Введение в высшую математику

Урок 3. Видеоурок “Введение в аналитическую геометрию. Графики на плоскости”

Практическое задание по теме “Введение в аналитическую геометрию”:
https://docs.google.com/document/d/1tY1GsY8fjo0Fl6NeYVaeEWUzOEfAIBYR6xvUU-cZ8Wg/edit#
(https://docs.google.com/document/d/1tY1GsY8fjo0Fl6NeYVaeEWUzOEfAIBYR6xvUU-cZ8Wg/edit)

Практическое задание по теме “Графики на плоскости”:
https://docs.google.com/document/d/1CLiXsJUaJxflizBWcneCqeX1uV84a3GV60tp76N1VQs/edit#
(https://docs.google.com/document/d/1CLiXsJUaJxflizBWcneCqeX1uV84a3GV60tp76N1VQs/edit)
Функция для №1: y = k * np.cos(x - a) + b

Задание выполнил Соковнин ИЛ

Практическое задание

0 Задания к уроку 2

1. Задание

Даны два вектора в трехмерном пространстве: (10,10,10) и (0,0,-10)

1) Найдите их сумму. (на листочке)

2) Напишите код на Python, реализующий расчет длины вектора, заданного его координатами. (в программе)

1.1 Найти сумму векторов a=(10,10,10) и b=(0,0,-10)

Сумма векторов **a** и **b**

$$(a + b) = (a_x + b_x; a_y + b_y; a_z + b_z)$$

Решениее:

$$(a + b) = (a_x + b_x; a_y + b_y; a_z + b_z) = (10 + 0; 10 + 0; 10 - 10) = (10; 10; 0)$$

1.2 Напишите код на Python, реализующий расчет длины вектора, заданного его координатами. (в программе)

B [1]:
```
from typing import List

def sum_vectors(a:  List[float], b: List[float]) ->  List[float]:
    '''Поиск суммы 2-х векторов'''

    if len(a) <= len(b):
        length = len(a)
    else:
        length = len(b)

    vector_c = []
    for i in range(length):
        vector_c.append(a[i] + b[i])

    print(f'a+b={vector_c}')


a=[10,10,10]
b=[0,0,-10]
# a=1
# b=2

try:
    sum_vectors(a, b),
except TypeError:
    print('TypeError: Вектора должны быть заданы списками чисел.')
```

```
a+b=[10, 10, 0]
```

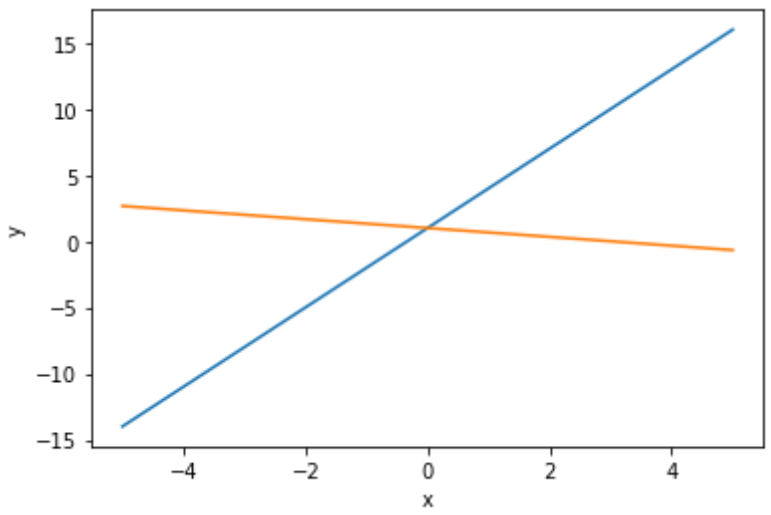
2. Задание

Задание (на листочке)
Почему прямые не кажутся перпендикулярными? (см.ролик)

B [2]:
```
%matplotlib inline
import numpy as np
import matplotlib.pyplot as plt
import math
```

B [3]:
```
x = np.linspace(-5, 5, 21)
y = 3*x+1
y2 = (-1/3)*x+1
plt.plot(x,y)
plt.plot(x,y2)
plt.xlabel("x")
plt.ylabel("y")
```

Out[3]:
```
Text(0, 0.5, 'y')
```

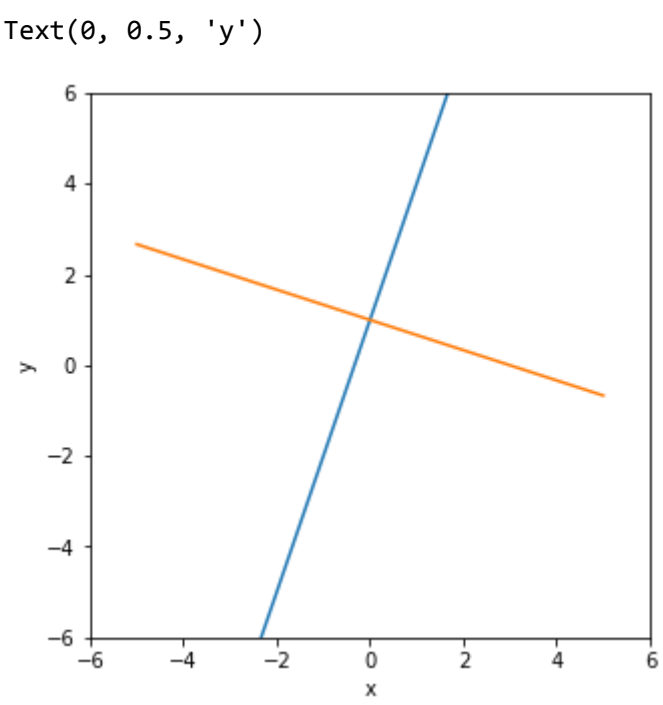


Ответ:
Причина того, что прямые не кажутся перпендикулярными, является то, что:

- для заданного интервала x и y искажён размер рисунка, высота и ширина;
- оси имеют разный интервалы, ось x: [-5, 5], а ось y:[-15, 15] (в общем случае, можно подобрать соответствующий размер рисунка).

B [4]:
```
plt.figure(figsize=(5,5))
x1, x2 = -6, 6
y1, y2 = -6, 6
plt.axis([x1, x2, y1, y2])

x = np.linspace(-5, 5, 21)
y = 3*x+1
y2 = (-1/3)*x+1
plt.plot(x,y)
plt.plot(x,y2)
plt.xlabel("x")
plt.ylabel("y")
```

Out[4]:
```
Text(0, 0.5, 'y')
```



3. Задание (в программе)

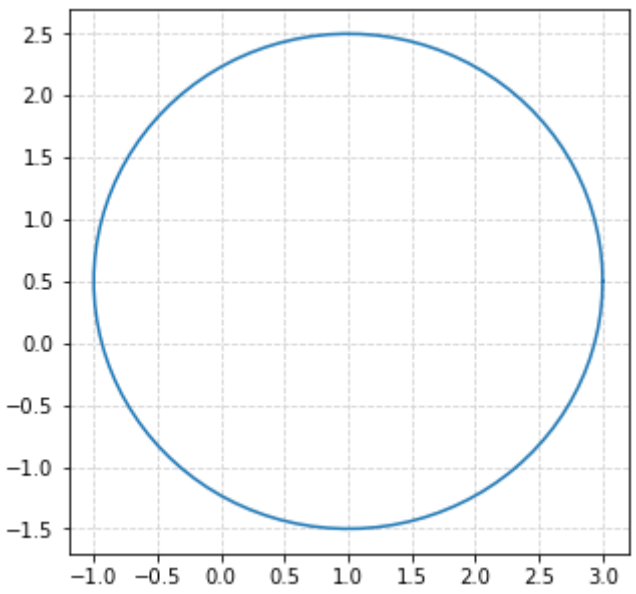
Напишите код на Python, реализующий построение графиков:

1. окружности,

2. эллипса,
3. гиперболы.

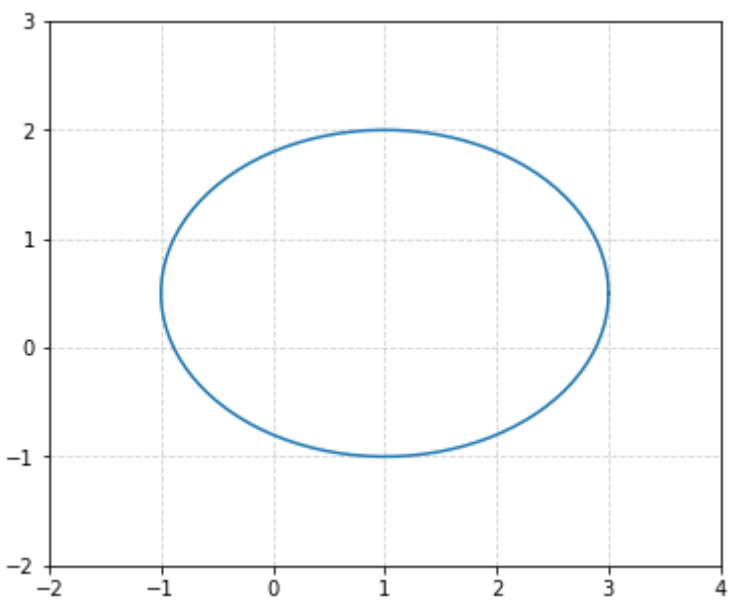
3.1 окружности

B [5]:
```
import numpy as np
from matplotlib import pyplot as plt
from math import pi, cos, sin

u=1.      #x-position of the center
v=0.5     #y-position of the center
a=2.      #radius on the x-axis
b=2.     #radius on the y-axis

plt.figure(figsize=(5,5))
t = np.linspace(0, 2*pi, 100)
plt.plot( u+a*np.cos(t) , v+b*np.sin(t) )
plt.grid(color='lightgray',linestyle='--')
plt.show()
```

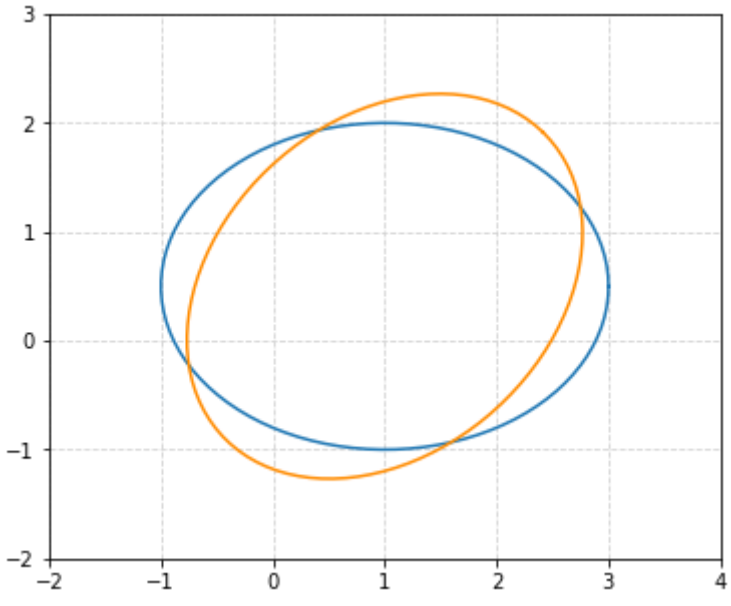


3.2 эллипса

B [6]:
```
u=1.      #x-position of the center
v=0.5     #y-position of the center
a=2.      #radius on the x-axis
b=1.5     #radius on the y-axis

plt.figure(figsize=(6,5))
x1, x2 = -2, 4
y1, y2 = -2, 3
plt.axis([x1, x2, y1, y2])

t = np.linspace(0, 2*pi, 100)
plt.plot( u+a*np.cos(t) , v+b*np.sin(t) )
plt.grid(color='lightgray',linestyle='--')
plt.show()
```



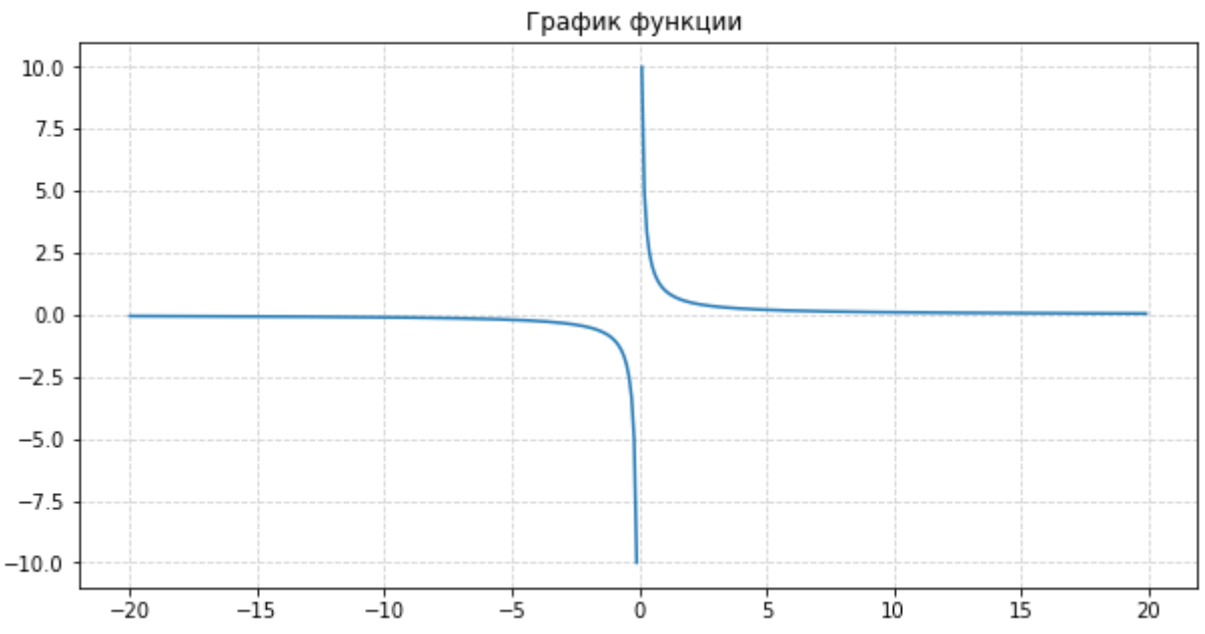
B [7]:
```
u=1.       #x-position of the center
v=0.5      #y-position of the center
a=2.       #radius on the x-axis
b=1.5      #radius on the y-axis
t_rot=pi/4 #rotation angle

plt.figure(figsize=(6,5))
x1, x2 = -2, 4
y1, y2 = -2, 3
plt.axis([x1, x2, y1, y2])

t = np.linspace(0, 2*pi, 100)
Ell = np.array([a*np.cos(t) , b*np.sin(t)])
     #u,v removed to keep the same center location
R_rot = np.array([[cos(t_rot) , -sin(t_rot)],[sin(t_rot) , cos(t_rot)]])
     #2-D rotation matrix

Ell_rot = np.zeros((2,Ell.shape[1]))
for i in range(Ell.shape[1]):
    Ell_rot[:,i] = np.dot(R_rot,Ell[:,i])

plt.plot( u+Ell[0,:] , v+Ell[1,:] )     #initial ellipse
plt.plot( u+Ell_rot[0,:] , v+Ell_rot[1,:],'darkorange' )    #rotated ellipse
plt.grid(color='lightgray',linestyle='--')
plt.show()
```



3. гиперболы.

B [8]:
```
import matplotlib.pyplot as plt
import decimal
import numpy as np

xmin = -20
xmax = 20
dx = 0.1

xlist = np.around(np.arange(xmin, xmax, dx), decimals=4)

ylist = 1 / xlist

plt.figure(figsize=(10,5))
plt.title('График функции ')
plt.plot(xlist, ylist)
plt.grid(color='lightgray',linestyle='--')
plt.show()
```

```
<ipython-input-8-7d45ad40ff17>:11: RuntimeWarning: divide by zero encountered in true_divide
  ylist = 1 / xlist
```



4. Задание (на листочке)

1) Пусть задана плоскость:

Ax + By + Cz + D = 0

Напишите уравнение плоскости, параллельной данной и проходящей через начало координат.

Решение

Чтобы плоскость проходила через начало координат (0,0,0), нужно, чтобы выполнялось A0+B0+C*0+D=0, т. е. надо взять D=0.

Ax + By + Cz = 0

2) Пусть задана плоскость: A1x + B1y + C1z + D1 = 0 и прямая x-x1/x2-x1 = y-y1/y2-y1 = z-z1/z2-z1

Как узнать, принадлежит прямая плоскости или нет?

