utf-8 -*-
"""
Created on Mon Oct 26 12:00:10 2020

@author: Furculesteanu Bianca Grupa 341 Varianta 16
"""

import numpy as np
import matplotlib.pyplot as plt

def f(x):
    y=x**3-10*x**2+5
    return y

def MetSecantei(f, x0, x1, tol):
    k = 1
    x2 = 0
    while abs(x1 - x0) / abs(x0) >= tol:

        x2 = (x0 * f(x1) - x1 * f(x0))  / (f(x1) - f(x0))
        k = k + 1
        x0 = x1
        x1 = x2
    return x1

def MetPozFalse(f, a, b, tol):
    k = 0
    a0 = a
    b0 = b
    x0 = (a0 * f(b0) - b0 * f(a0)) / (f(b0) - f(a0))

    k = k + 1
    if f(x0) == 0:
        x1 = x0
        return x0
    elif (f(a0) * f(x0)) < 0:
        b0 = x0
        x1 = (a0*f(b0) - b0 * f(a0)) / (f(b0) - f(a0))
    else:
        a0 = x0
        x1 = (a0*f(b0) - b0 * f(a0)) / (f(b0) - f(a0))

    while abs(x1 - x0) / abs(x0) >= tol:
        x0 = x1
        if f(x0) == 0:
            x1 = x0
            return x0
        elif (f(a0) * f(x0)) < 0:
            b0 = x0
            x1 = (a0*f(b0) - b0 * f(a0)) / (f(b0) - f(a0))
        else:
            a0 = x0
            x1 = (a0*f(b0) - b0 * f(a0)) / (f(b0) - f(a0))
    return x1

(a,b)=(-1,1)
x_grafic=np.linspace(-1,1,100)
y_grafic= f(x_grafic)
plt.plot(x_grafic,y_grafic, "-", color="blue", linewidth=3)
plt.axvline(0, color='black')
plt.axhline(0, color='black')
plt.xlabel('x')
plt.ylabel('y')
plt.title('Exercitiul 1')
plt.grid(True)

interval1 = MetSecantei(f,-0.75,-0.50,10**(-5))
interval2 = MetSecantei(f,0.75,0.50,10**(-5))
print(interval1)
print(interval2)
interval3 = MetPozFalse(f,-0.75,-0.50,10**(-5))
interval4 = MetPozFalse(f,0.75,0.50,10**(-5))
print(interval3)
print(interval4)
plt.plot([interval1, interval2], [0,0], 'ro')
plt.show()
x_grafic=np.linspace(-1,1,100)
y_grafic= f(x_grafic)
plt.plot(x_grafic,y_grafic, "-", color="blue", linewidth=3)
plt.axvline(0, color='black')
plt.axhline(0, color='black')
plt.xlabel('x')
plt.ylabel('y')
plt.title('Exercitiul 1')
plt.grid(True)

plt.plot([interval3, interval4], [0,0], 'g^')
plt.show()

x_grafic=np.linspace(-1,1,100)
y_grafic= f(x_grafic)
plt.plot(x_grafic,y_grafic, "-", color="blue", linewidth=3)
plt.axvline(0, color='black')
plt.axhline(0, color='black')
plt.xlabel('x')
plt.ylabel('y')
plt.title('Exercitiul 1')
plt.grid(True)
plt.plot([interval1, interval2], [0,0], 'ro')
plt.plot([interval3, interval4], [0,0], 'g^')
plt.show()
```

```
-0.6840945657039618
0.734603507619331
-0.6840944975840434
0.7346035058695684
```






Exercitiul 1
4
2
0
−2
−4
−6
y
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00
x

In [3]:

```
# -*- coding: utf-8 -*-
"""
Created on Sun Nov  1 17:09:30 2020

@author: Furculesteanu Bianca Grupa 341 Varianta 16
"""
import numpy as np
import matplotlib.pyplot as plt
n = 15

def SubsDesc(A, b, tol):
        """
        param A: matrice patratica, superior triunghiulara, cu toate elem pe diag princ
ipala nenule
        param b: vectorul term liberi
        param tol: val numerica ft mica in rap cu care vom compara numerele apropiate d
e 0
        return: sol sistem
        """


        m, n = np.shape(A)
        if m != n:
                print("Matricea nu este patratica. Introduceti o alta matrice.")
                x = None
                return x

        #Verificam daca matricea este superior triunghiulara
        for i in range(m):
                for j in range(i):
                        if abs(A[i][j]) > tol:
                                print("Matricea nu este superior triunghiulara")
                                x = None
                                return x

