Лабораторная работа №1 задание 1

Доскоч Роман вариант 9

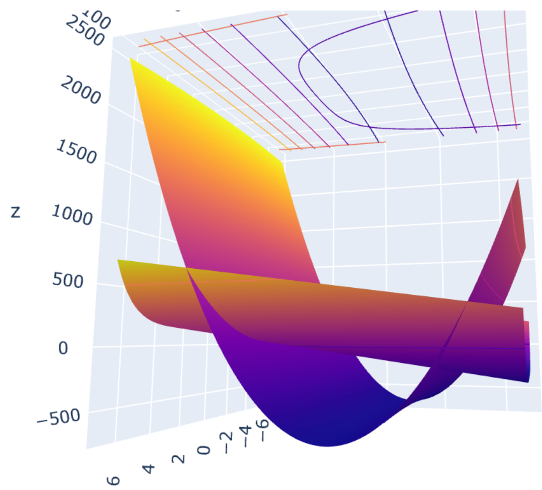
Решение системы нелинейных уравнений

$$\begin{cases} chy + 2x - 45 = 0 \\ \frac{x^2}{5} - y^2 + 10x - 500 = 0 \end{cases}$$

In [1]:
```
import plotly as pl
import numpy as np
import plotly.graph_objs as go
import plotly.express as px
import matplotlib.pyplot as plt
import math
from math import sinh
```

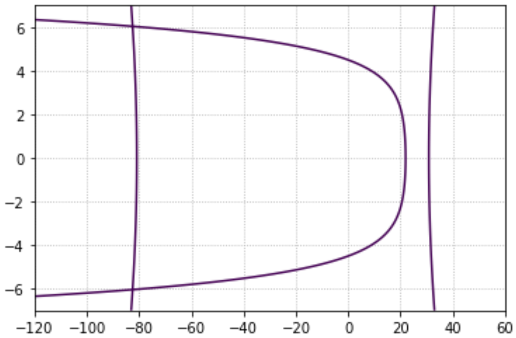
In [2]:
```
f1 = lambda x, y: np.cosh(y)+2*x-45
f2 = lambda x, y: 1/5*x**2-y**2+10*x-500
```

In [3]:
```
x, y = np.meshgrid(np.linspace(-120, 100, 300), np.linspace(-7, 7, 300))
fig = go.Figure(go.Surface(x=x,y=y,z=f1(x, y)))
fig.add_surface(x=x,y=y,z=f2(x, y))
fig.update_traces(contours_z=dict(show=True, usecolormap=True,highlightcolor="limegreen", project_z=True))
fig.show()
```



Срез графика при $z = 0$
можно заметить что корней всего 2
найдем один из них с начальной точкой
$(x_0, y_0) = (-50, 7)$

In [4]:
```
x,y = np.meshgrid(np.linspace(-120, 60, 300), np.linspace(-7, 7, 300))
plt.contour(x,y,f1(x,y),[0])
plt.contour(x,y,f2(x,y),[0])
plt.grid(ls=':')
```



In [5]:
```
#метод крамера для решения 2-матрицы
def Cramer(X,f):
    a1, a2, a3, a4 = X[0][0], X[0][1], X[1][0], X[1][1]
    b1, b2 = f[0], f[1]
    y = (a3*b1-a1*b2)/(a3*a2-a1*a4)
    x = (b2 - a4*y)/a3
    return x, y
```

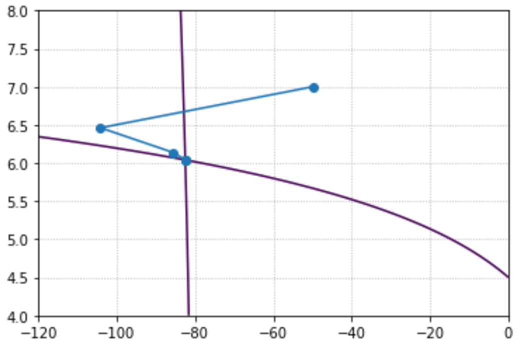
Метод Ньютона

$$\mathcal{J} = \begin{pmatrix} 2 & \sinh y \\ 0.4x + 10 & -2y \end{pmatrix}$$

In [6]:
```
def solver(x, y, occur=1e-6):
    iters=0
    path_X, path_Y = [x], [y]
    while(abs(f1(x,y)) > occur or abs(f2(x,y)) > occur):
        J = [[      2,     sinh(y)],
             [0.4*x+10,    -2*y    ]]
        _x, _y = Cramer(J,[-f1(x,y),-f2(x,y)])
        iters, x, y = iters+1, x + _x, y + _y

        path_X.append(x)
        path_Y.append(y)

    return iters, (x,y), (path_X, path_Y)
```

In [7]:
```
x,y = np.meshgrid(np.linspace(-120, 0, 300), np.linspace(4, 8, 300))
plt.contour(x,y,f1(x,y),[0])
plt.contour(x,y,f2(x,y),[0])
plt.grid(ls=':')

iters, root, path, = solver(-50.0, 7)
plt.plot(*path,'-o')
print(f'{iters=}')
print(f'(x,y) = {root}')
print(f'f1(x,y) = {f1(*root)} ≈ 0')
print(f'f2(x,y) = {f2(*root)} ≈ 0')
```

```
iters=5
(x,y) = (-82.51025073109597, 6.040346664741784)
f1(x,y) = 1.990122200368205e-08 ≈ 0
f2(x,y) = 4.476987669477239e-10 ≈ 0
```



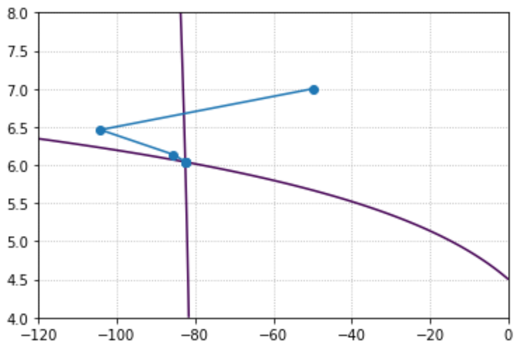
Дискретный метод Ньютона

$$\mathcal{J} = \begin{pmatrix} \frac{f_1(x+h,y)-f_1(x,y)}{h} & \frac{f_1(x,y+h)-f_1(x,y)}{h} \\ \frac{f_2(x+h,y)-f_1(x,y)}{h} & \frac{f_2(x,y+h)-f_2(x,y)}{h} \end{pmatrix}$$

In [8]:
```
def discrete_solver(x, y, h, occur=1e-6):
    iters=0
    path_X, path_Y = [x], [y]
    while(abs(f1(x,y)) > occur or abs(f2(x,y)) > occur):
        J = [[(f1(x+h,y) - f1(x,y))/h, (f1(x,y+h)-f1(x,y))/h],
             [(f2(x+h,y) - f2(x,y))/h, (f2(x,y+h)-f2(x,y))/h]]
        _x, _y = Cramer(J,[-f1(x,y),-f2(x,y)])
        iters, x, y = iters+1, x + _x, y + _y

        path_X.append(x)
        path_Y.append(y)

    return iters, (x,y), (path_X, path_Y)
```

In [9]:
```
x,y = np.meshgrid(np.linspace(-120, 0, 300), np.linspace(4, 8, 300))
plt.contour(x,y,f1(x,y),[0])
plt.contour(x,y,f2(x,y),[0])
plt.grid(ls=':')

iters, root, path, = discrete_solver(-50.0, 7, h=.001)
plt.plot(*path,'-o')

print(f'{iters=}')
print(f'(x,y) = {root}')
print(f'f1(x,y) = {f1(*root)} ≈ 0')
print(f'f2(x,y) = {f2(*root)} ≈ 0')
```

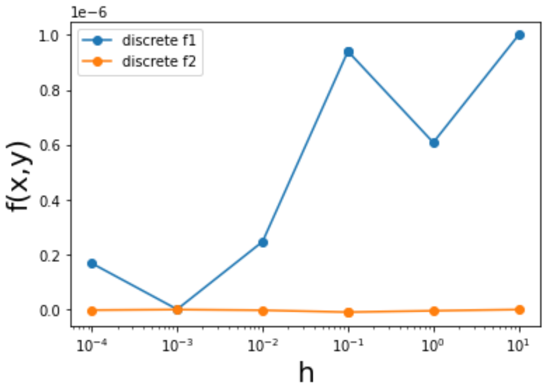
```
iters=6
(x,y) = (-82.51025073102824, 6.040346664650646)
f1(x,y) = 8.960796549217775e-10 ≈ 0
f2(x,y) = -9.322320693172514e-12 ≈ 0
```



Зависимость выбора h для дискретного метода точности 1e-6

In [10]:
```
start_point = (-50.0, 7)
H = [1e-4, 1e-3, 1e-2, 1e-1, .1, 1, 10]

data = [[f1(*discrete_solver(*start_point,h)[1]),
         f2(*discrete_solver(*start_point,h)[1])] for h in H]

pl = plt.plot(H, data, '-o')
plt.legend(iter(pl), ('discrete f1','discrete f2'));
ax = plt.gca()
plt.xscale('log',base=10)
ax.set_xlabel("h", fontsize=20)
ax.set_ylabel("f(x,y)", fontsize=20)
```

Out[10]:
```
Text(0, 0.5, 'f(x,y)')
```



Первая функция с гиперболическим косинусом сильно зависима от h в то время как 2 функция нет.
все из-за уровня "сложности"(кривизны) производной. Лучшим выбором h является 10^{-3}, если рассматривать слишком маленькие h возникнет проблема с обусловленностью.

В обоих случаях быстрая скорость сходимости обьясняется удачно подобранными начальынми точками и быстрым наклоном функции.

Так же доказано что у метода ньютона квадратичная сходимость.
В итоге метод ньютона сошёлся за 5 итераций, а дискретный аналог за 6.

In [11]:
```
def N(x,y):
    iters=0
    while(abs(f1(x,y)) > 1e-6 or abs(f2(x,y)) > 1e-6):
        J = [[     2,     sinh(y)],
             [0.4*x+10,    -2*y   ]]
        _x, _y = Cramer(J,[-f1(x,y),-f2(x,y)])
        iters, x, y = iters+1, x + _x, y + _y
    return iters

def Newton(x, y):
    return [[N(i,j) for i,j in zip(xi,yj)]for xi,yj in zip(x,y)]
```

Зависимость выбора начальной точки на колличества итераций точности 1e-6

In [12]:
```
x, y = np.meshgrid(np.linspace(-210, -40, 500), np.linspace(5, 7.5, 500))
fig = go.Figure(go.Surface(x=x,y=y,z=Newton(x, y)))
fig.update_traces(contours_z=dict(show=True, project_z=True))
fig.show()
```