Решение

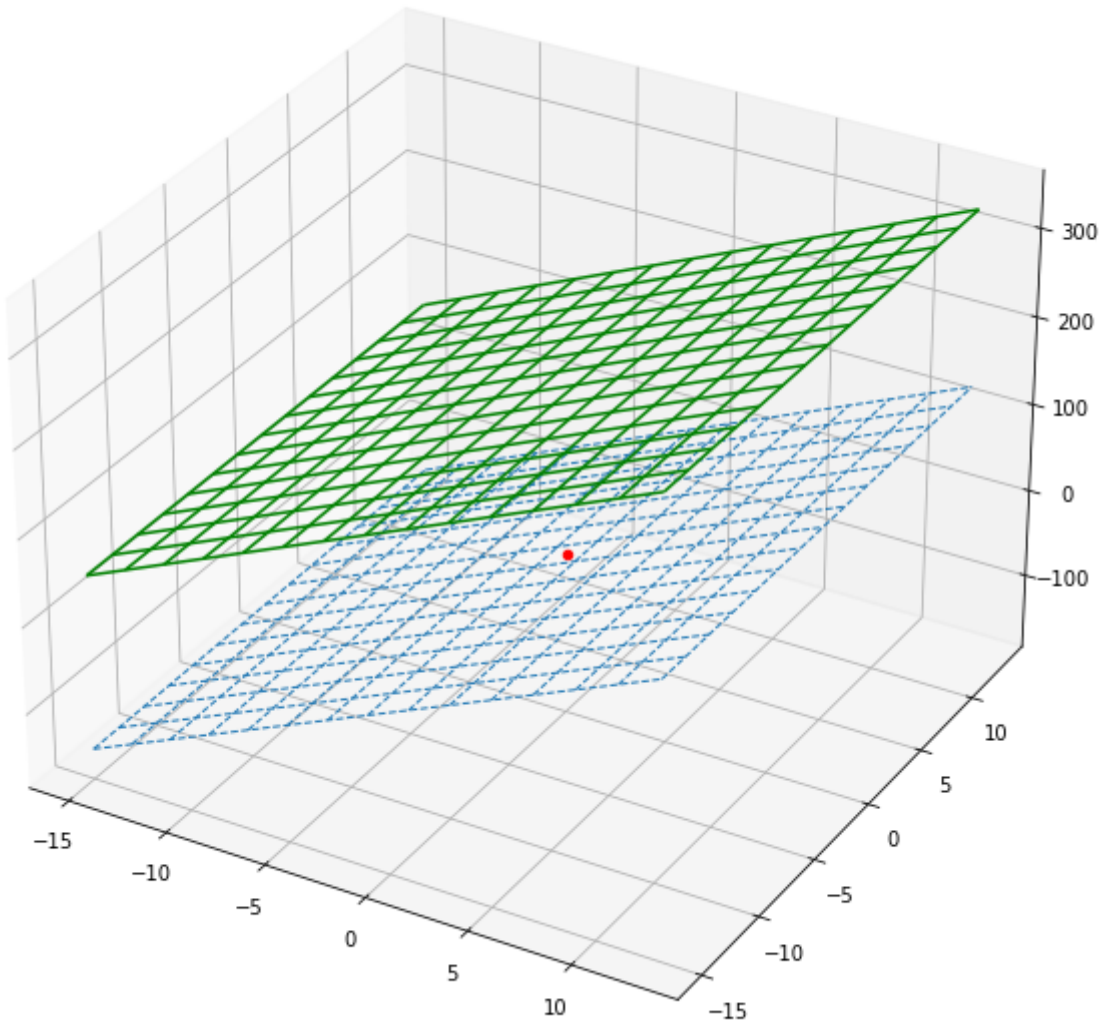
Имея уравнение прямой мы знаем координаты 2-х точек A(x, y, z) & A1(x1,y1,z1), принадлежащих этой прямой.
Возьмем координаты этих точек и подставим в уравнение плоскости.
Если при решении уравнения не выполняется равенство значит прямая не принадлежит этой плоскости.

5. Задание (в программе)

1) Нарисуйте трехмерный график двух параллельных плоскостей.
2) Нарисуйте трехмерный график двух любых поверхностей второго порядка.

1) Нарисуйте трехмерный график двух параллельных плоскостей.

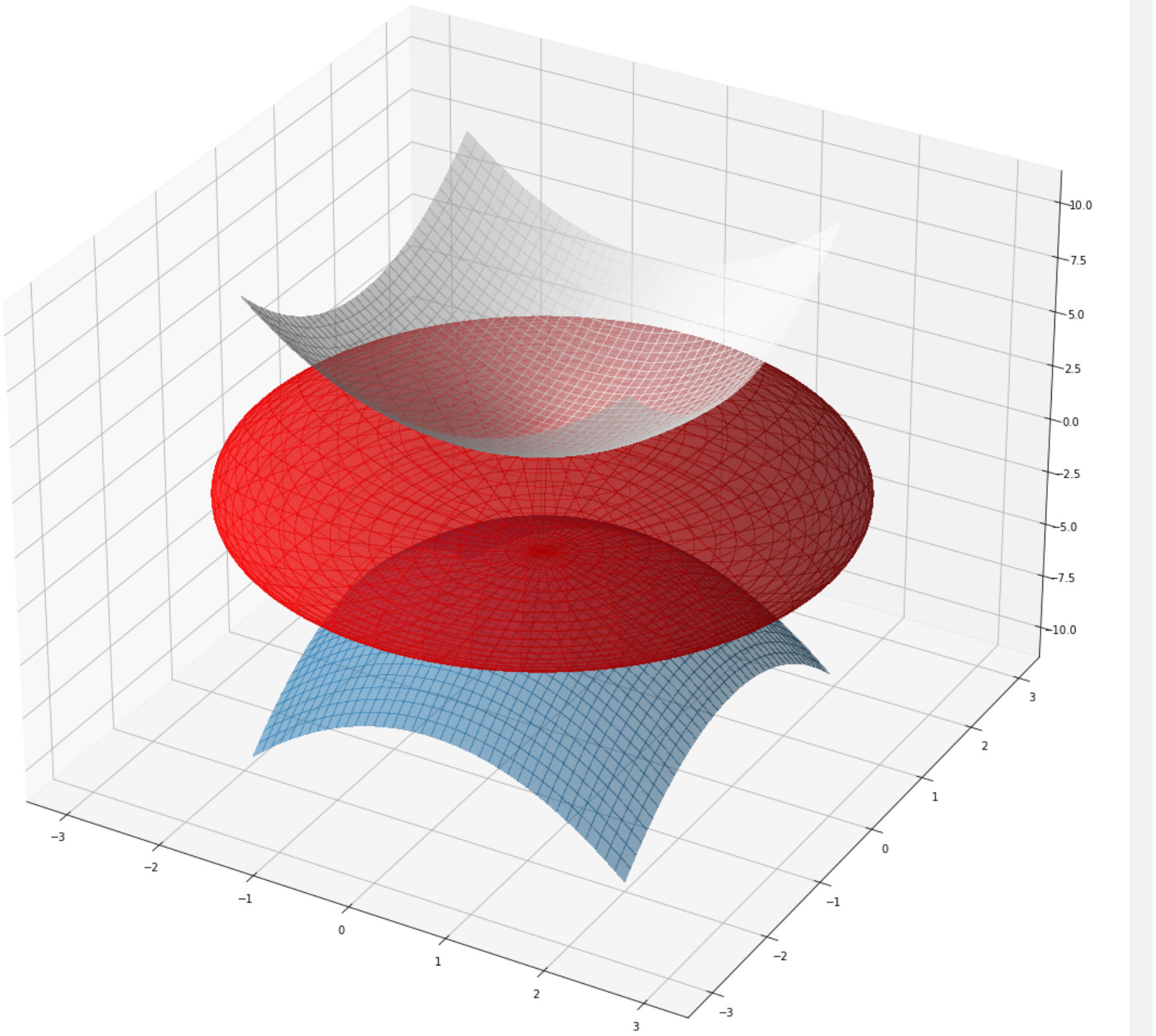
B [20]:
```
from mpl_toolkits.mplot3d import Axes3D

fig = plt.figure(figsize = (8, 8))
ax = Axes3D(fig)
X = np.arange(-15, 15, 2)
Y = np.arange(-15, 15, 2)

X, Y = np.meshgrid(X, Y)

# a, b, c1, c2 = 10, 5, 10, 200
a, b, c1, c2 = 10, 0, 0, 200
Z = a*X + b*Y + c1
Z2 = a*X + b*Y + c2

ax.plot_wireframe(X, Y, Z, linestyle='--', linewidth=1)  # первая плоскость
ax.plot_wireframe(X, Y, Z2,colors='green')  # вторая плоскость паралельная первой
ax.scatter(0, 0, 0,  color='red')  # Начало координат


plt.show()
```



2) Нарисуйте трехмерный график двух любых поверхностей второго порядка.

B [10]:
```
# fig = figure(figsize = (18, 18))
# ax = Axes3D(fig)

# a = 20
# b = 200
# angle = 40

# X, Y = np.meshgrid((np.arange(-15, 15, 2)), (np.arange(-15, 15, 2)))

# Z = b ** 2 + (b ** 2 * (X ** 2 / a ** 2))
# Z1 = -(b ** 2 + (b ** 2 * (X ** 2 / a ** 2)))
# Z2 = 2*a*X ** np.cos(angle) + np.sqrt(2*a*X ** np.cos(angle))
# ax.plot_surface(X, Y, Z,  linewidth=1, color='g')
# ax.plot_surface(X, Y, Z1,  linewidth=1, color='g')

# u = np.linspace(0,2*np.pi, 32)
# v = np.linspace(0, np.pi, 16)

# x = 10 * np.outer(np.cos(u), np.sin(v))
# y = 50 * np.outer(np.sin(u), np.sin(v))
# z = -100 * np.outer(np.ones(np.size(u)), np.cos(v))

# ax.plot_surface(x, y, z, rstride=4, cstride=4, color='r')


# show()
```

B [11]:
```
# Элипсоид вращения
phi = np.linspace(0,2*np.pi, 256).reshape(256, 1) # the angle of the projection in the xy-plane
theta = np.linspace(0, np.pi, 256).reshape(-1, 256) # the angle from the polar axis, ie the polar angle
radius = 3

# Transformation formulae for a spherical coordinate system.
x = radius*np.sin(theta)*np.cos(phi)
y = radius*np.sin(theta)*np.sin(phi)
z = radius*np.cos(theta)

# fig = plt.figure(figsize=plt.figaspect(1))  # Square figure
fig = plt.figure(figsize = (20, 20))

ax = fig.add_subplot(111, projection='3d')
ax.plot_surface(x, y, z,linewidth=0, antialiased=False, shade = True, color='red', alpha = 0.5)


# Параболоиды вращения
#set z values
x,y = np.mgrid[-2:2:0.1, -2:2:0.1]
z0 = x**2+y**2+radius

# rotate the samples by a radians around y
a = pi
# a = 0
t = np.transpose(np.array([x,y,z0]), (1,2,0))
m = [[cos(a), 0, sin(a)],[0,1,0],[-sin(a), 0, cos(a)]]
x,y,z = np.transpose(np.dot(t, m), (2,0,1))
# or `np.dot(t, m)` instead `t @ m`

ax.plot_surface(x,y,z,linewidth=1, antialiased=False, shade = True, alpha = 0.5)
ax.plot_surface(x,y,-z,linewidth=1, antialiased=False, shade = True, alpha = 0.5, color='w')


plt.show()
```



Задание 33

0. Задание (сделайте себе шпаргалку перед глазами, если не помните) - не присылать¶

Чему равны синус, косинус, тангенс перечисленных углов? Запишите значения в таблицу:

угол	sin	cos	tg
30	1/2	√3/2	1/√3
45	1/√2	1/√2	1
60	√3/2	1/2	√3
90	1	0	-
180	0	-1	0

1. Задание (в программе)