        #Verificam daca elementele de pe diagonala principala sunt nule (sist comp det)
        for i in range(n):
                if A[i][i] == 0:
                        print("Sistemul nu este compatibil determinat")
                        x = None
                        return x

        x = np.zeros((n,1))
        x[n-1] = 1 / A[n-1][n-1] * b[n-1]
        k = n-2
        while k >= 0:
                sum = 0
                for i in range(k+1, n):
                        sum += A[k][i] * x[i]
                x[k] = 1 / A[k][k] * (b[k] - sum)
                k = k - 1
        return x

def GaussPivTot(A,b):
    a = np.zeros((n, n + 1))
    for i in range(n):
        for j in range(n + 1):
            if j == n:
                a[i][j] = b[i]
            else:
                a[i][j] = A[i][j]

    for k in range(0, n):
        max = 0
        for i in range(k, n):
            for j in range(k, n + 1):
                if abs(abs(a[i][j]) > max):
                    max = abs(a[i][j])
                    p = i
                    m1 = j
        if a[p][m1] == 0:
            print('Sistem incompatibil sau compatibil nedeterminat')
            return 0
        if p != k:
            a[[p, k]] = a[[k, p]]
        if m1 != k:
            a[:,[m1,k]] = a[:,[k,m1]]
            aux = m1
            m1 = k
            k = m1
        for l in range(k + 1, n):
            ml = a[l][k] / a[k][k]
            for j in range(n +1):
                a[l][j] = a[l][j] - m1 * a[k][j]
    if a[n - 1][n - 1] == 0:
        print('Sistem incompatibil sau compatibil nedeterminat')
        return 0
    for i in range(n):
        for j in range(n):
            A[i][j] = a[i][j]
    for i in range(n):
        b[i] = a[i][n]
    y = SubsDesc(A, b, 10**(-15))
    return y




A = np.zeros((15, 15))
c = -3
d = 9
f = -8


A[0][0] = d
A[0][1] = f

A[1][0] = c
A[1][1] = d
A[1][2] = f

for i in range(2,n):
    for j in range(n):
        A[i][j] = A[i - 1][j - 1]

b = np.zeros((n, 1))

for i in range(1,n-1):
    b[i] = 1
b[0] = 2
b[n-1] = 2
```

```
g = GaussPivTot(A,b)
print(g)
```

```
Matricea nu este superior triunghiulara
None
```

In [4]:

```
# -*- coding: utf-8 -*-
"""
Created on Sun Nov  1 20:23:29 2020

@author: Furculesteanu Bianca Grupa 341 Varianta 16
"""

import numpy as np

def metSubsDesc(A, b, tol):
        """
        param A: matrice patratica, superior triunghiulara, cu toate elem pe diag princ
ipala nenule
        param b: vectorul term liberi
        param tol: val numerica ft mica in rap cu care vom compara numerele apropiate d
e 0
        return: sol sistem
        """

        m, n = np.shape(A)
        if m != n:
                print("Matricea nu este patratica. Introduceti o alta matrice.")
                x = None
                return x

        #Verificam daca matricea este superior triunghiulara
        for i in range(m):
                for j in range(i):
                        if abs(A[i][j]) > tol:
                                print("Matricea nu este superior triunghiulara")
                                x = None
                                return x

        #Verificam daca elementele de pe diagonala principala sunt nule (sist comp det)
        for i in range(n):
                if A[i][i] == 0:
                        print("Sistemul nu este compatibil determinat")
                        x = None
                        return x

        x = np.zeros((n,1))
        x[n-1] = 1 / A[n-1][n-1] * b[n-1]
        k = n-2
        while k >= 0:
                sum = 0
                for i in range(k+1, n):
                        sum += A[k][i] * x[i]
                x[k] = 1 / A[k][k] * (b[k] - sum)
                k = k - 1
        return x


def gaussPp(A, b, tol):
        """
        param A: matricea asoc sistemului, patratica
        param b: vectorul term liberi
        param tol: val cu care comparam nr nenule
        return x: solutia sistemului
        """
        m, n = np.shape(A)
        if m != n:
                print("Matricea nu este patratica. Introduceti o alta matrice.")
                x = None
                return x

        A_extins = np.concatenate((A, b), axis=1) #axis=0 l-ar pune pe b o noua linie,
 1 il pune drept coloana
        for k in range(n-1):
                max = A_extins[k][k]
                p = k
                for j in range(k+1, n):
                        if abs(A_extins[j][k]) > abs(max):
                                max = A_extins[j][k]
                                p = j

                if abs(max) <= tol:
                        print("Sistemul nu admite solutie unica")
                        x = None
                        return x

                if (p != k):
                        A_extins[[p, k]] = A_extins[[k, p]] #swap linia p cu linia k

                for j in range(k+1, n):
                        A_extins[j] = A_extins[j] - (A_extins[j][k] / A_extins[k][k]) *
A_extins[k]
                print(A_extins)
        if abs(A_extins[n-1][n-1]) <= tol:
                print("Sistemul nu admite solutie unica")
                x = None
                return x

        x = metSubsDesc(A_extins[:, 0:n], A_extins[:, n], tol)
        return x

A = np.array([[9,18,19], [15,13,12],[3,2,2]], float)
b = np.array([[84], [47], [7]], float)
tol = 10 ** -16

x = gaussPp(A, b, tol)
print(x)
```

```
[[15.   13.   12.   47. ]
 [ 0.   10.2 11.8 55.8]
 [ 0.   -0.6 -0.4 -2.4]]
[[15.         13.         12.         47.        ]
 [ 0.         10.2        11.8        55.8       ]
 [ 0.          0.          0.29411765  0.88235294]]
[[-1.]
 [ 2.]
 [ 3.]]
```

In [ ]: