# Лабораторный практикум 10. Приведение квадратичной формы к каноническому виду ортогональным преобразованием переменных.

Загрузите необходимые библиотеки:

>> import numpy as np

>> from sympy import \*

>> import matplotlib.pyplot as plt

%matplotlib widget

## Приведение матрицы линейного оператора к диагональному виду

В разных базисах линейному оператору соответствуют различные матрицы. Наиболее простой – диагональный вид принимает матрица линейного оператора *А* в базисе, составленном из собственных векторов оператора *А*.

Пусть имеются два базиса пространства: старый и новый, и матрица перехода *Т* от старого базиса к новому. Приведением матрицы *А* к диагональному виду называется нахождение такой матрицы перехода *Т*, для которой матрица линейного оператора *А* в новом базисе имеет диагональный вид: .

Для того, чтобы квадратная матрица *А* размера приводилась к диагональному виду необходимо и достаточно, чтобы она имела *n* линейно независимых векторов.

**Пример 1.** В некотором базисе линейный оператор задан матрицей . В действительном линейном пространстве найти базис, в котором матрица этого оператора имеет диагональный вид.

Матрица имеет диагональный вид в базисе из собственных векторов, поэтому найдем её собственные векторы.

Характеристическое уравнение: , его корни .

Если , то Каждому собственному значению соответствует бесконечное множество векторов , но все они будут коллинеарны между собой (то есть отличаются друг от друга некоторым постоянным множителем *с*). Отсюда собственный вектор . Аналогично для , получим собственный вектор . Таким образом, матрица *А* в базисе : .

В Python функция ***eig*()** модуля numpy.linalg находит собственные векторы и собственные значения заданной матрицы: ***numpy.linalg.eig*(*a*)**. Возвращает список из двух элементов: первый элемент списка - массив, содержащий собственные значения, второй элемент списка – матрица перехода, столбцы которой содержат соответствующие собственные векторы (собственные векторы возвращаются нормированными). Найдем эти числа и векторы для матрицы примера 1:

>> A = np.array([[1,3],[4,2]])

>> L,T=np.linalg.eig(А)

(array([-2., 5.]),

array([[-0.70710678, -0.6],

[ 0.70710678, -0.8]]))

Собственные векторы **пронормированы**!

Проверим выполнение равенства :

D=np.linalg.inv(T).dot(A).dot(T)

array([[-2.00000000e+00, 1.11022302e-16],

[ 4.44089210e-16, 5.00000000e+00]])

Метод ***.diagonalize*()** библиотеки ***Sympy*** возвращает кортеж (*Т, D*), где *Т* - матрица перехода (векторы не нормированы!), *D* - диагональная матрица:

**AA = Matrix([[1,3],[4,2]])**

**Т, D = AA.diagonalize()**

**Упражнение 10.1.** Найти матрицу перехода *T*, которая приводит матрицу к диагональному виду. Сделать проверку: .

## Приведение квадратичной формы к диагональному виду

Задача приведения квадратичной формы к диагональному виду заключается в следующем: требуется найти в векторном пространстве такой базис, в котором квадратичная форма будет иметь наиболее простой вид, называемый каноническим. Такой базис всегда существует. Матрица квадратичной формы в этом базисе имеет диагональный вид.

**Пример 2.** Привести к каноническому виду квадратичную форму .

Матрица квадратичной формы . Ее собственные числа и **нормированные** собственные векторы: , .Очевидно, что данные векторы составляют **ортонормированный базис** (проверьте). Матрица ортогонального преобразования имеет вид

.

Отсюда формулы преобразования координат :

.

После преобразования квадратичной формы получим ее канонический вид в новой системе координат :

A = Matrix([[3,5],[5,3]])

T,D=P.diagonalize()

print(T)

print(D)

x,y=symbols('x y')

f=D[0,0]\*x\*\*2+D[1,1]\*y\*\*2

f

**Обратите внимание, что полученные собственные векторы ортогональны.**

**Упражнение 10.2.** Привести к каноническому виду квадратичную форму .

## Упрощение уравнений фигур второго порядка

Характеристической квадратичной формой кривой второго порядка называется функция . По характеру квадратичной формы можно сделать заключения о виде кривой. Если квадратичная форма положительно определенная – это эллипс; если неопределенная – это гипербола; если полуопределенная (принимает только неотрицательные или только неположительные значения) – это парабола; если отрицательно определенная – это мнимый эллипс.

**Пример 3.** Определить тип кривой второго порядка . Найти каноническое уравнение и записать формулы перехода к новым переменным. Построить кривую в новой системе координат.

Квадратичная форма данной кривой является неопределенной (проверьте!). Значит, это гипербола. Найдем формулы перехода к новым переменным:

x1,y1=symbols('x1 y1')

X1=Matrix([[x1,y1]])

X=T/sqrt(2)\*X1.T # матрица Т - матрица перехода из примера 2

print('x=',simplify(X[0,0]))

print('y=',simplify(X[1,0]))

F=3\*X[0]\*\*2+10\*X[0]\*X[1]+3\*X[1]\*\*2-32

F=simplify(F/32)

F

Почему элементы матрицы *Т* делятся на ?

Каноническое уравнение гиперболы: . Изобразим ее в старой и новой системах координат. Для этого импортируем функцию *plot\_implicit* из пакета *sympy.plotting*:

var('x y')

plot\_implicit(Eq(3\*x\*\*2+10\*x\*y+3\*y\*\*2,32),title='В старой СК',

xlabel='x',ylabel='y')

plot\_implicit(Eq(-x\*\*2.16+y\*\*2.4,1),title='В новой СК',

xlabel='x1',ylabel='y1')

В старой системе координат гипербола повернута относительно координатных осей.

После поворота для получения канонического уравнения иногда нужно еще совершить сдвиг. Формулы соответствующей замены строятся при помощи **выделения полного квадрата**.

**Упражнение 10.3.** Определить тип кривой . Найти каноническое уравнение и записать формулы перехода к новым переменным. Построить кривую в новой системе координат.

**Упражнение 10.4.** Определить тип кривой . Найти каноническое уравнение и записать формулы перехода к новым переменным. Построить кривую в старой и новой системах координат.

**Пример 4.** Какую поверхность определяет уравнение?

x1,y1,z1=symbols('x1 y1 z1')

XX=Matrix([[x1,y1,z1]])

# Матрица Т1 - матрица перехода из упр.10.2

# Собственные векторы длины 3 ортогональны, но не ортонормированы!

X=T1/3\*XX.T

print(D1)

print('x=',simplify(X[0,0]))

print('y=',simplify(X[1,0]))

print('z=',simplify(X[2,0]))

H=6\*X[0]\*\*2-4\*X[0]\*X[1]+5\*X[1]\*\*2+7\*X[2]\*\*2+4\*X[0]\*X[2]-18

H=simplify(H/18)

H

Получили каноническое уравнение эллипсоида . Построение поверхности:

# Построение эллипсоида в сферических координатах

fig = plt.figure(figsize=(7, 7)) # создаём холст

ax = fig.add\_subplot(111, projection='3d')

coefs = (6, 3, 2)# коэффициенты уравнения

rx, ry, rz = 1/np.sqrt(coefs) # радиусы

# Сферические углы:

u = np.linspace(0, 2\*np.pi, 100)

v = np.linspace(0, np.pi, 100)

# Уравнение эллипсоида:

x = rx\*np.outer(np.cos(u), np.sin(v))

y = ry\*np.outer(np.sin(u), np.sin(v))

z = rz\*np.outer(np.ones\_like(u), np.cos(v))

ax.plot\_surface(x,y,z, rstride=5, cstride=5, color='m')

plt.xlabel('x')

plt.ylabel('y')

plt.title('Эллипсоид')

plt.show())

**Упражнение 10.5.** Определить тип поверхности и построить ее в новой системе координат.

Не забудьте убедиться в том, что новый базис является ортонормированным. В противном случае необходимо применить процесс ортогонализации.

## Дополнительное задание

Написать программу процедуры выделения полного квадрата в уравнении и с ее помощью определить тип поверхности . Построить эту поверхность.