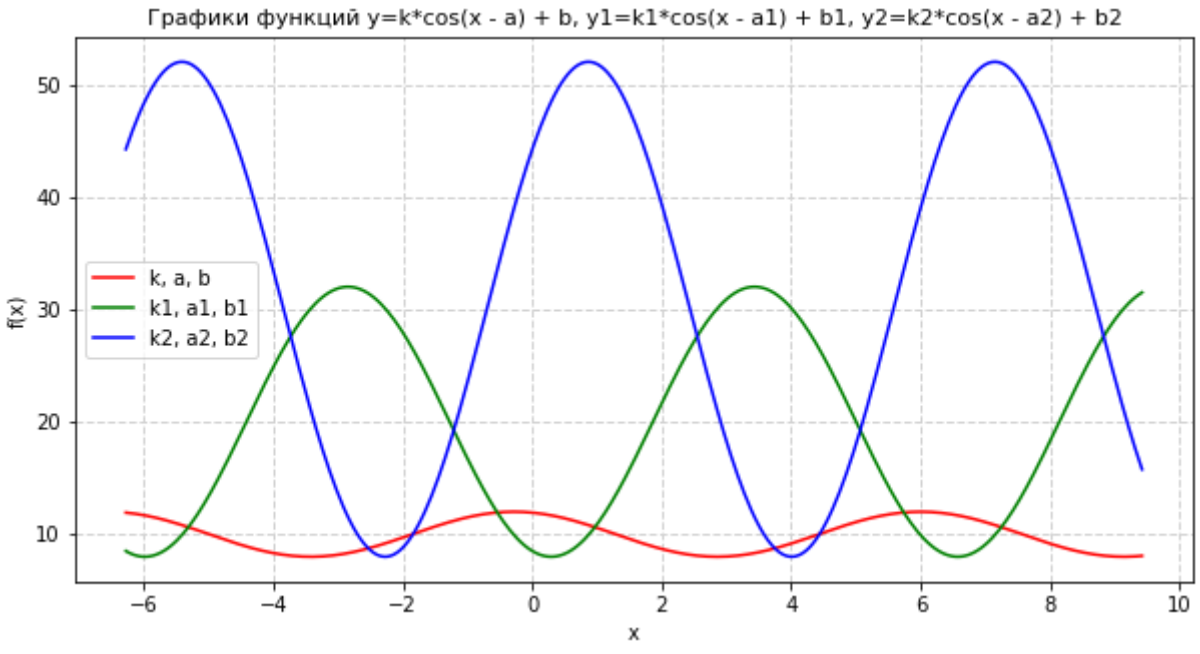
Нарисуйте график функции: для некоторых (2-3 различных) значений параметров k, a, b

$$y(x) = k·cos(x − a) + b$$

для некоторых (2-3 различных) значений параметров k, a, b

B [52]:
```
# fig, ax = plt.subplots()

x = np.linspace(-2*np.pi, 3*np.pi, num = 300)

k, a, b = np.linspace(2, 10, num = 3)
k1, a1, b1 = np.linspace(12, 20, num = 3)
k2, a2, b2 = np.linspace(22, 30, num = 3)


print(f'k={k}, a={a}, b={b}')
print(f'k1={k1}, a1={a1}, b1={b1}')
print(f'k2={k2}, a2={a2}, b2={b2}')

plt.figure(figsize = (10, 5))
plt.plot(x, k * np.cos(x - a) + b, color='r', label='k, a, b')
plt.plot(x, k1 * np.cos(x - a1) + b1, color='g', label='k1, a1, b1')
plt.plot(x, k2 * np.cos(x - a2) + b2, color='b', label='k2, a2, b2')

plt.xlabel('x')
plt.ylabel('f(x)')
# plt.grid(True)
plt.grid(color='lightgray',linestyle='--', linewidth=1)
plt.legend(frameon=True)

# ax.set_title('Графики трёх функций y=k*cos(x - a) + b')

figure_title='Графики функций y=k*cos(x - a) + b, y1=k1*cos(x - a1) + b1, y2=k2*cos(x - a2) + b2'
plt.title(figure_title, fontsize = 11)


plt.show()
```

```
k=2.0, a=6.0, b=10.0
k1=12.0, a1=16.0, b1=20.0
k2=22.0, a2=26.0, b2=30.0
```



2. Задание

Докажите, что при ортогональном преобразовании сохраняется расстояние между точками.

Пусть точки M1(x1, y1) и M2(x2, y2) посредством ортогонального преобразования, переводятся соответственно в точки M'1 (x'1, y'1) и M'2(x'2, y'2).
Требуется доказать, что отрезки M1M2 и M'1M'2 имеют равные длины.

Доказательство:

$$|M_1'M_2'|^2 = [x_1' - x_2']^2 + [y_1' - y_2']^2 = [a_{11}(x_2 - x_1) + a_{12}(y_2 - y_1)]^2 + [a_{21}(x_2 - x_1) + a_{22}(y_2 - y_1)]^2 =$$
$$= (a_{11}^2 + a_{21}^2)(x_2 - x_1)^2 + 2(a_{11}a_{12} + a_{21}a_{22})(x_2 - x_1)(y_2 - y_1) + (a_{12}^2 + a_{22}^2)(y_2 - y_1)^2 =$$
$$= (x_2 - x_1)^2 + (y_2 - y_1)^2 = |M_1M_2|^2$$

Таким образом доказано, что $|M_1'M_2'|^2 = |M_1M_2|^2$

3. Задание (в программе)

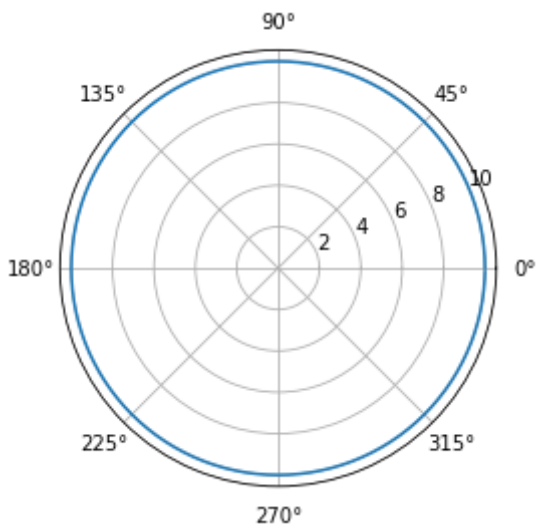
3.1 Напишите код, который будет переводить полярные координаты в декартовы.

B [54]:
```
def polar_to_decart(r, phi):
    x = r*math.cos(phi)
    y = r*math.sin(phi)
    return x,y

polar_to_decart(10,75)
```

Out[54]: (9.217512697247493, -3.8778163540943043)

3.2 Напишите код, который будет рисовать график окружности в полярных координатах.

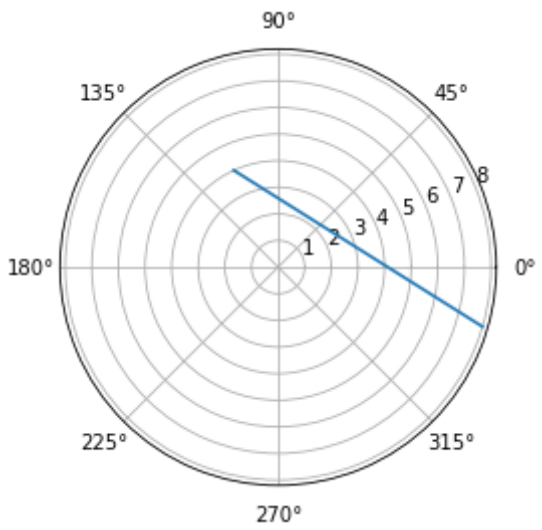
B [58]:
```
# arange([start,] stop[, step,], dtype=None) - Return evenly spaced values within a given interval.
# polar(theta, r, **kwargs) - Make a polar plot.

phi = np.arange(0., 2., 1./180.)*np.pi
plt.polar(phi, [10]*len(phi))

plt.show()
```



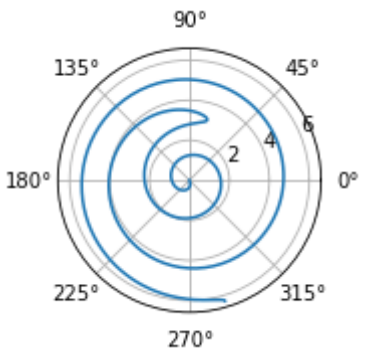
3.3 Напишите код, который будет рисовать график отрезка прямой линии в полярных координатах.

B [77]:
```
phi = np.arange(2, 10, 4)
#phi = [4, 6]
print(phi)
rho = np.arange(4, 10, 4)
# rho = [2, 6]
print(rho)

plt.polar(phi, rho)

plt.show()
```

```
[2 6]
[4 8]
```



B [74]:
```
# rpo = 5 * cos (phi)
fig = matplotlib.pyplot.figure(figsize=(8., 6.))

phi = numpy.arange(0,2*numpy.pi,0.01)
rho = 5 * numpy.cos(phi)
# phi = np.arange(0, 10, 0.1)
# rho = 2*phi + 5

ax1 = fig.add_subplot(232, projection='polar')
ax1.plot(rho, phi)
ax1.grid(True)

matplotlib.pyplot.show()
```



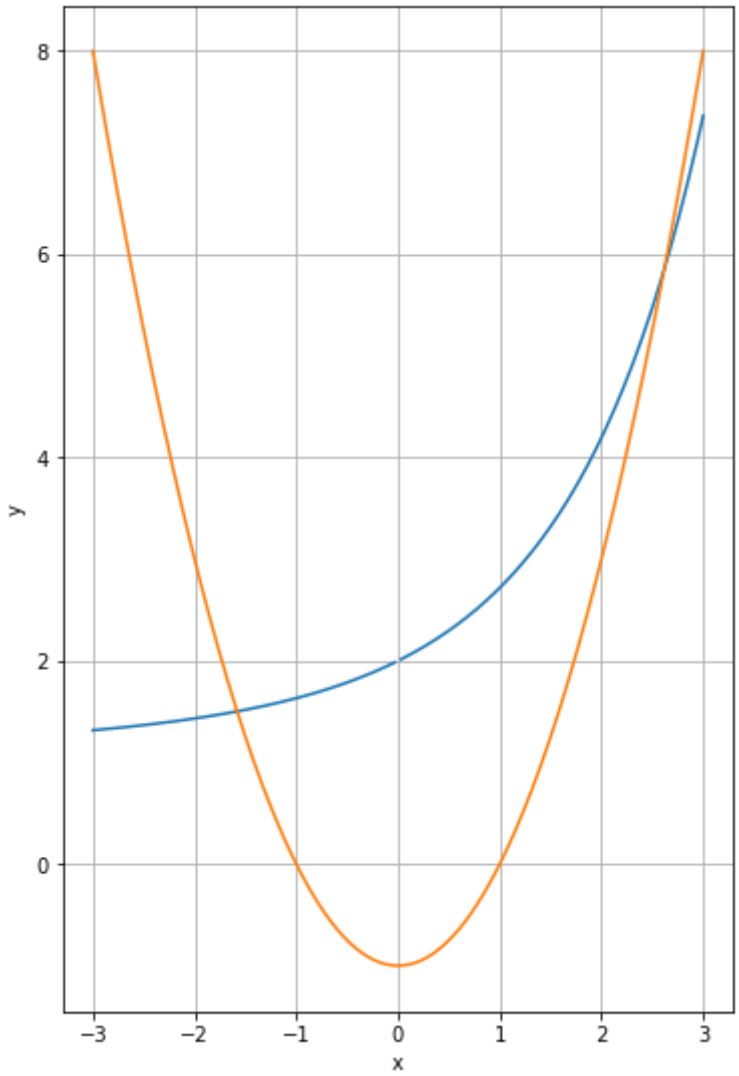
4. Задание (в программе)

```
Решите систему уравнений:

$y = x^2 - 1$<br>
$e^x + x(1 - y) = 1$
```

$y = x^2 - 1$
$y = (e^x + x - 1)/x$

B [90]:
```
x = np.linspace(-3, 3, 201)

plt.figure(figsize = (6, 12))

plt.plot(x, (np.exp(x)+ x - 1)/x)
plt.plot(x, x**2 - 1)
plt.xlabel('x')
plt.ylabel('y')
#plt.ylim(-1,8)
plt.grid(True)
plt.axis('scaled')
plt.show()

from scipy.optimize import fsolve

def equations(p):
    x, y = p
    return (x**2 - 1 - y, np.exp(x) + x * (1 - y) - 1)

x1, y1 =  fsolve(equations, (2, 3))
x2, y2 =  fsolve(equations, (-2, -1))

print (x1, y1)
print (x2, y2)
```

```
<ipython-input-90-e67748700c64>:5: RuntimeWarning: invalid value encountered in true_divide
  plt.plot(x, (np.exp(x)+ x - 1)/x)
```




```
2.618145573085453 5.854686241866956
-1.5818353529296008 1.5022030836866855
```

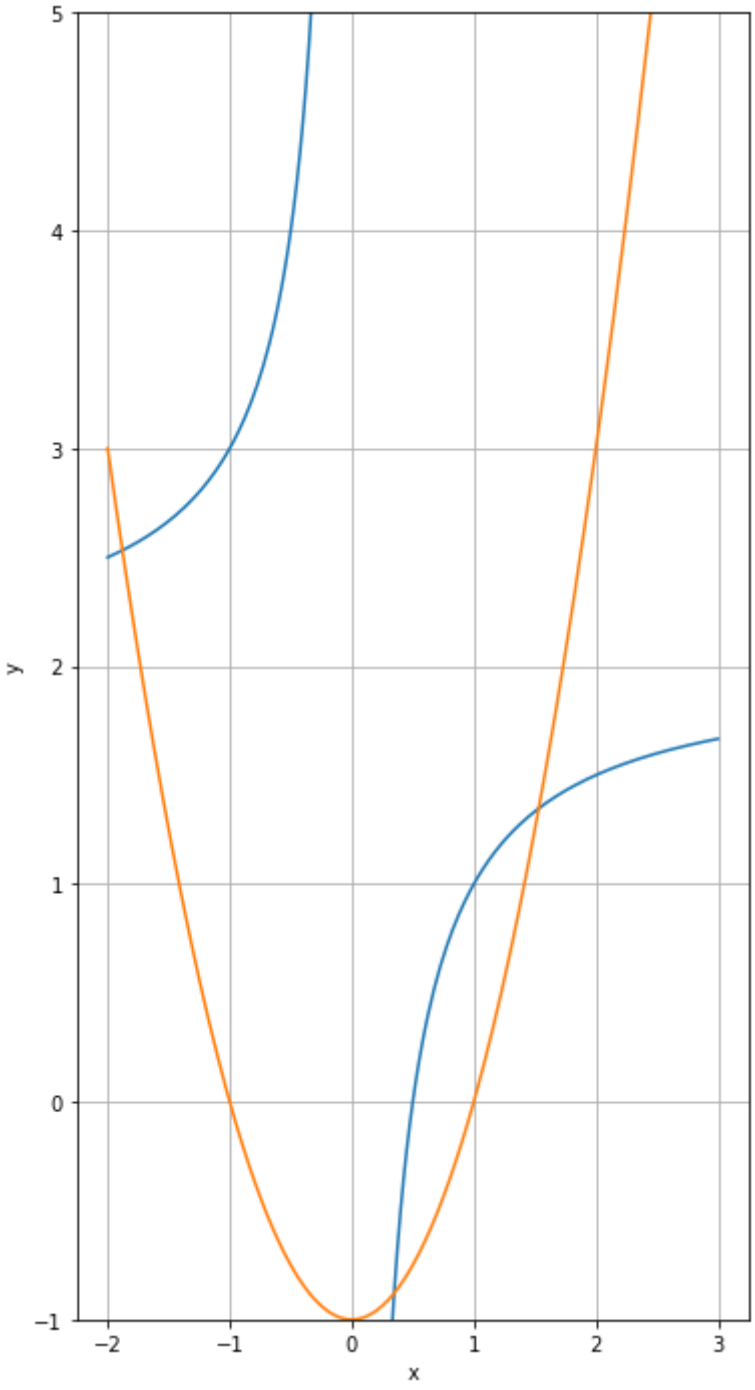
Решите систему уравнений и неравенств:

$y = x^2 – 1$
$e^x + x(1 – y) > 0$

$y = x^2 – 1$
$y = -\frac{1}{x} + 2$

B [94]:
```
x = np.linspace(-2, 3, 201)

plt.figure(figsize = (6, 12))
plt.plot(x, (- 1/x)+2)
plt.plot(x, x**2 - 1)
plt.xlabel('x')
plt.ylabel('y')
plt.ylim(-1,5)
plt.grid(True)

plt.show()

from scipy.optimize import fsolve

def equations(p):
    x, y = p
    return (x**2 - 1 - y, (- 1/x)+2 - y)

x1, y1 =  fsolve(equations, (1.2, 4))
x2, y2 =  fsolve(equations, (0.5, 0.5))
x3, y3 =  fsolve(equations, (-2, -1))

print (x1, y1)
print (x2, y2)
print (x3, y3)
```

```
<ipython-input-94-63a6f79434db>:4: RuntimeWarning: divide by zero encountered in true_divide
  plt.plot(x, (- 1/x)+2)
```




```
1.5320888862170892 1.3472963553093682
```

```
0.3472963553346955 -0.8793852415708359
-1.8793852415717947 2.5320888862379767
```

B [ ]: