Доля Петро Григорович
Харківський національний університет імені В. Н. Каразіна
факультет математики і інформатики
кафедра теоретичної та прикладної інформатики
2025 р.

Аналітичні методи геометричного моделювання з Python.

Огляд методів моделювання складених функцій, кривих і поверхонь.

Зміст

10. Моделювання складених функцій, кривих і поверхонь.

Складені криві і поверхні зазвичай моделюються за допомогою шматкових функцій, тобто таких, що задані різними формулами на різних ділянках області визначення. Для їх конструювання в модулі `numpy` існує функція `piecewise()`, яка доволі громіздка і повільна. Для неперервних функцій зручнішим інколи може бути єдиний «аналітичний» вираз, який можна побудувати з використанням функції абсолютного значення. Тут ми розглянемо лише кілька окремих випадків.

10.1. Рівняння ламаних.

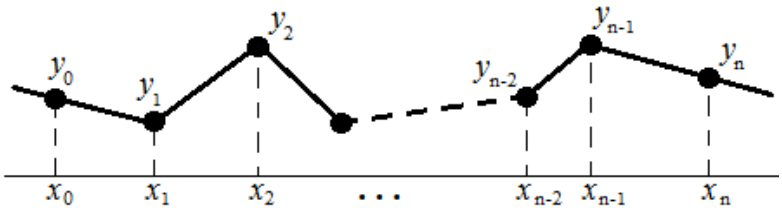
Формула Бернштейна ламаної. Розглянемо рівняння

$$y(x) = a_0 + a_1 x + \sum_{i=1}^{n-1} c_i |x - x_i| \qquad (1)$$

Воно представляє неперервну функцію і на кожній ділянці $x < x_1$, $x_{i-1} < x < x_i$, $(i = 2, \dots, n-1)$, $x > x_{n-1}$ є рівнянням деякої прямої. Тому (1) визначає ламану. Знайдемо рівняння такої, у якої відомі координати вузлів.

Нехай дано набір точок $\{(x_k, y_k)\}_{k=0}^{n}$, $x_0 < x_1 < \dots < x_n$. З'єднаємо їх послідовно прямолінійними відрізками. Точки $\{(x_k, y_k)\}_{k=1}^{n-1}$ будуть вершинами

(точками зламу) ламаною. Перша пара точок $(x_0, y_0), (x_1, y_1)$ для $x < x_1$ визначатиме промінь ламаної, що проходить через них. Останні дві точки $(x_{n-1}, y_{n-1}), (x_n, y_n)$ визначатимуть для всіх $x > x_{n-1}$ інший промінь ламаної. Тобто в точках (x_0, y_0), (x_n, y_n) зламів немає.



Знайдемо рівняння ламаної, що визначається таким набором точок. Для цього в (1) потрібно підібрати коефіцієнти $a_0, a_1, c_1, \ldots, c_{n-1}$. Їх рівно $n+1$ і ми маємо $n+1$ умову проходження кривої через точки $\{(x_k, y_k)\}_{k=0}^{n}$. В [2] показано, що це рівняння можна записати у вигляді

$$y = \frac{1}{2}\left(y_0 + \frac{y_1 - y_0}{x_1 - x_0}(x - x_0) + y_n + \frac{y_n - y_{n-1}}{x_n - x_{n-1}}(x - x_n)\right) + \\ + \frac{1}{2}\sum_{k=1}^{n-1}\left(\frac{y_{k+1} - y_k}{x_{k+1} - x_k} - \frac{y_k - y_{k-1}}{x_k - x_{k-1}}\right)|x - x_k|\,. \qquad (2)$$

Рівняння (2) носить ім'я С.Н. Бернштейн, який вперше застосував його аналог для випадку, коли точки $x_0, x_1, \ldots, x_{n-1}, x_n$ рівномірно розподілені вздовж осі абсцис [1].

Реалізуємо формулу (2), назвавши відповідну функцію `broken_line()`. Вона буде повертати символьний вираз (2), який залежить від символьної змінної, ім'я якої слід повідомляти функції в якості, наприклад, першого аргументу. Іншими її аргументами будуть масиви або списки X,Y абсцис і ординат вузлів ламаної включно з точками (x_0, y_0), (x_n, y_n), які не є точками зламу.

Робочий код функції обмежимо блоком `try`, щоб будь-яка помилка входових даних викликала виключну ситуацію і оброблялась власним блоком `except`. Зокрема масиви X,Y повинні бути однакової довжини і містити не менше двох точок. Серед елементів масиву X не повинно бути однакових значень і вони повинні йти у зростаючому порядку. Останнє не викликатиме виключної ситуації, але ми штучно її генеруватимемо, щоб вона оброблялась у тому ж блоці `except`. Можливі і інші помилки входових даних. Отже, якщо виключні ситуації виникатимуть, то вони оброблятимуться у блоці `except` нашої функції, в якому для спрощення лише друкуватиметься повідомлення і повертатиметься `None`.

```
from sympy import symbols,simplify,Abs,lambdify
import numpy as np
def broken_line(x,X,Y):
    """ Символьне рівняння ламаної від символьної змінної x.
    X,Y - масиви абсцис і ординат вузлів ламаної.
    Перша (X[0],Y[0]) і остання (X[-1],Y[-1]) точки не є точками
    зламу,а визначають кінцеві промені ламаної.
```

```
    Абсциси X[i] повинні монотонно зростати і бути різними.
    Довжини масивів X і Y однакові, і містять принаймні
    по 2 елементи.    """
   try:
        Xin=X[1:-1]
        dx=np.diff(X)
        if np.any(dx<=0): raise  # перевірка монотонності
        dy=np.diff(Y)
        dyx=dy/dx
        k=np.diff(dyx)
        p1=Y[0]+dyx[0]*(x-X[0])
        pn=Y[-1]+dyx[-1]*(x-X[-1])
        s0=simplify((p1+pn)/2)
        sn=sum([k[i]*Abs(x-Xin[i]) for i in range(len(Xin))])/2
        return s0+sn
   except:
        print("Помилка входових даних!")
        return None
```

Збережіть функцію в модулі broken.py.

Приклад 1. Генерувати рівняння ламаної, яка проходить через точки `(-2,0)`, `(-1,1)`, `(0,0)`, `(1,1)`, `(2,0)`, `(3,1)`, і побудувати її графік.

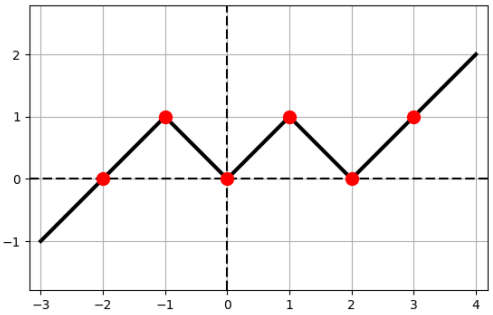
Розв'язання. Зробимо це в тому ж файлі broken.py.

```
if __name__ == '__main__':
    import matplotlib.pyplot as plt
    import sys
    plt.close('all')
    # генерування рівняння ламаної
    X=np.array([-2,-1,0,1,2,3])
    Y=np.array([0,1,0,1,0,1])
    x = symbols('x')
    fbl=broken_line(x,X,Y);
    if fbl is None:
        sys.exit()
    else:
        print(fbl)
   # побудова графіка ламаної і візуалізація входових точок
    fbline=lambdify(x,fbl,"numpy")
    fig,ax = plt.subplots(1,1)
    ax.axhline(0,color='black',dashes=(5,2))
    ax.axvline(0,color='black',dashes=(5,2))
    x_=np.linspace(-3,4,71)
    y_=fbline(x_)
    ax.plot(x_,y_,'-k',lw=3)
    ax.plot(X,Y,'or',ms=10)
    ax.axis('equal')
    ax.grid(True)
```

Рівняння ламаної буде надруковано в наступному вигляді.

```
1.0*x+Abs(x)+Abs(x-2)-1.0*Abs(x-1)-1.0*Abs(x+1)
```

А візуалізація створить таке зображення.



■

В попередньому прикладі для побудови графіка ламаної достатньо інструкції `ax.plot(X,Y,'-or',ms=10)`. Але рівняння дозволяє обчислювати значення функції в проміжних точках, що в деяких випадках дуже корисно. Щоб пояснити останнє, розглянемо наступну задачу.

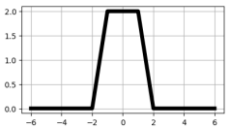
Приклад 2. Записати і візуалізувати розв'язок Д'аламбера на нескінченній прямій одновимірного хвильового рівняння $u''_{tt} = a^2 u''_{xx}$ $(t > 0, -\infty < x < \infty)$ з початковими умовами $u(x,0) = f(x)$, $u'_t(x,t) = 0$.

Розв'язання. Розв'язок Д'аламбера цієї задачі має вигляд

$$u(x,t) = \frac{1}{2}\left(f(x-t) + f(x+t)\right)$$

Він визначається функцією $f(x)$, яка може бути ламаною, заданою координатами своїх вершин. Побудувати графіки розв'язку $u(x,t_i)$ у різні моменти часу t_i, використовуючи лише вузли початкової ламаної $f(x)$, без спеціальних «трюків» не виходить. Але тепер ми можемо записати «аналітичне» рівняння ламаної $f(x)$, і в формулі Даламбера використовувати його.

```
import numpy as np
import matplotlib.pyplot as plt
from sympy import symbols,lambdify
from broken import broken_line
import sys
plt.close('all')
```

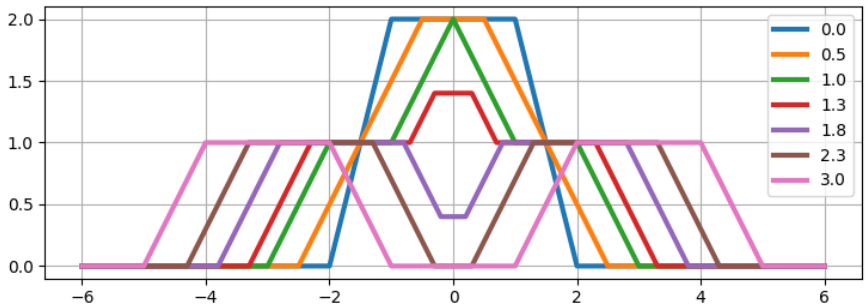


```
# вузли початкової ламаної
X=np.array([-3,-2,-1,1,2,3])
F=np.array([0,0,2,2,0,0])
```

Зверніть увагу, що для задання лівого та правого горизонтальних променів $y = 0$ ми використали по дві точки з нульовими ординатами `{-3,0},{-2,0}` та `{2,0}, {3,0}`.

```
# побудова рівняння початкової ламаної
x = symbols('x')
fbl=broken_line(x,X,F);
if fbl is None: sys.exit()
else:  print(fbl)
# створення розв'язку Даламбера
fbline=lambdify(x,fbl,"numpy")
def u(x,t):
    return (fbline(x-t)+fbline(x+t))/2
# графіки розв'язку u(x,t) в різні моменти часу
T=np.array([0,0.5,1,1.3,1.8,2.3,3])   # моменти часу
```

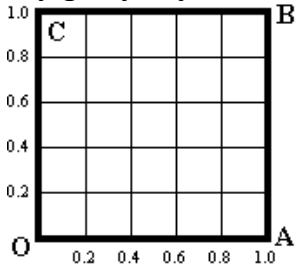
```
x_=np.linspace(-6,6,121)
fig,ax = plt.subplots(1,1)
U=np.array(list(map(lambda t: u(x_,t),T))).T
pl=ax.plot(x_,U,lw=3)
ax.legend(pl,T)
ax.grid(True)
```



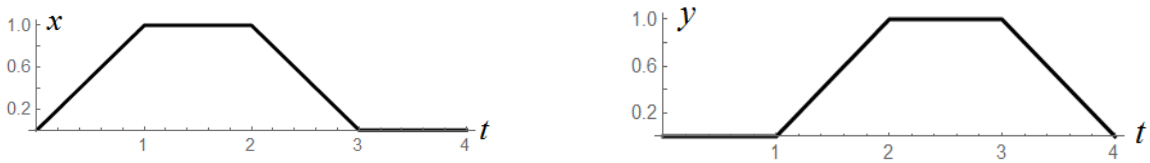
Зауваження. В Python існують інші конструкції, за допомогою яких можна обчислювати значення «ламаних» функцій. Наприклад, це `numpy.piecewise()` або `scipy.interpolate.interp1d()`. Вони теж підходять для розв'язання розглянутої задачі.

■

Параметричні рівняння ламаних. Проаналізуємо задачу побудови параметричних рівнянь ламаних на прикладі конструювання рівняння квадрата OABC, показаного на наступному рисунку.



Припустимо, що ми склали його параметричне рівняння $\boldsymbol{r}(t) = (x(t), y(t))$ таке, що параметр t лінійно зростає вздовж контура. Наприклад, нехай на відрізку OA параметр t лінійно зростає від 0 до 1, на відрізку AB від 1 до 2, на відрізку BC від 2 до 3, на відрізку CO від 3 до 4. Тоді зрозуміло, що координатна функція $x(t)$ на ділянці $0 \le t \le 1$ лінійно зростатиме від 0 до 1, а координатна функція $y(t)$ дорівнюватиме 0. На ділянці $1 \le t \le 2$ функція $x(t)$ буде сталою (дорівнюватиме 1), а функція $y(t)$ лінійно зростатиме від 0 до 1. При $2 \le t \le 3$ функція $x(t)$ буде лінійно спадати до 0, а $y(t) = 1$. При $3 \le t \le 4$ функція $y(t)$ буде лінійно зменшуватися від 1 до 0, а $x(t) = 0$. Оскільки для побудови ламаної буде використовуватися діапазон $0 \le t \le 4$, то поведінка функцій поза цим відрізком не матиме значення. Проведений аналіз показує, що графіки функцій $x(t)$ і $y(t)$ на відрізку $0 \le t \le 4$ можуть мати наступний вигляд



Можливі графіки функцій $x(t)$ (ліворуч) та $y(t)$ (праворуч).

Це ламані, рівняння яких можна генерувати за допомогою викликів `broken_line(t,T,X)` і `broken_line(t,T,Y)`. Оскільки побудови параметричних рівнянь ламаних в нашому посібнику зустрічатимуться доволі часто, то створимо окрему функцію `pbroken_line(t,T,X,Y)` яка, використовуючи формулу Бернштейна, по списку T значень параметра кривої t у вузлах (точках зламу) з координатами X,Y, генеруватиме параметричні рівняння ламаної (тобто повертатиме список з двох символьних виразів, які залежать від символьної змінної t).

```
def pbroken_line(t,T,X,Y):
    """  Параметричне рівняння ламаної від символьної змінної t.
    t - ім'я змінної, T- список/масив значень параметра кривої у
    вузлах ламаної.
    X,Y - масиви/списки абсцис і ординат вузлів ламаної.
    Значення T[i] повинні монотонно зростати і бути різними.
    Повинно бути: len(T)==len(X)==len(Y) та len(T)>=2    """
    xt=broken_line(t,T,X)
    if xt is None: return None
    yt=broken_line(t,T,Y)
    if yt is None: return None
    return [xt,yt]
```

Збережіть функцію `pbroken_line` в модулі broken.py.

Приклад 3. Згенерувати параметричні рівняння квадрата, показаного вище, і по ним побудувати його графік.

Розв'язання. Почніть новий сценарій.

```
import numpy as np
import matplotlib.pyplot as plt
from sympy import symbols,lambdify
from broken import pbroken_line      # імпортування нашої функції
import sys
t = symbols('t')
```

Створіть список T значень параметра t в вершинах ламаної враховуючи, що в точці (0,0) ми повинні побувати двічі при обході квадрата по контуру.

```
T=[0,1,2,3,4]       # значення параметра t=0 та t=4 відповідають вершині (0,0)
```

Задайте списки X,Y координат вершин квадрата і генеруйте його параметричні рівняння.

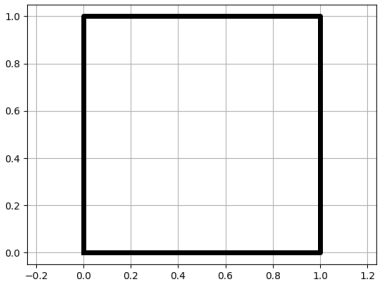
```
X=[0,1,1,0,0]
Y=[0,0,1,1,0]
xy=pbroken_line(t,T,X,Y)
if xy is None: sys.exit()
else: print(xy)
```

```
[0.5*t+0.5*Abs(t-3)-0.5*Abs(t-2)-0.5*Abs(t-1),
 -0.5*t-0.5*Abs(t-3)-0.5*Abs(t-2)+0.5*Abs(t-1)+2.0]
```

Використовуючи ці рівняння, побудуйте графік ламаної.

```
t_=np.linspace(0,4,41)
```

```
xt=lambdify(t,xy[0],"numpy")
yt=lambdify(t,xy[1],"numpy")
plt.close('all')
fig,ax = plt.subplots(1,1)
ax.plot(xt(t_),yt(t_),'k',lw=3)
ax.axis('equal')
ax.grid(True)
```



Зауваження. Тут знову, якщо вас цікавить лише графік ламаної, достатньо однієї інструкції `ax.plot(X,Y,'-`,...), де X,Y масиви координат вершин ламаної з нашого прикладу. Але, якщо для якихось цілей вам знадобляться координати проміжних точок, то наведені рівняння стануть вам в нагоді. Це потрібно, наприклад, коли рівняння ламаних використовуються при побудові параметричних рівнянь поверхонь.

10.2. Параметричні рівняння поверхонь з ребрами.

У цьому параграфі ми розглянемо способи побудови параметричних рівнянь поверхонь обертання ламаних і поверхонь подібних поперечних перерізів.

Рівняння поверхонь обертання ламаних конструюються так само, як для поверхонь обертання будь-якої параметричної кривої. Якщо криволінійний відрізок L в площині XZ має рівняння $x_L = \varphi(u),\ z_L = \psi(u)\ (u_0 \le u \le u_1)$, то параметричним рівнянням поверхні, утвореної його обертанням навколо осі Z буде

$$X = \varphi(u)\cdot\cos v,\ Y = \varphi(u)\cdot\sin v,\ Z = \psi(u)\ \text{ при } u_0 \le u \le u_1,\ 0 \le v \le 2\pi\,. \quad (1)$$

Приклад 1. Генерувати рівняння поверхні обертання навколо осі Z ламаної ⨘, яка в площині XZ має вершини `{0,0},{2,0},{1,1},{2,2},{0,2}`.

Розв'язання.

```
import numpy as np
import matplotlib.pyplot as plt
from sympy import symbols,lambdify
from broken import pbroken_line
import sys
```

Спочатку генеруємо параметричне рівняння ламаної.

```
u = symbols('u')
U=np.array([0,1,2,3,4])          # значення параметра у вузлах
X=np.array([0,2,1,2,0])
Z=np.array([0,0,1.5,3,3])
xz=pbroken_line(u,U,X,Z)         # генерування рівняння ламаної
if xz is None: sys.exit()
else: print(xz)
```

Будуть надруковані наступні вирази координатних функцій.

```
[-1.5*Abs(u-3)+Abs(u-2)-1.5*Abs(u-1)+4.0,
 -0.75*Abs(u-3)+0.75*Abs(u-1)+1.5]
```

Перетворіть символьні вирази на пітонівські функції і по ним побудуйте графік ламаної.

```
[xu,zu]=xz
```

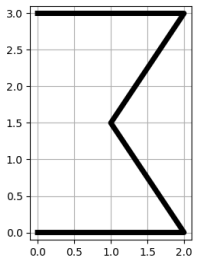
```
xc=lambdify(u,xu,"numpy")
zc=lambdify(u,zu,"numpy")
plt.close('all')
fig,ax0 = plt.subplots(1,1)
u_=np.linspace(0,4,41)
x=xc(u_)
z=zc(u_)
ax0.plot(x,z,'k',lw=5)          # наступний рисунок ліворуч
ax0.set_xlim(-0.1,2.1)
ax0.set_ylim(-0.1,3.1)
ax0.grid(True)
ax0.set_aspect(1)
```

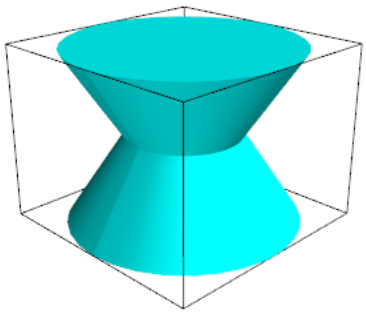
За формулами (1) створіть координатні функції $X(u,v), Y(u,v), Z(u,v)$ поверхні обертання ламаної і сітку точок (x,y,z) на ній.

```
def X(u,v):  return xc(u)*np.cos(v)
def Y(u,v):  return xc(u)*np.sin(v)
def Z(u,v):  return zc(u)
v_ = np.linspace(0,2*np.pi,31)
U,V=np.meshgrid(u_,v_)
x = X(U,V)
y = Y(U,V)
z = Z(U,V)
```

Поверхню побудуємо за допомогою функцій бібліотеки mayavi.

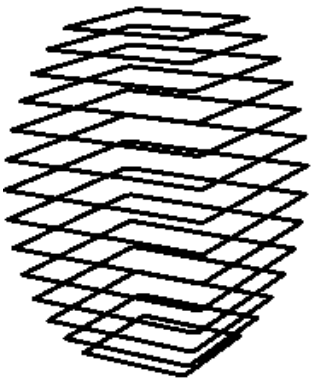
```
from mayavi import mlab
mlab.close(all=True)
mlab.figure(fgcolor=(0,0,0), bgcolor=(1,1,1))
sf=mlab.mesh(x,y,z,color=(0,1,1))     # наступний рисунок праворуч
mlab.outline(sf)
```

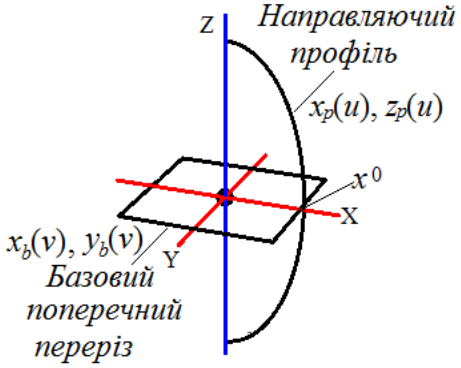




■

Поверхнею подібних поперечних перерізів [3] зветься поверхня, перерізи якої паралельними площинами є подібними кривими пропорційних розмірів. Наприклад, всі перерізи можуть бути квадратами, як показано на наступному рисунку ліворуч.





Припустимо, що площини перерізів перпендикулярні осі Z і всі подібні до одного базового перерізу з параметричним рівнянням $x_b(v), y_b(v)$ $(v_0 \le v \le v_1)$. Тобто в плоскостях $z = z_p(u)$ рівняння кривих (перерізів) є $x = \alpha(z_p) \cdot x_b(v)$, $y = \alpha(z_p) \cdot y_b(v)$, де $\alpha(z_p) = \alpha(z_p(u)) = \tilde{\alpha}(u)$ – коефіцієнт пропорційності. Але це означає, що параметричні рівняння поверхні матимуть вигляд

$$x(u,v) = \tilde{\alpha}(u) \cdot x_b(v)\,; \quad y(u,v) = \tilde{\alpha}(u) \cdot y_b(v); \quad z(u,v) = z_p(u)\,. \qquad (2)$$

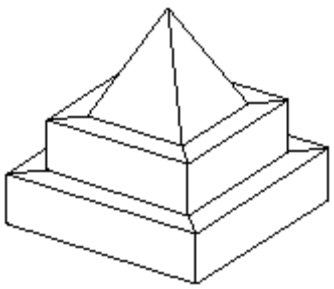
На базовій кривій/перерізі візьмемо значення параметра $v = v^0$, для якого $y_b(v^0) = 0$, і через $x^0 = x_b(v^0) \neq 0$ позначимо відповідне йому значення абсциси. Тоді формули $x(u, v^0) = \tilde{\alpha}(u) \cdot x^0 = x_p(u)$, $y = 0$, $z(u, v_0) = z_p(u)$ визначатимуть в площині XZ деяку криву (напрямний профіль). Якщо рівняння $x_p(u), z_p(u)$ $(u_0 \le u \le u_1)$ напрямного профілю в площині XZ задано, то $\tilde{\alpha}(u) = \frac{x_p(u)}{x^0}$, і рівняння поверхні набуває вигляду:

$$\begin{aligned} x(u,v) &= [x_p(u)/x^0] \cdot x_b(v), \\ y(u,v) &= [x_p(u)/x^0] \cdot y_b(v)\,, \qquad (u_0 \le u \le u_1\ ,\ v_0 \le v \le v_1) \\ z(u,v) &= z_p(u), \end{aligned} \qquad (3)$$

де x^0 – абсциса точки перетину базового перерізу з віссю X, що збігається з абсцисою напрямного профіля в цій же точці.

Таким чином, маючи параметричні рівняння базового перерізу та напрямного профілю, за формулами (3) можна сконструювати рівняння поверхні пропорційних поперечних перерізів.

Приклад 2. Написати параметричне рівняння чотирикутної триступінчастої піраміди з квадратною основою (див. рисунок), розглядаючи її як поверхню подібних поперечних перерізів.



Ламана L (напрямний профіль), розташована в площині XZ, і має вершини `(0,0),(4,0),(4,2),(3,2),(3,4),(2,4),(0,8)`, а крива S (базовий переріз) – в площині XY і є квадратом з центром в точці (0,0).

Розв'язання.

```
import numpy as np
import matplotlib.pyplot as plt
from sympy import symbols,lambdify
from broken import pbroken_line
import sys
```

Згенеруйте рівняння напрямної ламаної *L*.

```
u = symbols('u')
U=[0,1,2,3,4,5,6]
XL=np.array([0,4,4,3,3,2,0])
ZL=np.array([0,0,2,2,4,4,8])
xzL=pbroken_line(u,U,XL,ZL)
if xzL is None: sys.exit()
else: print(xzL)
```

Будуть надруковані наступні вирази

```
[u-0.5*Abs(u-5)-0.5*Abs(u-4)+0.5*Abs(u-3)-
                           0.5*Abs(u-2)-2.0*Abs(u-1)+6.0,
2.0*u+2.0*Abs(u-5)-1.0*Abs(u-4)+Abs(u-3)-
                           1.0*Abs(u-2)+Abs(u-1)-8.0]
```

```
xL=lambdify(u,xzL[0],"numpy")
zL=lambdify(u,xzL[1],"numpy")
```

Для перевірки побудуйте зображення ламаної по її рівнянням.

```
plt.close('all')
fig,ax = plt.subplots(1,2,figsize=(8,4))
u_=np.linspace(0,6,61)
ax[0].plot(xL(u_),zL(u_),'k',lw=5)       # наступний рисунок ліворуч
ax[0].set_xlim(-0.2,4.2)
ax[0].set_ylim(-0.2,8.2)
ax[0].grid(True)
ax[0].set_aspect(1)
```

Рівняння базової ламаної (квадрата) з вершинами в точках `(4,-4),(4,4),(-4,4),(-4,-4)` можна сконструювати аналогічно.

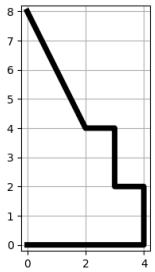
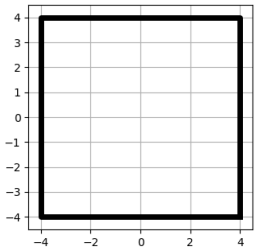
```
v = symbols('v')
V=[0,1,2,3,4]
XS=np.array([4,4,-4,-4,4])
YS=np.array([-4,4,4,-4,-4])
xyS=pbroken_line(v,V,XS,YS)
if xyS is None: sys.exit()
else: print(xyS)
```

```
[4.0*v+4.0*Abs(v-3)+4.0*Abs(v-2)-4.0*Abs(v-1)-12.0,
 4.0*v+4.0*Abs(v-3)-4.0*Abs(v-2)-4.0*Abs(v-1)-4]
```

```
xS=lambdify(v,xyS[0],"numpy")
yS=lambdify(v,xyS[1],"numpy")
```

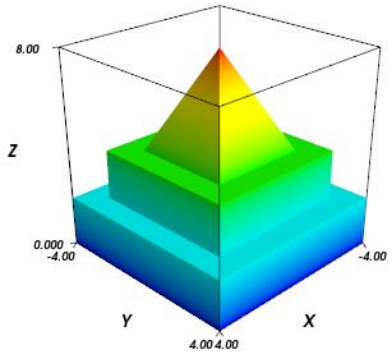
Побудуйте квадрат по згенерованим рівнянням.

```
v_=np.linspace(0,4,41)
ax[1].plot(xS(v_),yS(v_),'k',lw=5)    # наступний рисунок праворуч
ax[1].set_xlim(-4.5,4.5)
ax[1].set_ylim(-4.5,4.5)
ax[1].grid(True)
ax[1].set_aspect(1)
```

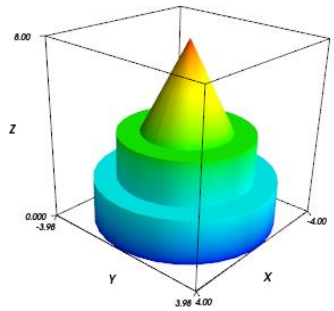
Абсциса точки перетину просторових ламаних L і S відома $x^0 = 4$. Тоді, відповідно до (3), можна сконструювати рівняння поверхні подібних поперечних перерізів, і побудувати її зображення.

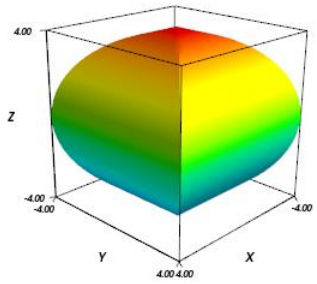
```
x0=4                                    # абсциса перетину L і S
def x(u,v): return xL(u)/x0*xS(v)
def y(u,v): return xL(u)/x0*yS(v)
def z(u,v): return zL(u)
U,V=np.meshgrid(u_,v_)
Xp = x(U,V)
Yp = y(U,V)
Zp = z(U,V)
from mayavi import mlab
mlab.close(all=True)
mlab.figure(fgcolor=(0,0,0), bgcolor=(1,1,1))
sf=mlab.mesh(Xp,Yp,Zp)
mlab.outline()
mlab.axes()
```



Якщо базисний переріз є колом $x_b(v) = x^0 \cos v,\ y_b(v) = x^0 \sin v$, то (3) збігаються з рівняннями поверхні обертання напрямного профілю $x_p(u),\ z_p(u)$ $(u_0 \le u \le u_1)$ навколо осі Z. Дійсно, до формул (3), якщо вони розглядаються як рівняння поверхні обертання, параметр x^0 входить двічі: один раз як дільник (з формули (3)), і другий раз, як множник (радіус) у рівнянні базового перерізу $x_b(v) = x^0 \cos v,\quad y_b(v) = x^0 \sin v$. Тому він скорочується.

```
v_ = np.linspace(0,2*np.pi,31)      # продовження попереднього сценарію
U,V=np.meshgrid(u_,v_)
Xr=xL(U)*np.cos(V)
Yr=xL(U)*np.sin(V)
Zr=zL(U)
mlab.figure(fgcolor=(0,0,0), bgcolor=(1,1,1))
sf=mlab.mesh(Xr,Yr,Zr)              # наступний рисунок ліворуч
mlab.outline()
mlab.axes()
```





Візьмемо напрямний профіль L у формі півкола $x_p(u) = 4\cos u$, $z_p(u) = 4\sin u$ $(-\pi/2 \le u \le \pi/2)$, а базовим – попередній квадрат. Тоді дільник $x^0 = 4$ в рівняннях $\mathrm{Xc}(u,v)$, $\mathrm{Yc}(u,v)$ скоротиться з радіусом кола $R = 4$ і ми матимемо поверхню «округленої скриньки».

```
u_=np.linspace(-np.pi/2,np.pi/2,31)  # продовж. попереднього сценарію
v_=np.linspace(0,4,41)
U,V=np.meshgrid(u_,v_)
Xc=np.cos(U)*xS(V)
Yc=np.cos(U)*yS(V)
Zc=4*np.sin(U)                       # радіус кола 4
mlab.figure(fgcolor=(0,0,0), bgcolor=(1,1,1))
sf=mlab.mesh(Xc,Yc,Zc)               # попередній рисунок праворуч
mlab.outline()
mlab.axes()
```

■

Поверхні більшості класичних геометричних фігур таких, як призма, циліндр, конус, піраміда (всі з основами), куб, паралелепіпед, є поверхнями подібних поперечних перерізів. Тому наведений спосіб побудови параметричних рівнянь таких поверхонь придатний і до них.

10.3. Модифікація рівнянь функцій, кривих і поверхонь.

В цьому параграфі наводяться приклади допоміжних функцій, застосування яких разом з рівняннями функцій, кривих або поверхонь даватиме рівняння схожих об'єктів зі змінами лише на деяких ділянках. Наприклад, так ми отримуватимемо криві або поверхні зі зрізами, «вм'ятинами», тощо.

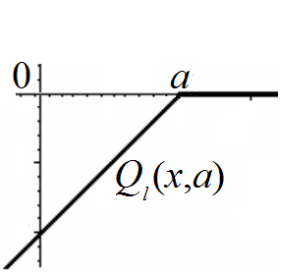
Визначимо функції

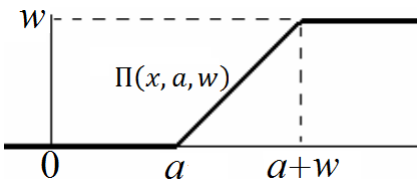
$$Q_l(x,a) = \frac{1}{2}\,(x - a - |\,x - a\,|) = \begin{cases} x - a & ,\ x \le a \\ 0 & ,\ x > a \end{cases} \qquad (1)$$

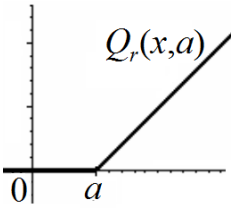
$$\Pi(x,a,w) = \frac{1}{2}\,(w + |x - a| - |x - a - w|) = \begin{cases} 0 & ,\ x \le a \\ x - a & ,\ a \le x \le a + w \\ w & ,\ x \ge a + w \end{cases} \qquad (2)$$

$$Q_r(x,a) = \frac{1}{2}\,(x - a + |\,x - a\,|) = \begin{cases} 0 & ,\ x \le a \\ x - a & ,\ x > a \end{cases} \qquad (3)$$

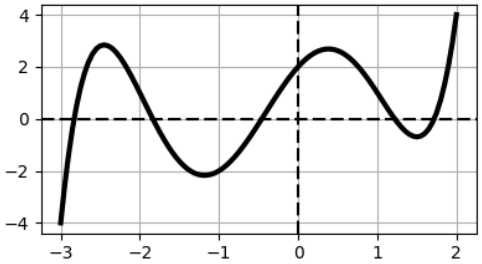
На наступному рисунку наведені графіки цих функцій, розглядуваних як функції однієї змінної x.



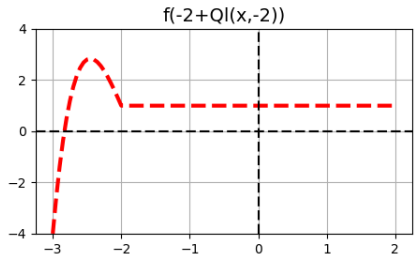
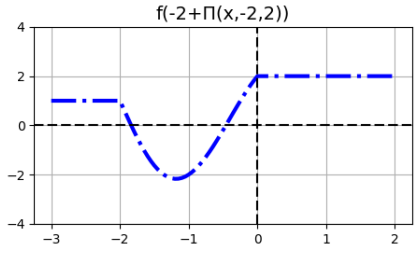
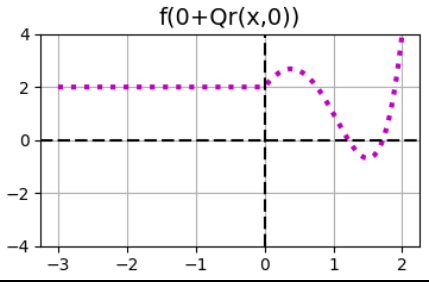




Якщо задана функція $f(x)$ на всій дійсній осі, то її суперпозиція з цими функціями буде неперервною і шматковою. Для пояснення візьмемо функцію $f(x) = 0.4\,x^5 + 0.875\,x^4 - 2.25\,x^3 - 3.375\,x^2 + 3.35\,x + 2.$



Покладемо $a = -2\,, w = 2$ і побудуємо функції $F_1(x) = f(a + Q_l(x,a))$, $F_2(x) = f\big(a + \Pi(x,a,w)\big)$, $F_3(x) = f(a + Q_r(x,a))$. Їх графіки наведено в правому стовпці наступної таблиці.

$F_1(x) = f(a + Q_l(x,a)) = \begin{cases} f(x) & , x \le a \\ f(a) & , x > a \end{cases}$	
$F_2(x) = f\big(a + \Pi(x,a,w)\big) = \begin{cases} f(a) & , \quad x \le a \\ f(x)\,, & a \le x \le a + w \\ f(a+w)\,, & x \ge a + w \end{cases}$	
$F_3(x) = f(a + Q_r(x,a)) = \begin{cases} f(a) & , x \le a \\ f(x) & , x > a \end{cases}$	

Фактично, функції $F_1(x), F_2(x), F_3(x)$ виділяють ділянки початкової функції $f(x)$. Навпаки, якщо задано функції $F_1(x), F_2(x), F_3(x)$, то сума $F(x) = F_1(x) + (F_2(x) - F_2(a)) + (F_3(x) - F_3(a+w))$ даватиме входову функцію $f(x)$. Але функції $F_i(x)$ у виразі $F(x)$ можуть бути створені з будь-яких аналітичних виразів $f_i(x)$, а не з однієї і тієї ж $f(x)$. Тоді результатом буде функція, створена

зі шматків $f_i(x)$. Узагальнюючи наведені міркування маємо наступне твердження.

Лема. Нехай дано множину точок $\{x_i\}_{i=0}^{n},\ x_0 < x_1 < \cdots < x_n$ і послідовність неперервних функцій $\{f_i(x)\}_{i=0}^{n+1}$ таких, що $f_i(x_i) = f_{i+1}(x_i)\ (i = 0,\ 1, \dots,\ n)$. Тоді неперервна кускова функція

$$f(x) = \begin{cases} f_0(x)\ ,\ x \le x_0 \\ f_1(x)\ ,\ x_0 < x \le x_1 \\ \quad \cdots \\ f_n(x)\ ,\ x_{n-1} < x \le x_n \\ f_{n+1}(x)\ ,\ x > x_n \end{cases}$$

може бути представлена у вигляді

$$\begin{aligned} f(x) = f_0\big(x_0 + Q_l(x, x_0)\big) \\ + \sum_{i=1}^{n} \big(f_i(x_{i-1} + \Pi(x, x_{i-1}, x_i - x_{i-1})) - f_i(x_{i-1})\big) + \\ + f_{n+1}\big(x_n + Q_r(x, x_n)\big) - f_{n+1}(x_n) \end{aligned} \tag{4}$$

Доведення виконується перевіркою рівності лівої і правої частин (4) окремо на кожній ділянці $x \le x_0,\ x_{i-1} \le x \le x_i\ (i = 1,2, \dots, n),\ x \ge x_n$ [4].

Зауваження. Для неперервних ламаних з вершинами в точках $(x_i, y_i), i = 0,1, \dots, n,\ x_0 < x_1 < \cdots < x_n$ остання формула набуває виду

$$\begin{aligned} l(x) = y_1 + \frac{y_1 - y_0}{x_1 - x_0} Q_l(x, x_1) + \sum_{i=2}^{n-1} \frac{y_i - y_{i-1}}{x_i - x_{i-1}} \Pi(x, x_{i-1}, x_i - x_{i-1}) + \\ + \frac{y_n - y_{n-1}}{x_n - x_{n-1}} Q_r(x, x_{n-1})\ . \end{aligned} \tag{5}$$

Після спрощення вона може бути конвертована до вигляду Бернштейна (ф. (2) п. 10.1).

Визначимо функції

$$mn(x, y) = \frac{1}{2}\ (x + y - |\ x - y\ |) = \begin{cases} x\ ,\ x \le y \\ y\ ,\ x > y \end{cases} \tag{6}$$

$$mx(x, y) = \frac{1}{2}\ (x + y + |\ x - y\ |) = \begin{cases} x\ ,\ x \ge y \\ y\ ,\ x < y \end{cases} \tag{7}$$

які повертають найменший та найбільший зі своїх аргументів. Підставляючи в перший аргумент будь-яку функцію $f(x)$ при фіксованому другому y_0, ми отримуємо рівняння нової функції $F(x) = mn(f(x), y_0)$ (або $mx(f(x), y_0)$), графік якої являє графік входової функції, обрізаний зверху (знизу) прямою $y = y_0$.

Приклад 1. Побудувати графік синусоїди $\sin \pi x$, обрізаної зверху значенням `0.5`, і окремо обрізаної знизу значенням `-0.5`.

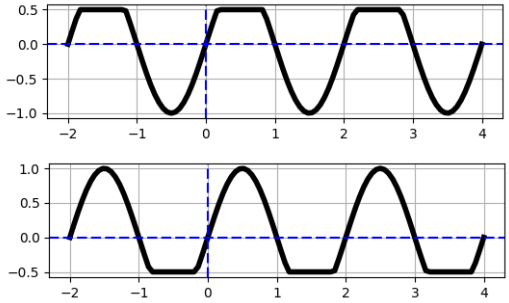
Розв'язання. Для демонстрації простоти запропонованого підходу будемо генерувати символьні рівняння функцій, а для побудови графіків перетворювати їх на `numpy` функції.

```
import numpy as np
```

```
import matplotlib.pyplot as plt
from sympy import symbols,lambdify,Abs,sin,pi
x,y = symbols('x y')
def mn(x,y): return (x+y-Abs(x-y))/2
def mx(x,y): return (x+y+Abs(x-y))/2
ymn=mn(sin(pi*x),0.5)                    # обрізання зверху
print(ymn)
sin(pi*x)/2-Abs(sin(pi*x)-0.5)/2+0.25
ymx=mx(sin(pi*x),-0.5)                   # обрізання знизу
print(ymx)
sin(pi*x)/2+Abs(sin(pi*x)+0.5)/2-0.25
```

Маючи рівняння перетворених функцій, побудуйте їх графіки.

```
Ymn=lambdify(x,ymn,"numpy")
Ymx=lambdify(x,ymx,"numpy")
fig,ax = plt.subplots(2,1)
def graphsettings0(ax):
    ax.axhline(0,color='b',dashes=(5,2))
    ax.axvline(0,color='b',dashes=(5,2))
    ax.set_aspect(1)
    ax.grid(True)
X=np.linspace(-2,4,101)
ax[0].plot(X,Ymn(X),'k',lw=4)
graphsettings0(ax[0])
ax[1].plot(X,Ymx(X),'k',lw=4)
graphsettings0(ax[1])
```



Зауваження. Звісно, що замість операцій $mn(x,y)$, $mx(x,y)$ ви можете використати функції `numpy.min()` та `numpy.max()`, застосувавши їх до відповідних масивів. Але працювати з функціями зручніше, бо трактовка мінімумів і максимумів як функцій спрощує розуміння задачі у випадках, коли наприклад, слід результівну функцію інтегрувати або використовувати в диференціальних рівняннях в якості коефіцієнтів.

■

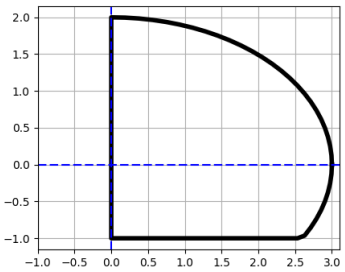
Схожим чином функції $mn(x,y)$, $mx(x,y)$ можна застосовувати до параметричних рівнянь кривих і поверхонь.

Приклад 2. Дано еліпс з центром на початку координат і напівосями довжиною 3 та 2. Побудувати параметричне рівняння контура фігури, отриманої обрізанням еліпса по лініям $x = 0$ та $y = -1$

Розв'язання.

```
import numpy as np
import matplotlib.pyplot as plt
from sympy import symbols,lambdify,Abs,sin,cos
```

```
t = symbols('t')
def mx(x,y): return (x+y+Abs(x-y))/2
def ell(t): return [3*cos(t),2*sin(t)]
ellx=mx(ell(t)[0],0)
elly=mx(ell(t)[1],-1)
print('Параметричне рівняння фігури:\n',ellx,'\n',elly)
```
```
Параметричне рівняння фігури:
  3*cos(t)/2+3*Abs(cos(t))/2
  sin(t)+Abs(2*sin(t)+1)/2-1/2
```
По отриманим рівнянням побудуйте графік кривої.
```
Xe=lambdify(t,ellx,"numpy")
Ye=lambdify(t,elly,"numpy")
fig,ax = plt.subplots(1,1)
T=np.linspace(0,2*np.pi,101)
ax.plot(Xe(T),Ye(T),'k',lw=4)
ax.axhline(0,color='b',dashes=(5,2))
ax.axvline(0,color='b',dashes=(5,2))
ax.set_aspect(1)
ax.set_xlim(-1,3.1)
ax.grid(True)
```



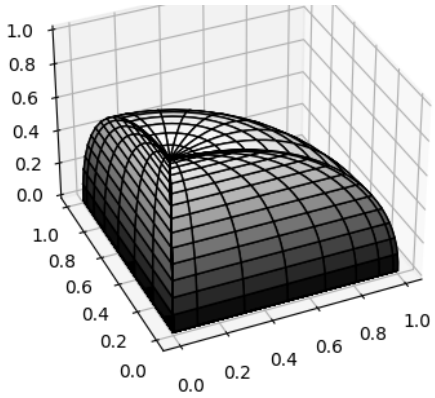
■

Схожий підхід дозволяє отримувати параметричні рівняння поверхонь «обрізаних» фігур.

Приклад 3. Побудувати параметричні рівняння поверхні частини одиничної кулі, що розташована в першому октанті $x \geq 0, y \geq 0, z \geq 0$.

Розв'язання.
```
import numpy as np
import matplotlib.pyplot as plt
from sympy import symbols,lambdify,Abs,sin,cos
u,v = symbols('u,v')
def mx(x,y): return (x+y+Abs(x-y))/2
def ball(u,v): return [cos(u)*cos(v),sin(u)*cos(v),sin(v)]
bx=mx(ball(u,v)[0],0)
by=mx(ball(u,v)[1],0)
bz=mx(ball(u,v)[2],0)
print('Параметричні рівняння фігури:\n',bx,'\n',by,'\n',bz)
```
```
Параметричні рівняння фігури:
  cos(u)*cos(v)/2+Abs(cos(u)*cos(v))/2
  sin(u)*cos(v)/2+Abs(sin(u)*cos(v))/2
  sin(v)/2+Abs(sin(v))/2
```
По отриманим рівнянням побудуйте поверхню.
```
plt.close('all')
fig = plt.figure()
```

```
ax = fig.add_subplot(111, projection='3d')
Xb=lambdify((u,v),bx,"numpy")
Yb=lambdify((u,v),by,"numpy")
Zb=lambdify((u,v),bz,"numpy")
U = np.linspace(0, 2 * np.pi, 61)
V = np.linspace(-np.pi/2, np.pi/2, 61)
V=V[:,None]
ax.plot_surface(Xb(U,V),Yb(U,V),Zb(U,V),
                    cmap='gray',edgecolor ='k',rstride=2,cstride=2)
```



■

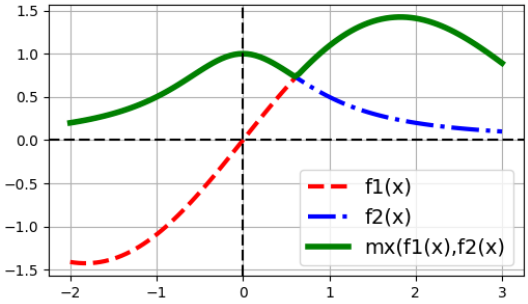
Досі в якості другого аргументу функцій $mn(x,y)$, $mx(x,y)$ ми використовували сталу. Підставляючи до їх першого аргументу одну функцію $f(x)$, а до другого – іншу $g(x)$, ми отримуватимемо рівняння нової функції $F(x)=mx(f(x),g(x))$ (або $mn(f(x),g(x))$), графік якої представлятиме неперервну криву складену із частин графіків функцій $f(x)$ та $g(x)$ розташованих вище (або нижче) відповідних частин іншої функції.

Приклад 4. Побудувати рівняння кривої, яка складається з більших (по вертикалі) значень функцій $f_1(x)=\frac{x}{4}+\sin x$ і $f_2(x)=\frac{1}{1+x^2}$.

Розв'язання.

```
import numpy as np
import matplotlib.pyplot as plt
from sympy import symbols,lambdify,Abs,sin
x = symbols('x')
def mx(x,y): return (x+y+Abs(x-y))/2
def f1(t): return t/4+sin(t)
def f2(t): return 1/(1+t**2)
def rz(t): return mx(f1(t),f2(t))
print(rz(x)
x/8+sin(x)/2+Abs(x/4+sin(x)-1/(x**2+1))/2+1/(2*(x**2+1))
Rz=lambdify(x,rz(x),"numpy")
F1=lambdify(x,f1(x),"numpy")
F2=lambdify(x,f2(x),"numpy")
plt.close('all')
fig,ax = plt.subplots(1,1)
X=np.linspace(-2,3,101)
ax.plot(X,F1(X),'r--',lw=3,label='f1(x)')
ax.plot(X,F2(X),'b-.',lw=3,label='f2(x)')
ax.plot(X,Rz(X),'g',lw=4,label='mx(f1(x),f2(x)')
ax.axhline(0,color='k',dashes=(5,2))
```

```
ax.axvline(0,color='k',dashes=(5,2))
ax.set_aspect(1)
ax.grid(True)
```



Зауважте, що точку перетинання кривих обчислювати не довелося. ■

Наведені функції $mn(x,y)$, $mx(x,y)$ можна застосовувати до функцій двох змінних, обрізаючи їх значення зверху або знизу.

Приклад 5. Демонстрація застосування операцій $mn(x,y)$, $mx(x,y)$ до функцій двох змінних.

Почніть новий сценарій. Імпортуйте потрібні модулі і функції, а також створіть допоміжні функції (1) – (3), (6), (7).

```
import numpy as np
import matplotlib.pyplot as plt
from broken import broken_line
from sympy import symbols,lambdify,Abs,sqrt,simplify,sin,cos
import sys
plt.close('all')
def Qr(x,a): return (x-a+Abs(x-a))/2
def Ql(x,a): return (x-a-Abs(x-a))/2
def PP(x,a,w): return (w+Abs(x-a)-Abs(x-a-w))/2
def mx(x,y): return (x+y+Abs(x-y))/2
def mn(x,y): return (x+y-Abs(x-y))/2
```

Нам доведеться виконувати однакові побудови графіків символьних функцій двох змінних кілька разів. Для скорочення коду напишемо підходящу функцію.

```
def draw_surface(f, x_, y_, aspect, **kwargs ):
    """ f - символьна функція двох символьних змінних
        x_, y_ - двовимірні масиви (сітка точок, на якій
        будуватиметься графік функцій)       """
    fig = plt.figure()
    ax = fig.add_subplot(111,projection='3d')
    x,y = symbols('x y')
    F=lambdify((x,y),f(x,y),"numpy")
    z=F(x_,y_)
    ax.plot_surface(x_,y_,z,color='cyan', edgecolor ='k',**kwargs)
    ax.set_box_aspect(aspect =aspect)
```

Створіть функцію $f_1(x,y)$, яка має конічну форму над еліптичною зоною в площині X,Y, і дорівнює нулю поза нею.

```
x,y = symbols('x y');
def f1(x,y): return mx(1-sqrt(x**2+3*y**2),0)
print('f1(x,y)=',simplify(f1(x,y)))
f1(x,y)=-sqrt(x**2+3*y**2)/2+Abs(sqrt(x**2+3*y**2)-1)/2+1/2
```

Побудуйте графік функції $f_1(x,y)$ (наступний рисунок ліворуч).

```
y_,x_=np.mgrid[-1:1:201j,-1:1:201j]
draw_surface(f1,x_,y_,(2,2,1),rstride=10,cstride=10 )
```

Створіть функцію $f_2(x,y)$, яка матиме форму «зламаної» площини. Її графік показано на наступному рисунку всередині.

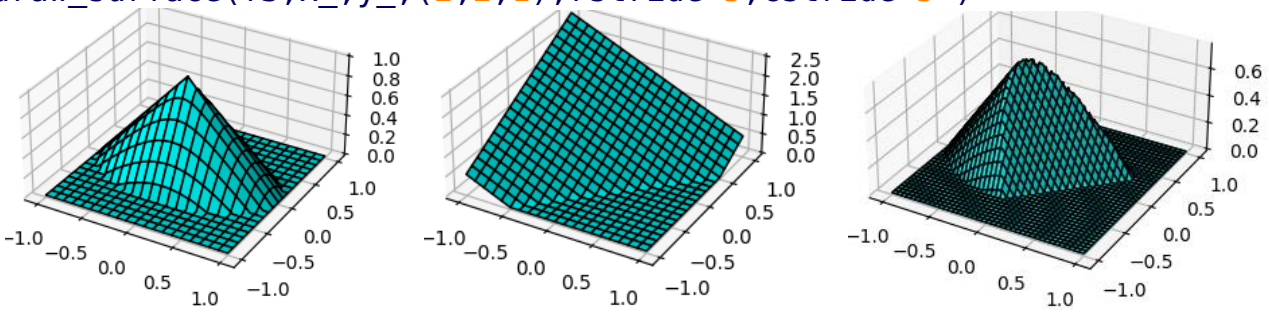
```
def f2(x,y): return Qr(0.5-x+y,0)
print('f2(x,y)=',simplify(f2(x,y)))
f2(x,y)=-x/2+y/2+Abs(-x+y+0.5)/2+0.25
draw_surface(f2,x_,y_,(2,2,1),rstride=10,cstride=10 )
```

Створіть функцію, графік якої має форму «зрізаної» по $f_2(x,y)$ конічної поверхні $f_1(x,y)$ (наступний рисунок праворуч).

```
def f3(x,y): return mn(f1(x,y),f2(x,y))
print('f3(x,y)=',simplify(f3(x,y)))
f3(x,y)=-x/4+y/4-sqrt(x**2+3*y**2)/4+
        Abs(sqrt(x**2+3*y**2)-1)/4+Abs(-x+y+0.5)/4-
        Abs(x-y-sqrt(x**2+3*y**2)+Abs(sqrt(x**2+3*y**2)-1)-
                                  Abs(-x+y+0.5)+0.5)/4+0.375
```

Останнє рівняння має доволі складний вигляд, але графік його виглядає «приємно».

```
draw_surface(f3,x_,y_,(2,2,1),rstride=5,cstride=5 )
```



В цьому ж сценарії побудуємо функцію, графік якої має вигляд усіченої чотирикутної піраміди. Для цього нам знадобиться наша функція `broken_line()` (див. п. 10.1), що генерує символьний вираз ламаної. За її допомогою ми створимо функцію двох змінних, графік якої має форму трапеції (наступний рисунок ліворуч).

```
X=[-3,-2,-1,0,1,2]
Z=[0,0,1,1,0,0]
fs4=broken_line(x,X,Z)
if fs4 is None: sys.exit()
def f4(x,y): return fs4
print('f4(x,y)=',simplify(f4(x,y)))
f4(x,y)=-0.5*Abs(x)+0.5*Abs(x-1)-0.5*Abs(x+1)+0.5*Abs(x+2)
y_,x_=np.mgrid[-3:2:201j,-3:2:201j]
draw_surface(f4,x_,y_,(6,6,1),rstride=10,cstride=10 )
```

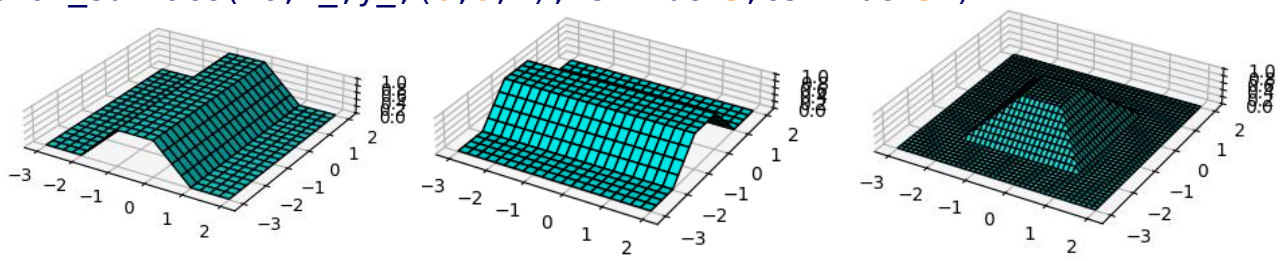
Таким же чином створіть функцію, графік якої має вигляд поверхні перенесення вздовж осі X (наступний рисунок всередині).

```
def f5(x,y): return fs4.subs(x,y)
draw_surface(f5,x_,y_,(6,6,1),rstride=10,cstride=10 )
```

Використання цих функцій в аргументах операції $mn(x,y)$ даватиме функцію з графіком у формі усіченої піраміди (наступний рисунок праворуч).

```
def f6(x,y): return mn(f4(x,y),f5(x,y))
```

```
print('f6(x,y)=',simplify(f6(x,y)))
f6(x,y)=-0.25*Abs(x)-0.25*Abs(y)+0.25*Abs(x-1)-
    0.25*Abs(x+1)+0.25*Abs(x+2)+0.25*Abs(y-1)-0.25*Abs(y+1)+
    0.25*Abs(y+2)-0.25*Abs(Abs(x)-Abs(y)-Abs(x-1)+Abs(x+1)-
    Abs(x+2)+Abs(y-1)-Abs(y+1)+Abs(y+2))
draw_surface(f6,x_,y_,(6,6,1),rstride=5,cstride=5 )
```



Якщо є рівняння двох функцій $f_1(x)$, $f_2(x)$ (або $f_1(x,y)$, $f_2(x,y)$), причому графік другої частково розташований нижче першої, то операція $mn(f_1,f_2)$ створюватиме рівняння f_1 з «вм'ятиною» у формі f_2.

$$F(x,y)=mn(f_1(x),f_2(x)) \quad \text{для явно заданих функцій } f_1(x),\ f_2(x) \tag{8}$$

$$F(x,y)=mn(f_1(x,y),f_2(x,y)) \quad \text{для явно заданих функцій } f_1(x,y),\ f_2(x,y) \tag{9}$$

Продемонструємо цю поведінку. Продовжіть попередній сценарій. Візьмемо конічну поверхню, рівняння якої `f1(x,y)` побудовано на початку прикладу, і побудуємо на ній «вм'ятину/виріз» у формі параболоїда (наступний рисунок ліворуч).

```
def f7(x,y): return x**2+y**2
def f8(x,y): return mn(f1(x,y),f7(x,y))
y_,x_=np.mgrid[-1:1:201j,-1:1:201j]
draw_surface(f8,x_,y_,(2,2,1) ,rstride=3,cstride=6,cmap='Greys')
```

Цей підхід також спрацьовує, коли перша поверхня S_1 задана параметричними рівняннями $x(u,v)$, $y(u,v)$, $z(u,v)$, а друга S_2, що частково розташована вище S_1, – явним $z=f(x,y)$. Для побудови параметричного рівняння поверхні S_1 з вм'ятиною у формі S_2 третю функцію $z(u,v)$ слід замінити на $mn\left(z(u,v),f\left(x(u,v),y(u,v)\right)\right)$. Тобто

$$\begin{aligned} X &= x(u,v) \\ Y &= y(u,v) \\ Z(u,v) &= mn\left(z(u,v),f\left(x(u,v),y(u,v)\right)\right) \end{aligned} \qquad \begin{aligned} &\text{Для параметричної поверхні} \\ &S_1\colon x(u,v),\ y(u,v),z(u,v) \\ &\text{Для явно заданої поверхні} \\ &S_2\colon z=f(x,y). \end{aligned} \tag{10}$$

Нехай першою поверхнею буде сфера, на якій слід створити вм'ятину у формі параболоїда. В продовженні попереднього сценарію введіть наступні інструкції.

```
def xsp(u,v): return cos(u)*cos(v)
def ysp(u,v): return sin(u)*cos(v)
def zsp(u,v): return sin(v)
def zdent(u,v):
    return mn(zsp(u,v),f7(xsp(u,v),ysp(u,v)))
u,v = symbols('u v');
Xsp=lambdify((u,v),xsp(u,v),"numpy")
Ysp=lambdify((u,v),ysp(u,v),"numpy")
Zsp=lambdify((u,v),zdent(u,v),"numpy")
```

```
u_,v_=np.mgrid[0:2*np.pi-1:100j,-np.pi/2:np.pi/2:50j]
X=Xsp(u_,v_)
Y=Ysp(u_,v_)
Z=Zsp(u_,v_)
zmax=Z.max()
fig = plt.figure()
ax = fig.add_subplot(111,projection='3d')
ax.plot_surface(X,Y,Z,cmap='gray')        # наступний рисунок всередині
ax.set_box_aspect(aspect =(1,1,zmax))
```

На створеному рисунку (всередині) ми спеціально обрали не повний діапазон $[0,\ 2\pi]$ зміни параметра u в рівнянні сфери, щоб можна було зазирнути всередину отриманої поверхні.

З використання функції абсолютного значення виконується операція «зминання», коли контур вм'ятини розташовано в площині $z = z_0$, і вм'ятина має форму зеркального відображення самої поверхні (зверху вниз) відносно площини $z = z_0$. Це робиться побудовою функцій

$$F(x,y) = z_0 - |f(x,y) - z_0| \quad \text{для явно заданої поверхні } z = f(x,y) \qquad (11)$$

$$\begin{aligned} X &= x(u,v) \\ Y &= y(u,v) \\ Z(u,v) &= z_0 - |z(u,v) - z_0| \end{aligned} \quad \text{для параметрично заданої поверхні } x(u,v),\ y(u,v), z(u,v) \qquad (12)$$

Такі ж перетворення можна виконувати і для кривих.

Для прикладу побудуємо параметричні рівняння поверхні конуса обертання навколо осі Z (розміри конуса дано в коді), «дзеркально зім'ятого» по площині $z = 1$. Продовжіть роботу з попереднім сценарієм. Конус є поверхнею обертання «трикутної» ламаної навкруги осі Z. Генеруємо її параметричне рівняння за допомогою нашої функції `pbroken_line()`.

```
ut=np.array([0,1,3])                    # значення параметра у вузлах
xt=np.array([0,1,0])
zt=np.array([0,0,2])
xz=pbroken_line(u,ut,xt,zt)             # генерування рівняння ламаної
if xz is None: sys.exit()
```

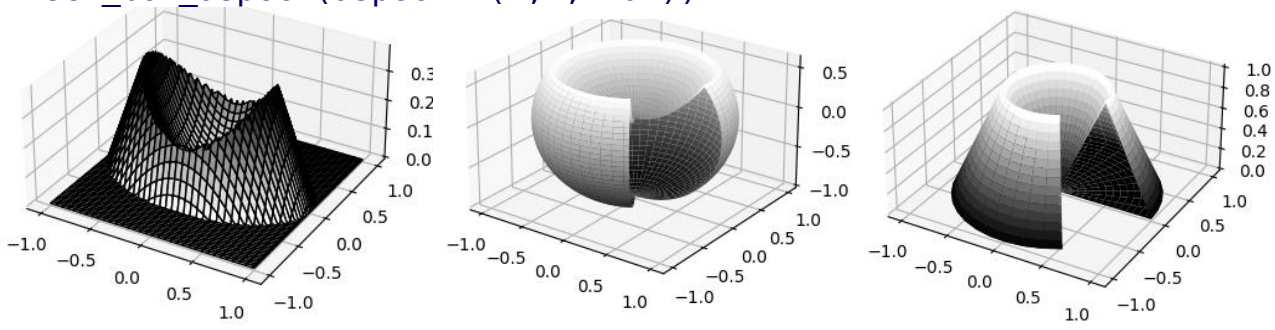
Використовуючи згенеровані вирази рівнянь ламаної разом з формулами (12), конструюємо параметричні рівняння «зім'ятого» конуса.

```
z0=1
def X(u,v): return xz[0]*cos(v); print('x=',X(u,v))
x=(0.25*u-0.75*Abs(u-1)+0.75)*cos(v)
def Y(u,v): return xz[0]*sin(v); print('y=',Y(u,v))
y=(0.25*u-0.75*Abs(u-1)+0.75)*sin(v)
def Z(u,v): return z0-Abs(xz[1]-z0); print('z=',Z(u,v))
z=1-Abs(0.5*u+0.5*Abs(u-1)-1.5)
```

Тепер побудуємо зображення отриманої поверхні.

```
V,U=np.mgrid[0:2*np.pi-1:31j,0:3:91j]
x=lambdify((u,v),X(u,v),"numpy")(U,V)
y=lambdify((u,v),Y(u,v),"numpy")(U,V)
z=lambdify((u,v),Z(u,v),"numpy")(U,V)
zmax=z.max()
fig = plt.figure()
```

```
ax = fig.add_subplot(111,projection='3d')
ax.plot_surface(x,y,z,cmap='gray')          # наступний рисунок праворуч
ax.set_box_aspect(aspect =(2,2,zmax))
```



■

Інколи зрізання параметричної кривої $(x(t), y(t))$ потрібно виконувати по похилій прямій. Нехай вона задана нормованим рівнянням $a\,x + b\,y + c = 0$ $(a^2 + b^2 = 1)$, де знаки коефіцієнтів a, b обрано так, що вектор $\boldsymbol{n} = (a, b)$ спрямований у бік відрізуваної частини кривої. Тоді параметричним рівнянням $(X(t), Y(t))$ «зрізаної» кривої буде

$$\begin{aligned} X &= a \cdot (mn(a\,x(t) + b\,y(t) + c, 0) - c) - b \cdot \big(-b\,x(t) + a\,y(t)\big) \\ Y &= b \cdot (mn(a\,x(t) + b\,y(t) + c, 0) - c) + a \cdot \big(-b\,x(t) + a\,y(t)\big) \end{aligned} \qquad (13)$$

Останні формули мають обмеження. Вони працюють некоректно, коли проекція відрізуваної частини кривої на зрізаючу пряму ширша за відрізок зрізу (див. наступний рисунок праворуч).

Приклад 6. Дано коло з центром на початку координат і радіусом `3/2`. Побудувати параметричне рівняння контура кривої, отриманої обрізанням меншої частини кола по прямій $\frac{x}{2} + y + 1 = 0$.

Розв'язання. Відрізуваний сегмент кола розташований лівіше і нижче зрізаючої прямої. Щоб застосувати формули (13), її рівняння слід переписати у вигляді: $-\frac{x}{2} - y - 1 = 0$.

Спочатку створіть функцію `cutline(linekoef,curveeq)`, яка повертатиме символьні вирази (13). Вона прийматиме список коефіцієнтів рівняння зрізаючої прямої (з правильними знаками) і список двох символьних виразів – параметричного рівняння оброблюваної кривої.

```
import numpy as np
import matplotlib.pyplot as plt
from sympy import symbols, lambdify, Abs, sqrt, cos, sin, simplify
def mn(x,y): return (x+y-Abs(x-y))/2
def cutline(linekoef, curveeq):
    """
    Конструювання параметричного рівняння кривої, зрізаної по прямій
    a*x+b*y+c=0, де linekoef=[a,b,c]. Рівняння кривої передаються в
    символьних виразах списку curveeq=[x(t),y(t)].
    Напрям вектора (a,b) у бік відрізуваної ділянки.
    """
    a,b,c=linekoef
    nrm=sqrt(a**2+b**2)
    a,b,c=a/nrm,b/nrm,c/nrm
```

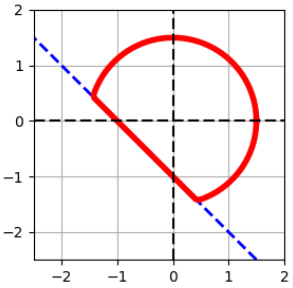
```
    x,y=curveeq
    return [simplify(a*(mn(a*x+b*y+c,0)-c)-b*(-b*x+a*y)),
              simplify(b*(mn(a*x+b*y+c,0)-c)+a*(-b*x+a*y))]
```

Задайте входові дані (коефіцієнти зрізаючої прямої і символьні вирази параметричного рівняння кола) та згенеруйте параметричні рівняння зрізаної кривої.

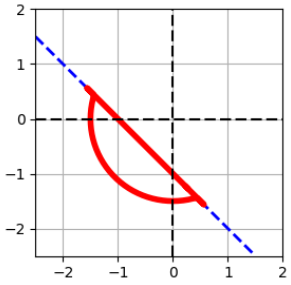
```
a,b,c,r=-1,-1,-1,3/2
t = symbols('t')
X,Y=cutline([a,b,c],[r*cos(t),r*sin(t)])
print('x=',X,'\n','y=',Y)
x=-0.375*sin(t)+1.125*cos(t)+
       0.25*sqrt(2)*Abs(1.5*sin(t+pi/4)+0.5*sqrt(2))-0.25
y=1.125*sin(t)-0.375*cos(t)+
       0.25*sqrt(2)*Abs(1.5*sin(t+pi/4)+0.5*sqrt(2))-0.25
```

Побудуйте зображення зрізаючої прямої і, по отриманим рівнянням, графік кривої.

```
Xc=lambdify(t, X, "numpy")
Yc=lambdify(t, Y, "numpy")
fig,ax = plt.subplots(1,1)
t_=np.linspace(-2.5,2,101)
Xt,Yt=np.meshgrid(t_,t_)
ax.contour(Xt, Yt,a*Xt+b*Yt+c, [0], linewidths=2,colors='b',
                                                linestyles='dashed')
T=np.linspace(0,2*np.pi,101)
ax.plot(Xc(T),Yc(T),'r',lw=4)      # наступний рисунок ліворуч
ax.axhline(0,color='k',dashes=(5,2))
ax.axvline(0,color='k',dashes=(5,2))
ax.set_aspect(1)
ax.grid(True)
```



Зрізаюча пряма $-\frac{x}{2}-y-1=0$.



Зрізаюча пряма $\frac{x}{2}+y+1=0$.

Правий рисунок отримано заміною знаків коефіцієнтів `a,b,c`.

■

Досі більшість перетворень ми виконували відносно координатних площин $z = Const$ (чи прямих). Перетворення аналогічно можна виконати відносно інших координатних напрямків. Та якщо є потреба виконати їх відносно похилої прямої або площини, то спочатку перетворення можна виконати відносно якогось координатного напрямку, а потім виконати поворот.

Нагадаємо, що точка P з координатами (x, y) після повороту проти годинникової стрілки навколо початку координат на кут α, матиме координати x', y', обчислювані за формулами

$$x' = x \cdot \cos\alpha - y \cdot \sin\alpha \ , \ y' = x \cdot \sin\alpha + y \cdot \cos\alpha \ , \qquad (14)$$

Перетворення повороту (14) можна застосувати до кожної точки параметричної кривої, створивши, тим самим, рівняння повернутої кривої.

Приклад 7. Змоделювати параметричні рівняння кола з вирізаною «скибкою» .

Розв'язання. Для цього спочатку застосуємо аналог формули (10), записаний для кривих. Якщо дано параметричну криву L_1 з рівняннями $x(u)$, $y(u)$ і криву L_2 з явним рівнянням $y = f(x)$, то параметричне рівняння кривої L_1 з «вм'ятиною» у формі L_2 матиме вигляд

$$\begin{aligned} X(u) &= x(u) && \text{Для параметричної кривої } L_1\colon x = x(u),\ y = y(u) \\ Y(u) &= mn\big(y(u), f(x(u))\big) && \text{Для явно заданої кривої } L_2\colon y = f(x). \end{aligned} \qquad (15)$$

В нашому прикладі параметричною кривою буде одиничне коло, а рівнянням «вм'ятини зверху» $y = k \cdot |x|$ (покладемо $k = 2$). Після конструювання параметричних рівнянь (15) можна буде виконати поворот на кут $\frac{\pi}{4}$ за годинниковою стрілкою.

```
import numpy as np
import matplotlib.pyplot as plt
from sympy import symbols,lambdify,Abs,simplify,sin,cos,pi
plt.close('all')
def mn(x,y): return (x+y-Abs(x-y))/2
```

Запишемо функції параметричних рівнянь одиничного коло з вм'ятиною $y = 2\,|x|$ зверху.

```
def x(u): return cos(u)
def y(u): return mn(sin(u),2*Abs(x(u)))
u = symbols('u');
print(y(u))
```

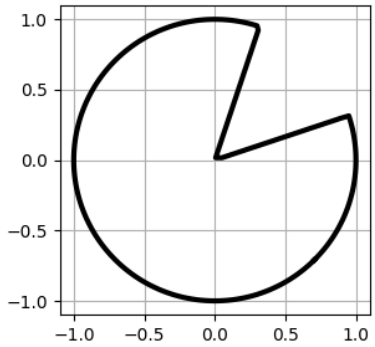
```
sin(u)/2-Abs(sin(u)-2*Abs(cos(u)))/2+Abs(cos(u))
```

Запишемо параметричне рівняння кола з вм'ятиною, повернутого на кут $\frac{\pi}{4}$ за годинниковою стрілкою.

```
alpha=-pi/4
X=cos(alpha)*x(u)-sin(alpha)*y(u)
Y=sin(alpha)*x(u)+cos(alpha)*y(u)
```

Побудуємо графік отриманої кривої.

```
xdent=lambdify((u),X,"numpy")
ydent=lambdify((u),Y,"numpy")
u_=np.linspace(0,2*np.pi,200)
x_=xdent(u_)
y_=ydent(u_)
fig,ax = plt.subplots(1,1)
ax.plot(x_, y_, 'k',lw=3)
ax.set(xlim=(-1.1 ,1.1), ylim=(-1.1,1.1), aspect=1)
ax.grid()
```



■

В загальному випадку, якщо потрібно виконати перетворення кривої відносно похилої прямої, то слід:

1. сумісти з похилою прямою одну з координатних осей і записати всі рівняння в нових осях `X′,Y′`;
2. застосувати бажане перетворення, працюючи в новій системі координат `X′,Y′`;
3. повернутися від рівнянь в системі `X′,Y′` до рівнянь в старій системі координат `X,Y`.

Запропонований підхід спрацьовує і для поверхонь, але матриця повороту тепер буде розміром `3 x 3`.

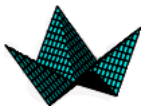
Приклад 8. Побудувати зображення поверхні кулі, у якої зроблено два бокових вирізи в ортогональних напрямках  .

Розв'язання. Формули (10) призначені для побудови на параметричній поверхні вирізу зверху. Щоб вони були збоку поверхню потім можна повернути.

```
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.colors as mc
from scipy.spatial.transform import Rotation as R
plt.close('all')
```

Створіть функцію $f(x,y)$, графік якої представлятиме поверхню вирізу.

```
def mn(x,y): return (x+y-np.abs(x-y))/2
def f(x,y): return mn(np.abs(x),np.abs(y)
```



Створіть рівняння $x(u,v)$, $y(u,v)$, $z_{dent}(u,v)$ поверхні кулі з таким вирізом зверху.

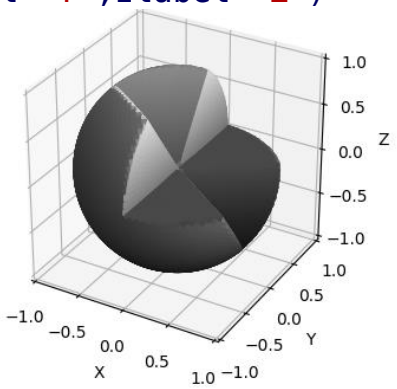
```
def xsp(u,v): return np.cos(u)*np.cos(v)
def ysp(u,v): return np.sin(u)*np.cos(v)
def zsp(u,v): return np.sin(v)
def zdent(u,v):
    return mn(zsp(u,v),f(xsp(u,v),ysp(u,v)))
```

Генеруйте сітку точок на поверхні отриманого тіла.

```
u,v=np.mgrid[0:2*np.pi:121j,-np.pi/2:np.pi/2:61j]
X=xsp(u,v)
Y=ysp(u,v)
Z=zdent(u,v)
```

Створіть об'єкт Rotation повороту навколо осі Y на кут π/4.

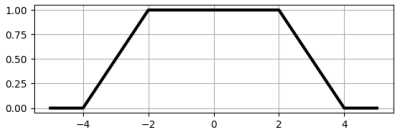
```
Axis =np.array([0,1,0])
theta = np.pi/4
axis = Axis / np.linalg.norm(Axis)
rot = R.from_rotvec(theta * axis)
```
Кожна трійка чисел `X[i,j],Y[i,j],Z[i,j]` являє радіус-вектор точок поверхні, який слід повернути відповідно до побудованого об'єкту `rot`. Для цього ми створимо тривимірний масив P.
```
P=np.stack([X,Y,Z],axis=2)
```
Оскільки метод `Rotation.apply()` можна застосовувати лише до одного вектора розміром (3,) або до масиву N векторів розміром (N,3), то метод `apply()` застосовуватимемо окремо до кожної групи векторів `P[i]`.
```
newP=np.empty_like(P)
for i in range(len(P)):
    newP[i]=rot.apply(P[i])
```
Перш ніж будувати зображення поверхні по сітці точок з координатами в масиві `newP`, створимо допоміжний масив кольорів `C` для розфарбовування поверхні.
```
cNorm=mc.Normalize();
Fcolors=cNorm(newP[:,:,2]);
C=plt.cm.gray(Fcolors)
```
Тепер побудуйте зображення поверхні.
```
fig = plt.figure()
ax = fig.add_subplot(111,projection='3d')
ax.plot_surface(newP[:,:,0], newP[:,:,1], newP[:,:,2],
                facecolors=C, shade=True, rstride=1, cstride=1)
ax.set(xlim=(-1,1),ylim=(-1,1),zlim=(-1,1))
ax.set_box_aspect(aspect =(1,1,1))
ax.set(xlabel='X',ylabel='Y',zlabel='Z')
```



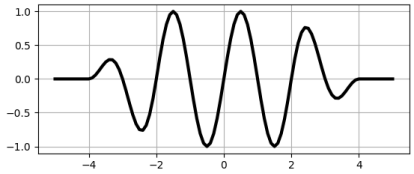
■

Інколи бажано побудувати на відрізку фінітну функцію. Наприклад, створимо з синусоїди $\sin \pi x$ фінітну, яка на відрізку [-2,2] співпадає з $\sin \pi x$, а на ділянках [-4,-2] і [2,4] плавно спадає до нуля. Для цього достатньо домножити $\sin \pi x$ на ламану $l(x)$, графік якої наведено на наступному рисунку ліворуч, а графік результату $y(x) = l(x) \cdot \sin \pi x$ – праворуч.

$l(x)$

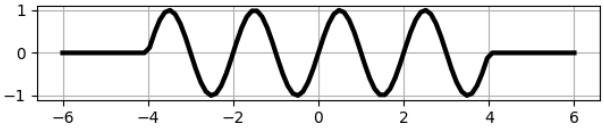


$y(x) = l(x) \cdot \sin \pi x$

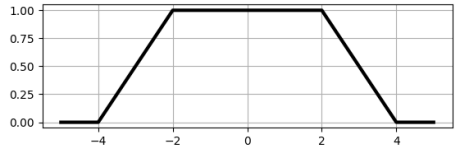
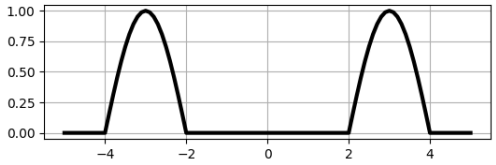
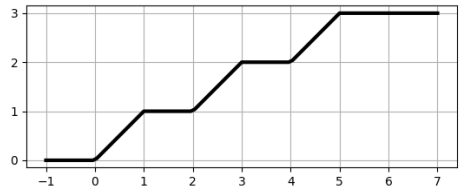
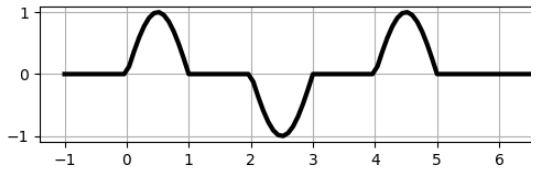


Сподіваємося, що читач самостійно зможе генерувати рівняння ламаної $l(x)$, використовуючи формулу Бернштейна або нашу функцію `broken_line()` (код наведено в п. 10.1 цієї глави).

Якщо потрібно виділити фіксовану ділянку функції з відрізка $[x_1,\ x_2]$, а поза ним нову функцію покласти сталими значеннями $f(x_i)$ $(i = 1{,}2)$, то можна застосувати, наприклад, формулу $f\big(x_1 + \Pi(x, x_1, x_2 - x_1)\big)$. Це випливає з міркувань, наведених для пояснення формули (4). Наприклад, щоб виділити фіксовану ділянку синусоїди $\sin \pi x$ на відрізку [-4,4] можна створити функцію $\sin \pi\big(-4 + \Pi(x, -4{,}8)\big)$. Вона матиме наступний графік.



Взагалі-то, завжди можна побудувати суперпозицію довільної функції $f(x)$ і ламаної $l(x)$, що створюватиме неперервну кускову функцію $f\big(l(x)\big)$. Нижче дано приклади суперпозиції синусоїди з ламаними $l(x)$, графіки яких наведено в лівому стовпці таблиці.

$l(x)$	$\sin\big(\pi\, l(x)\big)$
	
	

На останньому рисунку ліворуч кожній горизонтальній ділянці $l = l_0$ ламаної відповідає горизонтальна ділянка $\sin(\pi\, l_0)$ функції $\sin\big(\pi\, l(x)\big)$. Кожному шматку ламаної з одиничним нахилом відповідає ділянка початкової функції.

Наведені міркування допомагають будувати доволі складні поверхні

Приклад 9. Згенерувати параметричні рівняння поверхні кулі одиничного радіуса з вирізаною «скибкою».

Р о з в ' я з а н н я.

```
import numpy as np
import matplotlib.pyplot as plt
plt.close('all')
Pi=np.pi
```

Спочатку змоделюємо параметричні рівняння кола з вирізаною «скибкою»  . В них при зростанні параметра t від 0 до π/2 (на горизонтальному відрізку) абсциса `x` зменшується від 1 до 0, а ордината `y` дорівнює 0. На вертикальному відрізку параметр t зростатиме від π/2 до π, абсциса дорівнюватиме 0, а ордината збільшуватиметься від 0 до 1. Далі при $\pi \le t \le 5\pi/2$ параметрчні рівняння повинні давати точки на одиничному колі. Таку поведінку можна змоделювати

рівняннями $x = \cos u_1(t)$, $y = \sin u_2(t)$ з функціями $u_1(t)$, $u_2(t)$, графіки яких наведені на наступному рисунку ліворуч (на ділянці $t \geq \pi$ лінії співпадають).

За формулою Бернштейна створимо рівняння ламаних $u_1(t)$, $u_2(t)$, побудуємо їх графіки і криву по параметричним рівнянням $x = \cos u_1(t)$, $y = \sin u_2(t)$.

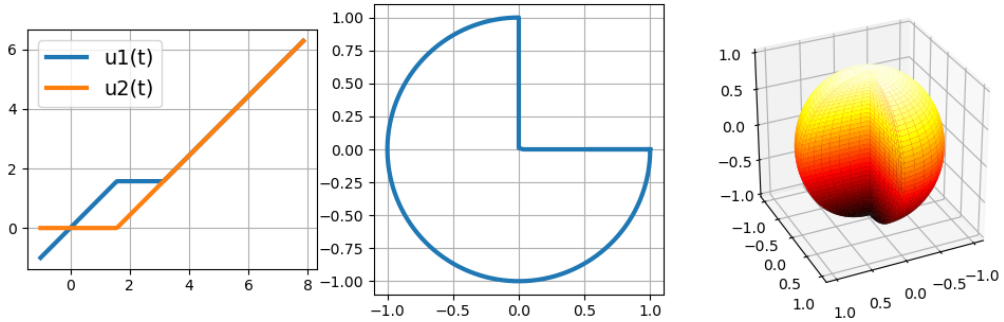
```
def u1(t): return t-Pi/4-np.abs(t-Pi/2)/2+np.abs(t-Pi)/2
def u2(t): return np.abs(t-Pi/2)/2+t/2-Pi/4
t=np.linspace(-1,5*Pi/2,200)
fig=plt.figure(figsize=(12,4),
                          num='Поверхня кулі з вирізаною скибкою')
ax1 = fig.add_subplot(131)
l1=ax1.plot(t,u1(t),t,u2(t),lw=3)        # наступний рисунок ліворуч
ax1.legend(l1,('u1(t)','u2(t)'),loc='upper left',fontsize=14)
ax1.set_aspect('equal')
ax1.grid(True)
ax2 = fig.add_subplot(132)
ax2.plot(np.cos(u1(t)), np.sin(u2(t)), lw=3)    # наст. рис. всередині
ax2.set_aspect('equal')
ax2.grid(True)
```

Використовуючи функції $u_1(u)$, $u_2(u)$ замість парамера u в параметричних рівняннях кулі, тобто

$$x(u,v) = \cos u_1(u) \cdot \cos v$$
$$y(u,v) = \sin u_2(u) \cdot \cos v$$
$$z(u,v) = \sin v$$

ми отримаємо параметричні рівняння поверхні кулі з вирізаною «скибкою».

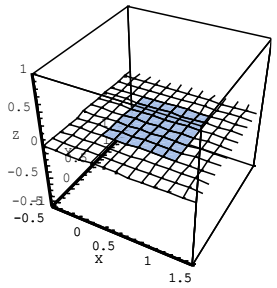
```
ax3 = fig.add_subplot(133,projection='3d')
u,v=np.mgrid[0:5*Pi/2:100j,-Pi/2:Pi/2:50j]
X=np.cos(u1(u))*np.cos(v)
Y=np.sin(u2(u))*np.cos(v)
Z=np.sin(v)
ax3.plot_surface(X,Y,Z,cmap='hot')
ax3.set_box_aspect(aspect =(1,1,1))
ax3.view_init(30,65)
```



Зауваження. Використовуючи функції $u_1(t)$, $u_2(t)$ в параметричних рівняннях інших поверхонь обертання, ви можете конструювати рівняння таких поверхонь з вирізаними «скибками».

Вправа. В останньому прикладі замість сфери використайте конус обертання і отримайте рівняння конуса з вирізом.

Наведемо ще одне корисне зауваження. При параметричному заданні поверхонь ми виходимо з рівнянь $x = x(u,v)$, $y = y(u,v)$, $z = z(u,v)$, де u, v – параметри, що змінюються в деяких інтервалах. Але використовуючи функції (1) – (3), можна отримувати рівняння, які визначають частину поверхні незалежно від довжини інтервалів зміни параметрів u, v, якщо ці інтервали достатньо широкі. Наприклад, рівняння $x = \Pi(u, 0{,}1)$, $y = \Pi(v, 0{,}1)$, $z = 0$ при $-\infty < u < \infty$, $-\infty < v < \infty$ визначають квадратну зону $0 \le x \le 1,\ 0 \le y \le 1$ на площині $z = 0$.



У цьому сенсі рівняння $x = \Pi(u, 0{,}1)$, $y = \Pi(v, 0{,}1)$ можна називати параметричними рівняннями квадрата, оскільки при досить широких межах зміни параметрів u, v область не змінюватиметься зі збільшенням інтервалів зміни u, v. Схожим чином можна будувати параметричні рівняння багатьох плоских областей, заміняючи в їх звичайних рівняннях параметри $u_0 \le u \le u_1$, $v_0 \le v \le v_1$ на вирази $u_0 + \Pi(u, u_0, u_1 - u_0)$, $v_0 + \Pi(v, v_0, v_1 - v_0)$ (якщо одна з меж нескінченна, наприклад $u_0 = -\infty$, то замість $u_0 + \Pi(u, u_0, u_1 - u_0)$ слід використовувати $Q_l(u, u_1)$). Такі рівняння $x = x(u,v)$, $y = y(u,v)$ (без урахування третьої координати) ми називатимемо параметричними рівняннями плоскої області. Саме незалежність форми від інтервалів зміни параметрів (якщо ці інтервали досить широкі) дає підставу називати їх рівняннями області. Фактично пара цих функцій здійснює відображення всієї площини (u, v) на обмежену зону площини (x, y). Воно не є однозначним, що для математики дуже незвично, однак зручно для геометричного моделювання.

Повернемося до параметричних рівнянь квадрата $x = \Pi(u, 0{,}1)$, $y = \Pi(v, 0{,}1)$. У цьому прикладі зону квадрата ми могли зобразити на обох площинах (x, y) або (u, v) - позначимо ці області D_{xy} та D_{uv}. Тоді функції, що реалізують рівняння області, відображають точки області D_{uv} площини (u, v) в «ті ж самі» точки області D_{xy} площини (x, y). Точки межі області D_{uv} при цьому відображаються у точки межі D_{xy}. Однак зовнішні точки області D_{uv} площини (u, v) відображаються також у точки межі D_{xy}. Якщо в площині (u, v) побудувати параметричну криву $u = u(t), v = v(t)$, що цілком охоплює область D_{uv}, то функції $x = \Pi(u(t), 0{,}1)$, $y = \Pi(v(t), 0{,}1)$ даватимуть параметричне рівняння контура квадрата. Наприклад, в площині (u, v) візьмемо криву $u = 2\cos t$, $v = 2\sin t$ $(0 \le t \le 2\pi)$. Тоді параметричним рівнянням контура квадрата $0 \le x \le 1,\ 0 \le y \le 1$ буде $x = \Pi(2\cos t\,, 0{,}1)$, $y = \Pi(2\sin t\,, 0{,}1)$, При чому деяким точкам на контурі квадрата відповідатимуть цілі інтеравали зміни t, але для побудови зображення кривої це несуттєво.

Аналогічно, можна будувати параметричні рівняння тривимірних тіл і їх поверхонь.

Приклад 10. Згенерувати параметричні рівняння поверхні куба $-1 \le x \le 1$, $-1 \le y \le 1$, $-1 \le z \le 1$.

Розв'язання. Параметричними рівняннями куба (тривимірного тіла) будуть

$$x = -1 + \Pi(u, -1, 2),\ y = -1 + \Pi(v, -1, 2),\ z = -1 + \Pi(w, -1, 2).$$

Підставивши замість u, v, w в ці вирази параметричне рівняння охоплюючої поверхні, наприклад, сфери радіуса 2, отримаємо параметричне рівняння поверхні куба.

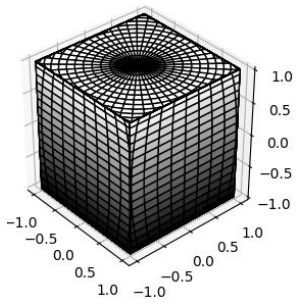
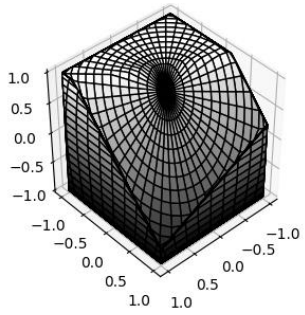
```
import numpy as np
import matplotlib.pyplot as plt
from sympy import symbols,lambdify,Abs,simplify,sin,cos
plt.close('all')
def PP(x,a,w): return (w+Abs(x-a)-Abs(x-a-w))/2
def usp(u,v): return 2*cos(t)*cos(tau)
def vsp(u,v): return 2*sin(t)*cos(tau)
def wsp(u,v): return 2*sin(tau)
t,tau = symbols('t tau');
x=-1+PP(usp(t,tau),-1,2); print('x=',x)
x=-Abs(2*cos(t)*cos(tau)-1)/2+Abs(2*cos(t)*cos(tau)+1)/2
y=-1+PP(vsp(t,tau),-1,2); print('y=',y)
y=-Abs(2*sin(t)*cos(tau)-1)/2+Abs(2*sin(t)*cos(tau)+1)/2
z=-1+PP(wsp(t,tau),-1,2); print('z=',z)
z=-Abs(2*sin(tau)-1)/2+Abs(2*sin(tau)+1)/2
```

Тепер побудуємо зображення поверхні по її параметричним рівнянням (наступний рисунок ліворуч).

```
t_,tau_=np.mgrid[0:2*np.pi:50j,-np.pi/2:np.pi/2:50j]
X=lambdify((t,tau),x,"numpy")(t_,tau_)
Y=lambdify((t,tau),y,"numpy")(t_,tau_)
Z=lambdify((t,tau),z,"numpy")(t_,tau_)
fig = plt.figure()
ax = fig.add_subplot(111,projection='3d')
ax.plot_surface(X,Y,Z,cmap='gray',edgecolor='k')
ax.set_box_aspect(aspect =(1,1,1))
```

Працювати з параметричними рівняннями отриманої поверхні можна так само, як і з усіма іншими. Наприклад, генеруємо рівняння поверхні куба, зрізаного по похилій площині $z = 0.75 - 0.5\,x - y$ (наступний рисунок праворуч).

```
def mn(x,y): return (x+y-Abs(x-y))/2
def pl(x,y): return 0.75-0.5*x-y
zpl=mn(z,pl(x,y))
Zpl=lambdify((t,tau),zpl,"numpy")(t_,tau_)
fig = plt.figure()
ax = fig.add_subplot(111,projection='3d')
ax.plot_surface(X,Y,Zpl,cmap='gray',edgecolor='k')
ax.set_box_aspect(aspect =(1,1,1))
```

■

Конкретизуємо запропонований метод. Нехай є рівняння тривимірного тіла (інколи їх простіше побудувати, ніж рівняння поверхні):

$$x = x(\alpha,\beta,\gamma)\,,\ \ y = y(\alpha,\beta,\gamma)\,,\ \ z = z(\alpha,\beta,\gamma) \tag{16}$$

$$\alpha_0 \le \alpha \le \alpha_1\,,\ \beta_0 \le \beta \le \beta_1\,,\ \gamma_0 \le \gamma \le \gamma_1. \tag{17}$$

Створимо з них параметричні рівняння тривимірної області, виконавши в (16) заміни

$$\alpha \to \alpha_0 + \Pi(\alpha,\alpha_0,\alpha_1-\alpha_0),\ \beta \to \beta_0 + \Pi(\beta,\beta_0,\beta_1-\beta_0),\ \gamma \to \gamma_0 + \Pi(\gamma,\gamma_0,\gamma_1-\gamma_0).$$

Тоді, щоб записати параметричні рівняння поверхні тіла, в просторі параметрів α,β,γ слід записати параметричні рівняння поверхні $\alpha = \alpha(u,v)$, $\beta = \beta(u,v)$, $\gamma = \gamma(u,v)$ $(u_0 \le u \le u_1\,, v_0 \le v \le v_1)$, що повністю охоплює прямокутний паралелепіпед (17), і підставити їх в модифіковані рівняння тривимірної області

$$x = x(\alpha_0 + \Pi(\alpha(u,v),\alpha_0,\alpha_1-\alpha_0),\beta_0 + \Pi(\beta(u,v),\beta_0,\beta_1-\beta_0),\gamma_0 + \Pi(\gamma(u,v),\gamma_0,\gamma_1-\gamma_0))$$
$$y = y(\alpha_0 + \Pi(\alpha(u,v),\alpha_0,\alpha_1-\alpha_0),\beta_0 + \Pi(\beta(u,v),\beta_0,\beta_1-\beta_0),\gamma_0 + \Pi(\gamma(u,v),\gamma_0,\gamma_1-\gamma_0))$$
$$z = z(\alpha_0 + \Pi(\alpha(u,v),\alpha_0,\alpha_1-\alpha_0),\beta_0 + \Pi(\beta(u,v),\beta_0,\beta_1-\beta_0),\gamma_0 + \Pi(\gamma(u,v),\gamma_0,\gamma_1-\gamma_0))$$

Отримані вирази являтимуть параметричні рівняння поверхні тіла (відносно параметрів u,v). Слід лише пам'ятати, що одній точці (x,y,z) поверхні може відповідати не одна точка u,v, а кілька, або навіть цілі зони з прямокутника $u_0 \le u \le u_1\,, v_0 \le v \le v_1$.

Приклад 11. Згенерувати параметричні рівняння поверхні «розрізаної з боку» труби з внутрішнім радіусом $r_0 = 3$, зовнішнім $r_n = 6$, довжиною/висотою $H = 10$, а бокова частина, що залишається, обмежена вертикальними площинами з полярними кутами $\alpha_0 = 0$ і $\alpha_1 = \frac{3}{2}\pi$.

Р о з в ' я з а н н я . Використовуючи замість α,β,γ ідентифікатори r,α,h, рівняння розрізаної збоку труби (тривимірного тіла) можна записати у вигляді

$$x = r\cos\alpha\ , y = r\sin\alpha\ ,\ z = h\ ,$$

де $r_0 \le r \le r_1,\ \alpha_0 \le \alpha \le \alpha_1,\ 0 = h_0 \le h \le h_1 = H$. Тоді рівняння тривимірної області матимуть вигляд

$$\begin{aligned} x &= \big(r_0 + \Pi(r,r_0,r_1-r_0)\big)\cos\big(\alpha_0 + \Pi(\alpha,\alpha_0,\alpha_1-\alpha_0)\big)\ ,\\ y &= \big(r_0 + \Pi(r,r_0,r_1-r_0)\big)\sin\big(\alpha_0 + \Pi(\alpha,\alpha_0,\alpha_1-\alpha_0)\big)\ ,\\ z &= h_0 + \Pi(h,h_0,h_1-h_0) \end{aligned} \tag{18}$$

Взявши параметричні рівняння поверхні кулі, що повністю охоплюватиме паралелепіпед $3 \le r \le 6, 0 \le \alpha \le \frac{3}{2}\pi, 0 \le h \le 10$, і підставивши їх в (18) замість

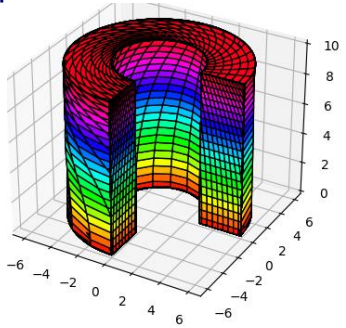
r, α, h, ви отримаєте рівняння поверхні труби (розрізаної збоку, якщо $[\alpha_0, \alpha_1] \in [0, 2\pi]$).

Для реалізації вказаної послідовності конструювання функцій, що являтимуть параметричні рівняння поверхні розрізаної з боку труби, створимо функцію `cut_out_cylinder(r0,rn,H,amax)`, яка в якості аргументів прийматиме внутрішній `r0` і зовнішній `rn` радіуси, висоту труби H, і `amax` – межу діапазона полярного кута $0 \le \alpha \le \alpha_{max}$.

```
import numpy as np
def PP(x,a,w): return (w+np.abs(x-a)-np.abs(x-a-w))/2
def cut_out_cylinder(r0,rn,H,amax):
    def x(r,a,h): return (r0+PP(r,r0,rn-r0))*np.cos(PP(a,0,amax))
    def y(r,a,h): return (r0+PP(r,r0,rn-r0))*np.sin(PP(a,0,amax))
    def z(r,a,h): return PP(h,0,H)
    rah=np.floor(np.sqrt(rn**2+(amax)**2+H**2))+1
    def sphera(u,v):
        return [rah*np.cos(u)*np.cos(v),
                rah*np.sin(u)*np.cos(v),
                rah*np.sin(v)]
    def X(u,v): return x(*sphera(u,v))
    def Y(u,v): return y(*sphera(u,v))
    def Z(u,v): return z(*sphera(u,v))
    return [X,Y,Z]
```

Протестуємо функцію `cut_out_cylinder()`, побудувавши зображення поверхні тіла по згенерованим рівнянням.

```
if __name__ == '__main__':
    import matplotlib.pyplot as plt
    r0,rn,H,amax=3,6,10,3*np.pi/2
    u_=np.linspace(0,2*np.pi,200)
    v_ = np.linspace(-np.pi/2,np.pi/2,150)
    U,V=np.meshgrid(u_,v_)
    flist=cut_out_cylinder(r0, rn, H, amax)
    Xd,Yd,Zd=[f(U,V) for f in flist]
    fig= plt.figure()
    ax = fig.add_subplot(111, projection='3d')
    surf=ax.plot_surface(Xd,Yd,Zd,rstride=2,cstride=1,
                         linewidth=1,cmap='hsv',edgecolor='k')
    ax.set_box_aspect(aspect =(2*rn,2*rn,H))
```



Збережіть цей сценарій в файлі `surface.py`, бо функція `cut_out_cylinder()` нам ще знадобиться.

Зауваження. Поверхня вийде «красивішою», якщо її будувати як поверхню обертання вертикального прямокутного перетину з подальшим вирізанням «скибки», як в прикладі 9. Зробіть це самостійно.

10.4. Неявні рівняння меж складених областей.

При розв'язанні прикладних задач часто доводиться зображати досліджувану область по системі обмежень/нерівностей, накладених на координати точок. Одним з можливих підходів до цієї проблеми є застосування маскованих масивів. Інший підхід полягає в конструюванні неявних рівнянь меж областей (на площині $F(x,y)=0$ або $F(x,y,z)=0$ в просторі). Тоді за допомогою функції `matplotlib.pyplot.contour()` та `mayavi.mlab.contour3d()` їх можна досить якісно зобразити. Окрім того, інколи виникають ситуації, коли неявне рівняння межі області (лінії або поверхні) конче потрібно, наприклад, для побудови базисних функцій в методі Гальоркіна.

У 1963 р. Харківський математик В.Л. Рвачов запропонував спосіб побудови неявних рівнянь кривих та поверхонь, які є межами складених областей [5]. Нехай Ω область n-вимірного простору. Ідея побудови неявного рівняння межі S області Ω полягала в методиці конструювання функції $\omega(x_1,\dots,x_n)$, яка строго позитивна всередині області, негативна поза нею і обертається на нуль на її межі. Очевидно, що тоді $\omega(x_1,\dots,x_n)=0$ буде неявним рівнянням межі області.

Нехай в тривимірному (двовимірному) просторі дано області $\Omega_i \quad (i=1,2)$. Для кожної з яких відома функція $\omega_i(x,y,z)$ (в 2D $\omega_i(x,y)$), яка позитивна всередині області, негативна - зовні, і дорівнює нулю на її межі. Функції ω_i, що мають вказані властивості, ми будемо називати функціями ідентифікації або просто ідентифікаторами областей Ω_i. Очевидно, що неявне рівняння поверхні області Ω_i можна тоді записати у вигляді $\omega_i(x,y,z)=0$. Крім того, побудуємо функції

$$\begin{aligned} ir(u,v) &= \frac{1}{2}(u+v-|u-v|) && \text{(region intersection);} \\ ur(u,v) &= \frac{1}{2}(u+v+|u-v|) && \text{(region union);} \\ dr(u,v) &= \frac{1}{2}(u-v-|u+v|) && \text{(region difference).} \end{aligned} \qquad (1)$$

Ці функції називатимемо виконуючими функціями операцій перетину, об'єднання і різниці областей, розглядуваних як множини точок. Функція $ir(u,v)$ позитивна в зоні першого квадранта $x>0, y>0,$ дорівнює нулю на його межах $y=0\ (x\geq 0)$, $x=0\ (y\geq 0)$, і негативна для інших значень x,y (фактично, це функція ідентифікації $\omega(x,y)$ області першого квадранта). Функція $ur(u,v)$ негативна у внутрішніх точках першого квадранта $x>0, y>0$, обертається на нуль на його межах $y=0\ (x\geq 0)$, та $x=0\ (y\geq 0)$, і позитивна в інших точках (x,y) площині XY (це функція ідентифікації зони, отриманої об'єднанням 2, 3 і 4 квадрантів). Функція $dr(u,v)$ позитивна в зоні 4 квадранта $x>0, y<0,$

дорівнює нулю лише на його межах і від'ємна в інших точках (x, y) площині XY (функція ідентифікації області четвертого квадранта).

Тоді справедливі наступні твердження:

1. Якщо $\Omega_{ir} = \Omega_1 \cap \Omega_2$, то функція $\omega_{ir}(x, y, z) = ir\big(\omega_1(x, y, z), \omega_2(x, y, z)\big)$ буде ідентифікатором області Ω_{ir} (в 2D $\omega_{ir}(x, y) = ir\big(\omega_1(x, y), \omega_2(x, y)\big)$);
2. Якщо $\Omega_{ur} = \Omega_1 \cup \Omega_2$, то $\omega_{ur}(x, y, z) = ur\big(\omega_1(x, y, z), \omega_2(x, y, z)\big)$ буде функцією ідентифікації області Ω_{ur} (в 2D випадку $\omega_{ur}(x, y) = ur\big(\omega_1(x, y), \omega_2(x, y)\big)$);
3. Якщо $\Omega_{\mathrm{dr}} = \Omega_1 \backslash \Omega_2$, то функція $\omega_{dr}(x, y, z) = dr\big(\omega_1(x, y, z), \omega_2(x, y, z)\big)$ буде ідентифікатором області Ω_{dr} (в 2D випадку $\omega_{dr}(x, y) = dr\big(\omega_1(x, y), \omega_2(x, y)\big)$).

Доведення цих формул виконується обчисленням знаку/значення результівної функцій $\omega_{\_r}(x, y, z)$ в усіх можливих місцях розташування точки (x, y, z): всередині одночасно Ω_1, Ω_2; зовні Ω_1, але всередині Ω_2; на межі Ω_1 і всередині Ω_2, і так далі.

Зауваження. Замість формул (1) можна обирати багато інших, наприклад,

$$
\begin{aligned}
&ir_\alpha(u, v) = \frac{1}{1+\alpha}\left(x + y - \sqrt{x^2 + y^2 - 2\,\alpha\, x\, y}\right), && \text{(region intersection);}\\
&\text{при будь-якому } -1 < \alpha \le 1. \\
&ur_\alpha(u, v) = \frac{1}{1+\alpha}\left(x + y + \sqrt{x^2 + y^2 - 2\,\alpha\, x\, y}\right), && \text{(region union);}\\
&\text{при будь-якому } -1 < \alpha \le 1. \\
&dr_\alpha(u, v) = \frac{1}{1+\alpha}\left(x - y - \sqrt{x^2 + y^2 - 2\,\alpha\, x\, y}\right), && \text{(region difference).}\\
&\text{при будь-якому } -1 < \alpha \le 1.
\end{aligned}
\qquad (2)
$$

Застосовуючи твердження 1 – 3, можна будувати неявні рівняння меж майже будь-яких областей.

Нехай тіло Ω, як множина точок у просторі (або на площині), утворюється за допомогою деяких булевих операцій (перетин, об'єднання, доповнення, тощо) над підобластями Ω_i, що мають простішу ніж Ω форму. Припустимо що для кожної з підобластей Ω_i відома функція ідентифікації $\omega_i(x, y, z)$. Тоді суперпозиція функцій (1) і ідентифікаторів ω_i областей Ω_i дозволяє побудувати функцію ідентифікації результівної області Ω. Наприклад, якщо $\Omega = (\Omega_1 \cap \Omega_2 \cup \Omega_3) \cap \bar{\Omega}_4$, то $\omega_\Omega = ir(ur(ir(\omega_1, \omega_2), \omega_3), -\omega_4)$, де для стислості ми опустили аргументи x, y, z всіх функцій ω_i $(i = 1,2,3,4)$. Як наслідок, неявне рівняння поверхні тіла Ω матиме вигляд $\omega_\Omega(x, y, z) = 0$. Цей підхід працює також для плоских областей, функції ідентифікації яких залежать від двох просторових координат x, y. В такому випадку криві, побудовані по неявним рівнянням $\omega_\Omega(x, y) = 0$, зображатимуть контури плоских областей.

Приклад 1. Побудувати неявне рівняння контура прямокутника $-a \le x \le a$, $-b \le y \le b$.

Р о з в ' я з а н н я . Функція ω_1 ідентифікації смуги $-a \le x \le a$ $(a > 0)$ може бути записана, наприклад, у вигляді $\omega_1(x,y) = a - |x|$. Аналогічно, ідентифікатор смуги $-b \le y \le b$ $(b > 0)$ може мати вигляд $\omega_2(x,y) = b - |y|$. Використовуючи функцію $ir(\omega_1, \omega_2)$ ідентифікатор $\omega_r(x,y)$ області перетину смуг, тобто зони прямокутника, матиме вигляд

$$\omega_r(x,y) = \frac{1}{2}(\omega_1(x,y) + \omega_2(x,y) - |\omega_1(x,y) - \omega_2(x,y)|) =$$
$$= \frac{1}{2}\big(a + b - |x| - |y| - \big|\, a - b + |y| - |x|\big|\big).$$

Приклад 2. Побудувати неявне рівняння контура чверті одиничного кола, розташованого в першому квадранті $x \ge 0$, $y \ge 0$.

Р о з в ' я з а н н я . Подамо область чверті кола як перетин напівплощин $y \ge 0$ і $x \ge 0$ з одиничним колом $1 - x^2 - y^2 \ge 0$.

Функції ідентифікації цих зон слід задати наступними:

$$\omega_1(x,y) = y\,, \qquad \omega_2(x,y) = x\,, \qquad \omega_3(x,y) = 1 - x^2 - y^2.$$

Перетин напівплощин дає зону першого квадранта з функцією ідентифікації

$$\omega_{12}(x,y) = ir\big(\omega_1(x,y), \omega_2(x,y)\big) = ir(y,x) = (x + y - |x - y|)/2$$

Отриману зону ми перетинаємо з областю одиничного кола. Функція ідентифікації результівної області дорівнюватиме

$$\omega_{123}(x,y) = ir\big(\omega_{12}(x,y), \omega_3(x,y)\big)$$

Щоб не робити зайвої роботи, виконаємо обчислення за допомогою модуля символьної математики SymPy.

```
from sympy import symbols, Abs, simplify
from sympy.plotting import plot_implicit
x,y = symbols('x y')
```

Відповідно до (1) створіть виконуючу функцію $ir(u,v)$ операції перетинання.

```
def ir(u,v): return (u+v-Abs(u-v))/2
```

Створіть символьні вирази, які відповідають функціям ідентифікації $\omega_1, \omega_2, \omega_3$.

```
w1=y
w2=x
w3=1-x**2-y**2
```

Виконайте перетворення цих виразів відповідно до операції перетинання областей і надрукуйте результівну функцію ідентифікації.

```
w12=ir(w1, w2)
w=ir(w12, w3)
w=simplify(w); print(w)
-x**2/2+x/4-y**2/2+y/4-Abs(x-y)/4-
                    Abs(2*x**2+x+2*y**2+y-Abs(x-y)-2)/4+1/2
```

Отже неявним рівнянням контура чверті кола буде

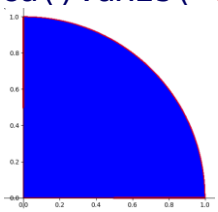
$$2 + x + y - 2x^2 - 2y^2 - |x - y| - \big|-2 + x + y + 2x^2 + 2y^2 - |x - y|\big| = 0$$

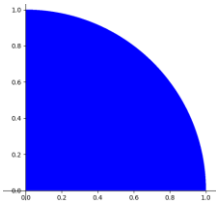
Множник $1/4$ є несуттєвим і ми його відкинули.

Використовуючи символьний вираз `w` функції ідентифікації, побудуйте зображення області і контура чверті кола.

```
p1=plot_implicit(w>=0,depth=2,show=False)    # зафарбована область
```

```
p2=plot_implicit(w, depth=2,line_color='red',show=False)      # контур
p1.extend(p2)
p1.show()                                                     # наступний рисунок ліворуч
plt.gcf().gca().axis('equal')
```

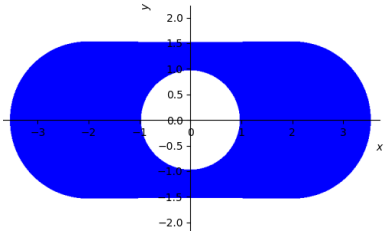




Зауваження. Якщо потрібно побудувати лише однокольорове зображення плоскої області, то ідентифікуючу функцію можна не створювати, оскільки до інструкції `sympy.plotting.plot_implicit(w,...)` в якості `w` можна передати булевий вираз (комбінацію «символьних» нерівностей). Наприклад, наступні інструкції будують зображення тієї ж зони (без контура).

```
w=And(x>=0, y>=0, 1-x**2-y**2>=0)
plot_implicit(w,(x,-0.1,1.1),(y,-0.1,1.1))   # попередній рис. праворуч
```

Приклад 3. Побудувати неявне рівняння зони «стадіону з круговим отвором».



Розв'язання. Нехай розміри фігури будуть таким, як на наведеному вище рисунку: висота 3, ширина 7, радіус внутрішнього кола $r = 1$, абсциси центрів напівкіл, що утворюють ліву і праву межу, дорівнюють ± 2. Ввівши параметри $a = 2$ (абсциса центра правого півкола), $b = 1.5$ (напіввисота), бажану область Ω можна подати як послідовну комбінацію булевих операцій з наступними областями:

Ω_1 (смуга)	$-a \le x \le a$	$\omega_1(x,y) = a^2 - x^2$
Ω_2 (смуга)	$-b \le y \le b$	$\omega_2(x,y) = b^2 - y^2$
$\Omega_{12} = \Omega_1 \cap \Omega_2$	Прямокутник	$\omega_{12}(x,y) = ir(\omega_1, \omega_2)$
Ω_3 (ліве коло)	$b^2 - (x+a)^2 - y^2 \ge 0$	$\omega_3(x,y) = b^2 - (x+a)^2 - y^2$
Ω_4 (праве коло)	$b^2 - (x-a)^2 - y^2 \ge 0$	$\omega_4(x,y) = b^2 - (x-a)^2 - y^2$
$\Omega_{123} = \Omega_{12} \cup \Omega_3$	Прямокутник з лівим колом	$\omega_{123}(x,y) = ur(\omega_{12}, \omega_3)$
$\Omega_{1234} = \Omega_{123} \cup \Omega_4$	Стадіон без отвору	$\omega_{1234}(x,y) = ur(\omega_{123}, \omega_4)$
Ω_5 (коло-отвір)	$r^2 - x^2 - y^2 \ge 0$	$\omega_5(x,y) = r^2 - x^2 - y^2$
$\Omega = \Omega_{1234} \setminus \Omega_5$	Результівна фігура	$\omega(x,y) = dr(\omega_{1234}, \omega_5)$

Реалізуємо ці дії над прямокутною сіткою точок, що покриває досліджувану фігуру.

```
import numpy as np
import matplotlib.pyplot as plt
```

Створіть виконуючі до булевих операцій функції.

```
def ir(u,v): return (u+v-np.abs(u-v))/2
def ur(u,v): return (u+v+np.abs(u-v))/2
def dr(u,v): return (u-v-np.abs(u+v))/2
a,b,r=2,1.5,1
```

Створіть сітку точок, що гарантовано покриває досліджувану фігуру.

```
x_=np.linspace(-4,4,81)
y_=np.linspace(-2,2,81)
x,y=np.meshgrid(x_,y_)
```

Послідовно обчисліть вирази по формулам, наведеним в таблиці.

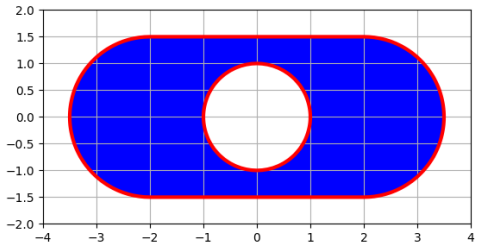
```
w1=a**2-x**2
w2=b**2-y**2
w12=ir(w1,w2)
w3=b**2-(x+a)**2-y**2
w4=b**2-(x-a)**2-y**2
w123=ur(w12,w3)
w1234=ur(w123,w4)
w5=r**2-x**2-y**2
w=dr(w1234,w5)
```

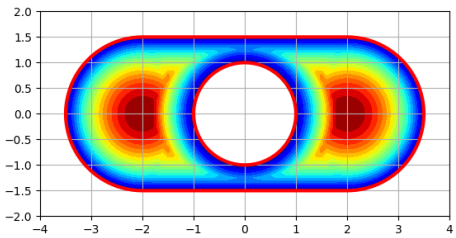
Тут величини `w` задають значення функції ідентифікації області Ω в точках сітки `(x,y)`. Побудуємо графік її нульової лінії рівня, тобто контур межі області.

```
fig,ax = plt.subplots(1,1)
ax.contour(x,y, w, [0], linewidths=3,colors='r')
```

Щоб зафарбувати область, можна використати функцію `contourf()`. Але вона фарбує зони між лініями рівня, тому їй потрібно передати принаймні два значення, одне з яких буде нулем, а інше – максимумом `w`. Оскільки `w` негативне зовні, то можна просто обрати величину `w.max()`.

```
ax.contourf(x,y, w, [0,w.max()], colors='b')
ax.grid(True)
ax.set_aspect(1)     # наступний рисунок ліворуч
```





Зауваження. Діапазон `[0,w.max()]` можна розбити на кілька рівнів і побудувати зафарбований контурний графік індикаторної функції. Замініть інструкцію `ax.contourf(x,y,w,...)` на наступні

```
vals=np.linspace(0,w.max(),20)
ax.contourf(x,y,w,vals,cmap='jet')  # попередній рисунок праворуч
```

і ви отримаєте зображення, яке показано на попередньому рисунку праворуч.

Але функція ідентифікації, зазвичай, грає допоміжну роль, а в області потрібно візуалізувати якесь скалярне поле. Для цього можна використати масковані масиви. Наприклад, нехай скалярне поле має вигляд $f(x,y) = x^2 + y^2$. Розфарбуємо досліджувану область (стадіон з отвором) відповідно до значень $f(x,y)$.

В продовження попереднього сценарію введіть наступні інструкції.

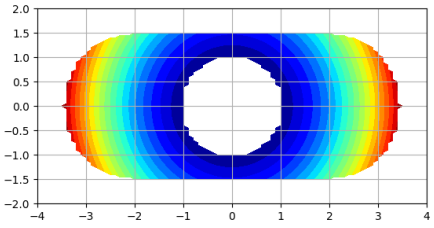
```
f=lambda x,y: x**2+y**2
```

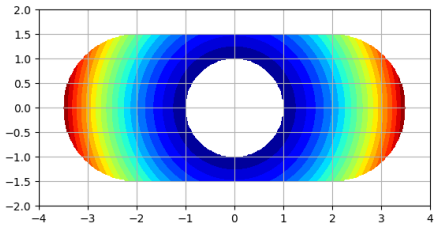
Створіть масковані масиви координат точок сітки і значень скалярного поля.

```
X=np.ma.masked_where(w<0,x)
Y=np.ma.masked_where(w<0,y)
F=f(X,Y)          # маскований масив значень скалярного поля в області
```

Побудуйте залитий контурний графік поля в області.

```
fig2,ax2 = plt.subplots(1,1)
vals=np.linspace(F.min(),F.max(),20)
ax2.contourf(X,Y,F,vals,cmap='jet')          # наступний рисунок ліворуч
ax2.grid(True)
ax2.set_aspect(1)
```





Щоб зображення по контуру виглядало менш «ребристим», потрібно подрібнити сітку, розбивши діапазони зміни абсцис `[-4,4]` і ординат `[-2,2]`, наприклад, на 401 і 201 значення (попередній рисунок праворуч).

Приклад 4. Побудувати зображення поверхні тривимірного тіла, що задано нерівностями $x^2+y^2\le 1$ і $y^2+z^2\le 1$ (циліндри одиничного радіуса з осями вздовж координатних прямих OZ і OX).

Розв'язання. Нерівності виділяють зони тривимірного простору. Для кожної з них створимо функцію ідентифікації (додатну всередині, нуль – на межі, від'ємну зовні).

Область	Функція ідентифікації ($\omega \ge 0$)
$x^2+y^2\le 1$	$\omega_1(x,y,z)=1-x^2-y^2$
$y^2+z^2\le 1$	$\omega_2(x,y,z)=1-y^2-z^2$

Перетин цих областей утворюватиме тіло з функцією ідентифікації

$$\omega(x,y,z)=ir(\omega_1(x,y,z),\omega_2(x,y,z)).$$

Отже неявним рівнянням поверхні тіла буде $\omega(x,y,z)=0$. Реалізуємо ці побудови в Python, застосовуючи символьні обчислення.

```
from sympy import symbols, simplify, lambdify, Abs
x,y,z= symbols('x y z')
```

Створіть виконуючу до операції перетину функцію $ir(x,y)$.

```
def ir(x,y): return (x+y-Abs(x-y))/2
```

Створіть функції ідентифікації областей $x^2+y^2\le 1$, $y^2+z^2\le 1$ та області їх перетину.

```
w1=1-x**2-y**2
w2=1-y**2-z**2
w=simplify(ir(w1,w2))
print('w=',w)
w=-x**2/2-y**2-z**2/2-Abs(x**2-z**2)/2+1
```

Тобто неявним рівнянням поверхні результівного тіла є

$$1-\frac{x^2}{2}-y^2-\frac{z^2}{2}-\frac{1}{2}|x^2-z^2|=0\,.$$

Для візуалізації поверхні по неявному рівнянню застосуємо функцію `mlab.contour3d()`. Але спочатку з символьного виразу `w` створіть `ufunc` функцію `F`, щоб її можна було застосовувати до масивів координат точок тривимірної сітки.

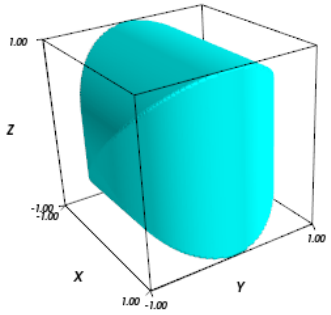
```
F=lambdify((x,y,z), w, 'numpy')
```

Створіть просторову сітку точок з координатами `(X,Y,Z)`, яка покриває досліджувану область, і масив значень виразу $\omega(x,y,z)$ в цих точках.

```
import numpy as np
from mayavi import mlab
mlab.close(all=True)
X,Y,Z = np.mgrid[-1.1:1.1:100j,-1.1:1.1:100j,-1.1:1.1:100j]
W=F(X,Y,Z)
```

Тепер побудуйте поверхню нульового рівня виразу $\omega(x,y,z)$.

```
fig=mlab.figure(fgcolor=(0, 0, 0), bgcolor=(1, 1, 1))
isurf=mlab.contour3d(X,Y,Z,W, contours=[0],color=(0,1,1))
mlab.outline()
mlab.axes()
```



Приклад 5. Побудувати зображення куба $|x|\le 1\,,|y|\le 1\,,|z|\le 1$ з циліндричним отвором $x^2+y^2\le 0.7^2$.

Для зон простору, що відповідають нерівностям, створимо функції ідентифікації.

Область	Функція ідентифікації ($\omega \ge 0$)
$\|x\|\le 1$	$\omega_1(x,y,z)=1-\|x\|$
$\|y\|\le 1$	$\omega_2(x,y,z)=1-\|y\|$
$\|z\|\le 1$	$\omega_3(x,y,z)=1-\|z\|$
$x^2+y^2\le 0.7^2$	$\omega_4(x,y,z)=0.49-\mathrm{x}^2-y^2-z^2$

Перетин перших трьох зон/шарів утворює область в формі куба з функцією ідентифікації

$$\omega_{cube}(x,y,z)=ir(ir(\omega_1,\omega_2),\omega_3)$$

Область куба з циліндричним отвором матиме наступну функцію ідентифікації

$$\omega(x,y,z)=dr(\omega_{cube},\omega_4)$$

де $dr(\omega_1,\omega_2)$ є виконуючою функцією (1) до логічної операції різниці областей.

Реалізуємо ці побудови в Python, застосовуючи символьні обчислення і бібліотеку наукової графіки `mayavi`.

```
from sympy import symbols, simplify, lambdify, Abs
x, y, z= symbols('x y z')
```

Створіть виконуючі функції $ir(\omega_1,\omega_2), dr(\omega_1,\omega_2)$ до операцій перетину і різниці.

```
def ir(u,v): return (u+v-Abs(u-v))/2
def dr(u,v): return (u-v-Abs(u+v))/2
```

Створіть вирази для функцій індикації шарів $|x| \le 1$, $|y| \le 1$, $|z| \le 1$ і циліндра $x^2 + y^2 \le 0.7^2$

```
w1=1-Abs(x)
w2=1-Abs(y)
w3=1-Abs(z)
w4=0.49-x**2-y**2
```

Згенеруйте вирази для функцій індикації куба `wcube` і куба з отвором `w`.

```
w12=simplify(ir(w1,w2))
wcube=simplify(ir(w12,w3))
print(f'Неявне рівняння поверхні куба:\n{wcube}=0')
1-Abs(x)/4-Abs(y)/4-Abs(z)/2-Abs(Abs(x)-Abs(y))/4-
        Abs(Abs(x)+Abs(y)-2*Abs(z)+Abs(Abs(x)-Abs(y)))/4=0
w=dr(wcube,w4)
```

З символьного виразу `w` створіть функцію `F`, щоб її можна було застосовувати до масивів координат точок тривимірної сітки.

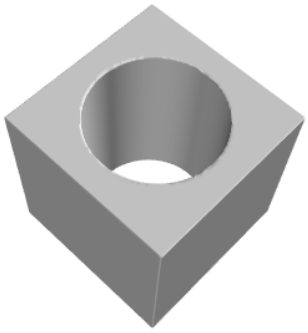
```
F=lambdify((x,y,z),w,'numpy')
```

Створіть просторову сітку точок і масив значень виразу $\omega(x,y,z)$ в цих точках.

```
import numpy as np
from mayavi import mlab
mlab.close(all=True)
X,Y,Z = np.mgrid[-1.1:1.1:100j,-1.1:1.1:100j,-1.1:1.1:100j]
W=F(X,Y,Z)
```

Тепер побудуйте поверхню нульового рівня виразу/масиву $\omega(x,y,z)$.

```
fig=mlab.figure(fgcolor=(0, 0, 0), bgcolor=(1, 1, 1))
isurf=mlab.contour3d(X,Y,Z,W, contours=[0],color=(0.8,0.8,0.8))
```



Приклад 6. Побудувати зображення області, заданої нерівностями $x^2 + y^2 \le 1$, $|z| \le 2$ та $x^2 + z^2 \le 1$, $|y| \le 2$.

Нерівності виділяють зони тривимірного простору. Для кожної з них створіть функцію ідентифікації.

Область	Функція ідентифікації ($\omega \ge 0$)
$x^2 + y^2 \le 1$	$\omega_1(x,y,z) = 1 - x^2 - y^2$
$\|z\| \le 2$	$\omega_2(x,y,z) = 2 - \|z\|$
$x^2 + z^2 \le 1$	$\omega_3(x,y,z) = 1 - x^2 - z^2$
$\|y\| \le 2$	$\omega_4(x,y,z) = 2 - \|y\|$

Перші дві нерівності $x^2+y^2\le 1,\ |z|\le 2$ виділяють зону циліндра (фіксованої довжини 4). Його функція ідентифікації буде $\omega_{12}(x,y,z)=ir(\omega_1(x,y,z),\omega_2(x,y,z))$. Аналогічно нерівності $x^2+z^2\le 1,\ |y|\le 2$ задають зону іншого циліндра, функцією ідентифікації якого є $\omega_{34}(x,y,z)=ir(\omega_3(x,y,z),\omega_4(x,y,z))$. Поєднання обох циліндрів утворює тіло з функцією ідентифікації

$$\omega(x,y,z)=ur\big(\omega_{12}(x,y,z),\omega_{34}(x,y,z)\big),$$

де $ur(\omega_1,\omega_2)$ є виконуючою функцією (1) булевої операції об'єднання областей.

Реалізуємо ці побудови в Python, при чому обійдемося без символьних розрахунків, і відразу будуватимемо числові масиви.

```
import numpy as np
from mayavi import mlab
mlab.close(all=True)
```

Створіть виконуючі функції операцій перетину та об'єднання

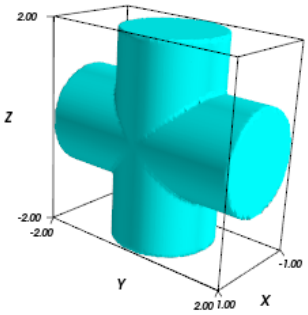
```
def ir(u,v): return (u+v-np.abs(u-v))/2
def ur(u,v): return (u+v+np.abs(u-v))/2
x,y,z=np.mgrid[-1.1:1.1:50j,-2.1:2.1:100j,-2.1:2.1:100j]# сітка точок
```

Побудуйте масиви значень функцій ідентифікації на обраній сітці точок

```
w1=1-x**2-y**2
w2=2-np.abs(z)
w12=ir(w1,w2)
w3=1-x**2-z**2
w4=2-np.abs(y)
w34=ir(w3,w4)
w=ur(w12,w34)     # масив значень результівної функції ідентифікації
```

Побудуйте поверхню $\omega(x,y,z)=0$

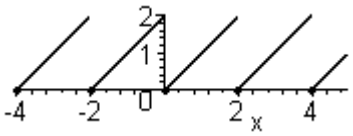
```
fig=mlab.figure(fgcolor=(0, 0, 0), bgcolor=(1, 1, 1))
isurf=mlab.contour3d(x, y, z, w, contours=[0],color=(0,1,1))
mlab.outline()
mlab.axes()
```



Зауваження. Незручністю розглянутого підходу є неможливість фарбування поверхонь відповідно до значень скалярного поля відразу, наприклад, для візуалізації температури на поверхні тіла. Щоб таке зробити доводиться застосовувати спеціальні трюки. Але і без цього функції ідентифікації областей дуже зручні при побудові базисних функцій в методі Гальоркіна, бо їх вибір є чи не «найнеприємнішим» моментом для цього дуже зручного метода наближеного розв'язання диференціальних рівнянь в частинних похідних.

10.5. Періодичне продовження функцій.

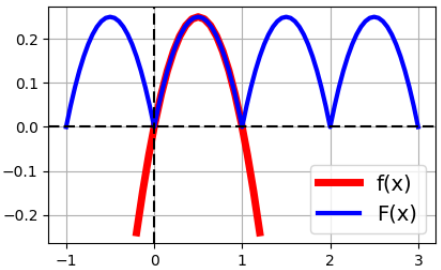
В деяких прикладних задачах потрібно будувати періодичне продовження функції $\varphi(x)$ зі скінченного відрізка на всю дійсну вісь. Основою таких побудов буде елементарна функція $\mathrm{mod}(x,T)$ – взяття залишку від ділення числа x на число T. В пакеті `numpy` є відповідна вбудована функція `numpy.mod(x,T)`. Ось приклад графіка $\mathrm{mod}(x,2)$



Функція $\mathrm{mod}(x,T)$ є періодичною з періодом $T>0$, на відрізку $0\le x<T$ вона співпадає з лінійною функцією $y=x$, і в точках $x=k\cdot T$ $(k=0,\pm1,\pm2,\dots)$ має розриви.

Використовуючи $\mathrm{mod}(x,T)$, ми можемо побудувати періодичне продовження будь-якої функції. Візьмемо, наприклад, функцію у формі параболи $f(x)=x\cdot(1-x)$. Її графік показано на наступному рисунку червоним кольором. Якщо замість аргументу x підставити функцію $\mathrm{mod}(x,1)$, тобто побудувати функцію $F(x)=f\big(\mathrm{mod}(x,1)\big)=\mathrm{mod}(x,1)\cdot\big(1-\mathrm{mod}(x,1)\big)$, то для аргументів з інтервалу $0\le x<1$ нічого не зміниться ($\mathrm{mod}(x,1)=x$ при $0\le x<1$) і функція $F(x)$ буде збігатися з входовою функцією $f(x)$. Для аргументів поза цим інтервалом функція $\mathrm{mod}(x,1)$ буде періодично повторювати свої значення, отже і $F(x)$ буде періодичною. Її графік показано на наступному рисунку синім кольором.

```
import numpy as np
import matplotlib.pyplot as plt
def f(x):  return x*(1-x)
def F(x): return f(np.mod(x,1))
fig,ax = plt.subplots(1,1)
x=np.linspace(-0.2,1.2,21); y=f(x)
ax.plot(x,y,'r',lw=5,label='f(x)')  # графік входової параболи
X=np.linspace(-1,3,81); Y=F(X)
ax.plot(X,Y,'b',lw=3,label='F(x)')  # графік періодичного продовження
ax.axhline(0,color='black',dashes=(5,2))
ax.axvline(0,color='black',dashes=(5,2))
ax.legend(loc='lower right',fontsize=14)
ax.grid(True)
```

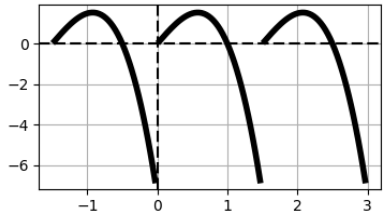


Тобто, якщо якусь функцію $f(x)$ потрібно періодично продовжити з ділянки $[0,T)$ на всю дійсну вісь з періодом T, то можна застосувати формулу $F(x)=f(\mathrm{mod}(x,T))$. Якщо потрібно періодично повторити функцію $f(x)$ з напівінтервала $[a,b)$ на всю вісь, то слід використати формулу [6]

$$F(x) = f\big(a + \mathrm{mod}(x - a, b - a)\big)\,. \qquad (1)$$

Зазначимо, що коли функція $f(x)$ на кінцях відрізка [a,b] набуває однакових значень $f(a) = f(b)$, то результівне періодичне продовження $F(x)$ буде неперервною функцією. Якщо це не так, то отримувана періодична функція буде розривною, наприклад, такою, як зображено на наступному рисунку.

Продовження функції $f(x) = 4\,x\,(1 - x^2)$ з відрізка [0, 1.5] з періодом 1.5.



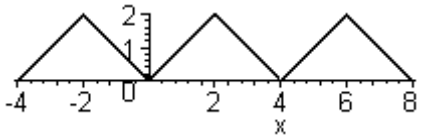
Нам також часто потрібно буде будувати парне та непарне періодичне продовження функцій. Парне періодичне продовження полягає в продовженні функції $f(x)$ з відрізка $[0,\ L]$ на відрізок $[-L,\ 0]$ парно, а потім з відрізка $[-L,\ L]$ періодично на всю дійсну вісь. З будь-якої функції $f(x)$ заданої для $x \geq 0$ можна отримати парну функцію $\tilde{f}(x)$ за формулою $\tilde{f}(x) = f(|\,x\,|)$. Для її періодичного продовження з відрізка $[-L,\ L]$, відповідно до (1), аргумент x слід замінити на $-L + \mathrm{mod}(x + L, 2L)$, тобто побудувати функцію

$$F(x) = f(|-L + \mathrm{mod}(x + L, 2L)|) = f(|-L + \mathrm{mod}(x - L, 2L)| \qquad (2)$$

(тут враховано, що $\mathrm{mod}(x - L, 2L) = \mathrm{mod}(x + L, 2L)$). Для подальшого спрощення визначимо функцію $\mathrm{stc}(x, w)$

$$\mathrm{stc}(x, w) = \left|\ \mathrm{mod}\left(x - \frac{w}{2}, w\right) - \frac{w}{2}\ \right| \qquad (3)$$

Вона є неперервною парною періодичною функцією з періодом w, яка на відрізку $-\frac{w}{2} \leq x \leq \frac{w}{2}$ співпадає з функцією $y = |\,x\,|$. Графік $\mathrm{stc}(x, 4)$ представлений на наступному рисунку.



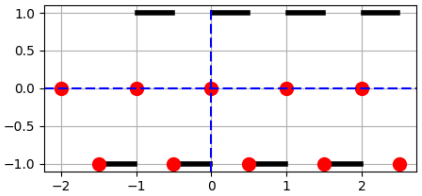
Враховуючи (3), де покладемо $w = 2L$, формулу (2) парного періодичного продовження функції $f(x)$ з відрізка $[0,\ L]$ на всю дійсну вісь перепишемо у вигляді

$$F(x) = f(\mathrm{stc}(x, 2L)) \qquad (4)$$

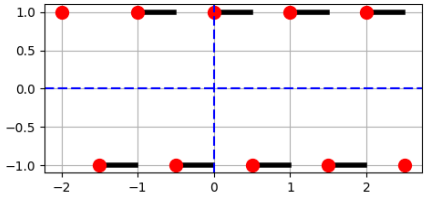
З будь-якої функції $f(x)$, заданої для $x \geq 0$, непарну функцію $\tilde{f}(x)$ можна побудувати за формулою $\tilde{f}(x) = \mathrm{sign}(x) \cdot f(|\,x\,|)$, де $\mathrm{sign}(x)$ – функція знака. Непарне періодичне продовження $f(x)$ з відрізка $[0,\ L]$ полягає в непарному продовженні $f(x)$ на напівінтервал $[-L,\ 0)$, а потім з відрізка $[-L,\ L]$ періодично на всю дійсну вісь. Відповідно до (1) і, враховуючи періодичність $\mathrm{mod}(x - L, 2L) = \mathrm{mod}(x + L, 2L)$), таке продовження можна виконати формулою

$$\begin{aligned} F(x) = \mathrm{sign}\big(-L + \mathrm{mod}(x - L, 2L)\big) \cdot f(|-L + \mathrm{mod}(x - L, 2L)|) = \\ = \mathrm{psign}(x, 2L) \cdot f\big(\mathrm{stc}(x, 2L)\big)\,, \end{aligned} \qquad (5)$$

де для зручності позначено $\mathrm{psign}(x,w)=\mathrm{sign}\left(-\frac{w}{2}+\mathrm{mod}\left(x-\frac{w}{2},w\right)\right)$. Це є періодичною імпульсною функцією з періодом w. Приклад її графіка при $w=1$ з окремо показаними значеннями в точках розриву наведено на наступному рисунку ліворуч.



$\mathrm{sign}\left(-\frac{w}{2}+\mathrm{mod}\left(x-\frac{w}{2},w\right)\right)$ при $w=1$



$(-1)^{\left[\frac{x}{w/2}\right]}$ при $w=1$

Для функції $\mathrm{psign}(x,w)$ можна запропонувати кілька інших формул, наприклад,

$$\mathrm{psign}(x,w)=(-1)^{\left[\frac{x}{w/2}\right]},$$
$$\mathrm{psign}(x,w)=2\left(\left[\frac{x}{w}\right]-\left[\frac{x}{w}-\frac{1}{2}\right]\right)-1\,. \qquad (6)$$

де квадратні дужки [z] позначають функцію обчислення найближчого цілого, яке не перевищує z. У неї є реалізація `numpy.floor(z)`.

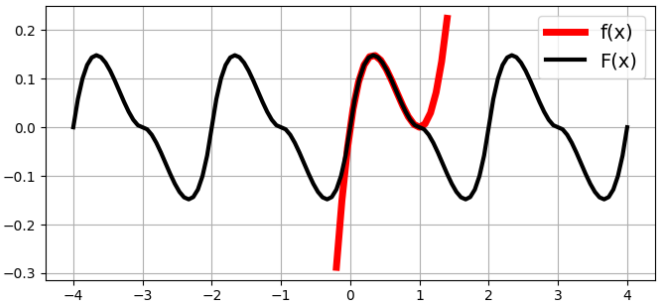
Насправді, формули (6) відрізняються від виразу $\mathrm{sign}\left(-\frac{w}{2}+\mathrm{mod}\left(x-\frac{w}{2},w\right)\right)$ значеннями в точках $x=k\cdot w$, де k – ціле (див. попередній рисунок праворуч). Але для нашої мети це несуттєво. В подальшому нам буде зручно застосовувати вираз $(-1)^{\left[\frac{x}{w/2}\right]}$, і формулу (5) непарного періодичного продовження функції $f(x)$ з відрізка $[0,\ L]$ ми записуватимемо у вигляді [6]

$$F(x)=(-1)^{\left[\frac{x}{L}\right]}\cdot f(\mathrm{stc}(x,2L)) \qquad (7)$$

Зауважимо, що якщо входова функція $f(x)$ на кінцях відрізка $[0,L]$ дорівнюватиме нулю, то отримувана за формулою (7) (або (5)) функція $F(x)$ буде неперервною (і навіть диференційованою в цих точках).

Для прикладу візьмемо кубічну параболу $f(x)=x\cdot(1-x)^2$ і продовжимо її непарно і періодично з відрізка $[0,\ 1]$ на всю дійсну вісь.

```
import numpy as np
import matplotlib.pyplot as plt
def f(x):  return x*(1-x)**2
def F(x): return f(stc(x,2))*(-1)**np.floor(x)
def stc(x,w): return np.abs(np.mod(x-w/2,w)-w/2)
fig,ax = plt.subplots(1,1)
x=np.linspace(-0.2,1.4,21); y=f(x)
ax.plot(x,y,'r',lw=5,label='f(x)')     # графік входової функції
X=np.linspace(-4,4,121); Y=F(X)
ax.plot(X,Y,'k',lw=3,label='F(x)')     # непарне періодичне продовження
ax.legend(loc='upper right',fontsize=14)
ax.grid(True)
```



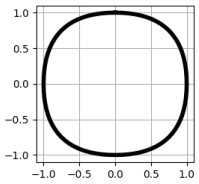
Для формул (1) – (7) нам знадобилися функції обчислення залишку $\mathrm{mod}(x,T)$, найближчого меншого цілого [z] та модуля $|\,x\,|$, які вбудовані практично у всі системи програмування, швидко і точно обчислюються. Тому для моделювання періодичних продовжень функцій ми застосовуватимемо методику, яка використовує ці функції.

Зауваження. Для пилкоподібної функції $\mathrm{stc}(x,w)=\left|\,\mathrm{mod}\left(x-\frac{w}{2},w\right)-\frac{w}{2}\,\right|$ можна запропонувати кілька інших формул обчислення, наприклад,

$$\begin{aligned}
\mathrm{stc}(x,w) &= \frac{w}{2\pi}\arccos\left(\cos\frac{2\pi x}{w}\right),\\
\mathrm{stc}(x,w) &= \frac{w}{\pi}\arcsin\left(\left|\,\sin\frac{\pi x}{w}\,\right|\right),\\
\mathrm{stc}(x,w) &= \frac{w}{2}-\left|\frac{w}{2}-\mathrm{mod}(x,w)\right|,\\
\mathrm{stx}(x,w) &= \left|x-w\left[\frac{x}{w}+\frac{1}{2}\right]\right|.
\end{aligned} \tag{8}$$

Зауваження. Будь-яка пара періодичних функцій $x(t),y(t)$ з однаковим періодом утворює параметричні рівняння замкненої кривої. Наприклад, наступна замкнена крива утворена через непарне періодичне продовження параболи:

$f(x)=4\,x\,(1-x)$
$F(x)=(-1)^{[x]}f\big(\mathrm{stc}(x,2)\big)$
$x=F(t)$
$y=F(t+0.5)$



Для функцій кількох змінних також можна запропонувати спосіб їх періодичного продовження з прямокутної області на весь простір. Розглянемо лише двовимірний випадок.

Нехай є функція $f(x,y)$, яка задана на прямокутнику $D_{xy}=[0,\,L_x]\times[0,\,L_y]$. Тоді, щоб її періодично продовжити у всіх напрямках (вздовж осі X та осі Y), достатньо аргументи x,y замінити на $\mathrm{mod}(x,L_x)$ та $\mathrm{mod}(y,L_y)$, тобто побудувати функцію

$$F(x,y)=f\big(\mathrm{mod}(x,L_x),\mathrm{mod}(y,L_y)\big) \tag{9}$$

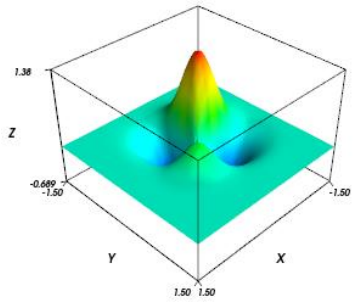
Приклад. Періодично продовжити функцію

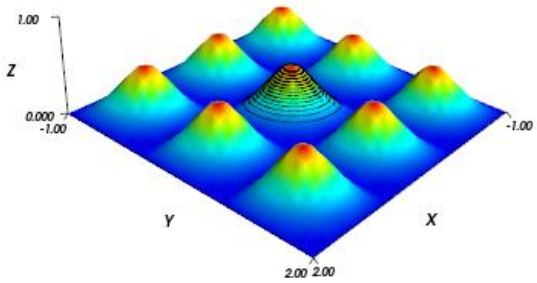
$$f(x,y)=16\,x\,(1-x)\,y\,(1-y)\,e^{-4x^2-4y^2}\,.$$

з квадрата [0,1]×[0,1] на всю площину XY.

Р о з в ' я з а н н я . Функція $f(x,y)$ має наступний графік

```
import numpy as np
from mayavi import mlab
mlab.close(all=True)
def f(x,y):
    return 16*x*(1-x)*y*(1-y)*np.exp(-4*x**2-4*y**2)
x,y = np.mgrid[-1.5:1.5:41j,-1.5:1.5:41j]
z=f(x,y)
mlab.figure(fgcolor=(0,0,0), bgcolor=(1,1,1))
sh=mlab.mesh(x,y,z)          # графік функції f(x,y) ліворуч
mlab.outline(sh)
mlab.axes()
```





Щоб виділити її частину над квадратом `[0,1]x[0,1]` і періодично продовжити на всю площину XY, можна застосувати формулу (9).

```
def F(x,y):  return f(np.mod(x,1),np.mod(y,1))
x1,y1 = np.mgrid[-1:2:31j,-1:2:31j]
z1=F(x1,y1)
mlab.figure(fgcolor=(0,0,0),bgcolor=(1,1,1))
sf=mlab.mesh(x1,y1,z1)          # попередній рисунок праворуч
x,y = np.mgrid[0:1:21j,0:1:21j]
z=f(x,y)
vals=np.linspace(z.min(),z.max(),16)
mlab.contour_surf(x,y,z,contours=list(vals),color=(0,0,0))
mlab.axes(sf,extent=[-1,2,-1,2,0,1])
```

На останньому рисунку розмножувана частинка функції додатково виділена чорним каркасом.

В даному прикладі входова функція на контурі квадрата дорівнює нулю. Тому періодична поверхня виявилася неперервною, але в загальному випадку це не обов'язково повинно бути.

■

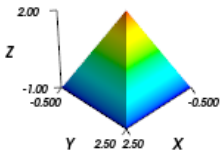
Якщо функцію, задану над прямокутною коміркою $D_{xy} = [0,\ L_x] \times [0,\ L_y]$, потрібно в обох напрямках продовжити парно і періодично, то у вираз $f(x,y)$ замість змінних x, y слід підставляти $\mathrm{stc}(x, 2\,Lx)$ і $\mathrm{stc}(y, 2\,Ly)$, тобто створити функцію

$$F(x,y) = f(\mathrm{stc}(x, 2\,Lx), \mathrm{stc}(y, 2\,Ly)) \qquad (10)$$

Аналогічно можна виконати непарне періодичне продовження $f(x,y)$ з прямокутника D_{xy} на всю площину XY, побудувавши функцію

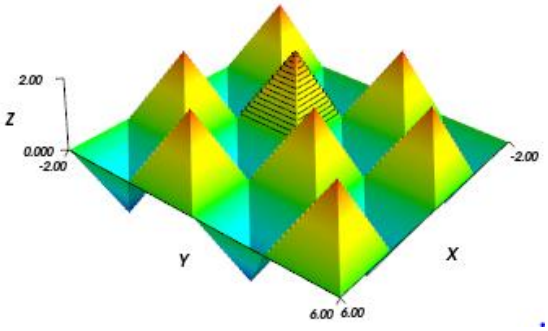
$$F(x,y) = (-1)^{\left[\frac{x}{L_x}\right]} \cdot (-1)^{\left[\frac{y}{L_y}\right]} \cdot f(\mathrm{stc}(x, 2\,L_x), \mathrm{stc}(y, 2\,L_y)) \qquad (11)$$

Приклад. Побудувати непарне періодичне продовження функції-піраміди $f(x,y) = 2 - |x-y| - |x+y-2|$ з квадрата `[0,2]x[0,2]` на всю площину XY.



Розв'язання.

```
import numpy as np
from mayavi import mlab
mlab.close(all=True)
def f(x,y): return (2-np.abs(x-y)-np.abs(x+y-2))
def stc(x,w): return np.abs(np.mod(x-w/2,w)-w/2)
def F(x,y):          # функція непарного періодичного продовження
    return f(stc(x,4),stc(y,4))*(-1)**np.floor(x/2)*(-1)**np.floor(y/2)
x1,y1 = np.mgrid[-2:6:121j,-2:6:121j]
z1=F(x1,y1)
mlab.figure(fgcolor=(0,0,0),bgcolor=(1,1,1))
sf=mlab.mesh(x1,y1,z1)
x,y = np.mgrid[0:2:41j,0:2:41j]; z=f(x,y)
vals=np.linspace(z.min(),z.max(),16)
mlab.contour_surf(x,y,z,contours=list(vals),color=(0,0,0))
mlab.axes(sf,extent=[-2,6,-2,6,0,2])
```



На останньому рисунку розмножувана ділянка функції-піраміди додатково виділена чорним каркасом.

10.6. Фізичні застосування.

Інженери інколи стикаються з математичними моделями процесів, фізичні характеристики яких, а отже і параметри описуючих їх рівнянь, різняться на різних ділянках досліджуваної області. При чому головні обчислювані величини майже завжди вважаються неперервними при переході з однієї зони до іншої. Одночасно з цим зовнішні впливи часто моделюються як точково-зосереджені або кусково-неперервні, так само як і функції, що їх представляють.

Одиничні зовнішні впливи, що зосереджені на нескінченно вузькій ділянці дійсної осі моделюються дельта-функцією Дірака $\delta(t,a)$, де a – точка прикладання зовнішньої дії. Вона визначається як нуль всюди, крім точки $t = a$, де вона має сингулярність таку, що $\int_{-\infty}^{\infty} \delta(t,a)dt = 1$. Невизначений інтеграл

$\int_\alpha^x \delta(\xi,a)d\xi\,,\ \alpha<a$ від дельта-функції Дірака покладається рівним $\int_\alpha^x \delta(\xi,a)d\xi = H(x,a)$, де $H(x,a)=\begin{cases}0\,,\ x<a\\1\,,\ x\geq a\end{cases}$ – функція Гевісайда (функція одиничної сходинки, при чому значення в точці $x=a$ часто несуттєво). Також очевидно, що $\int_\alpha^x H(t,a)dt = Q_r(x,a)$ $(\alpha<a)$, тобто первісна функції Гевісайда співпадає з неперервною кусково-лінійною функцією $Q_r(x,a)$, визначеною формулою (3) п. 10.3. Взагалі то, під невизначеним інтегралом $Z(x)$ кусково-неперервної функції $z(x)$ можна розуміти визначений інтеграл, що розглядається як функція верхньої межі $Z(x)=\int_\alpha^x z(\xi)d\xi$. Він є неперервною кусковою функцією, і визначатиметься з точністю до довільної сталої через присутність параметра α.

В цьому розділі нам знадобиться інтегрувати кускові функції. Тому побіжно розглянемо це питання і наведемо таблицю первісних деяких «нетрадиційних» функцій, в якій вважатиметься, що нижня межа інтегрування α завжди менша за параметр a, що зустрічається в формулах (тобто $\alpha<a$).

$$\int_\alpha^x \delta(\xi,a)d\xi = H(x,a)\,. \tag{1}$$

$$\int_\alpha^x H(\xi,a)d\xi = Q_r(x,a)\,. \tag{2}$$

$$\int_\alpha^x \big(Q_r(\xi,a)\big)^m d\xi = \frac{1}{m+1}(x-a)^m\cdot Q_r(x,a) = \frac{1}{m+1}\big(Q_r(x,a)\big)^{m+1}, m=1,2,\ldots \tag{3}$$

$$\int_\alpha^x (x-a)^{m-1}|x-a|\,d\xi = \mathrm{C} + \frac{1}{m+1}(x-a)^m|\,x-a\,|\,, m=1,2,\ldots\,, \tag{4}$$

де $C=\dfrac{(\alpha-a)^{m+1}}{m+1}$.

Зауважте, що в правій частині формул (3) і (4) стоять функції класу $C^m(-\infty,\,\infty)$, тобто такі, що мають неперервні похідні до m-го порядку на всій дійсній осі.

Наведемо ще формулу обчислення «первісної» $\int_{x_0}^x k(\xi)\,d\,\xi$ від шматково-сталої функції $k(x)$:

$$k(x)=\begin{cases}k_1,\ x\leq x_1\\ k_2,\ x_1<x\leq x_2\\ \cdots\\ k_n,\ x_{n-1}<x\end{cases},\ x_0<x_1<\cdots<x_{n-1}.$$

Використовуючи функцію Гевісайда, $k(x)$ можна представити у вигляді

$$k(x)=k_1+\sum_{i=1}^{n-1}(k_{i+1}-k_i)\cdot H(x,x_i)\ . \tag{5}$$

Інтегруючи ліву і праву частини від x_0 до x, і враховуючи (2), матимемо

$$I(x) = \int_{x_0}^{x} k(\xi)\, d\,\xi = k_1(x - x_0) + \sum_{i=1}^{n-1} (k_{i+1} - k_i) \cdot Q_r(x, x_i)\ . \qquad (6)$$

Згадуючи, що $Q_r(x, x_i) = \frac{1}{2}\ (x - x_i + |\ x - x_i\ |)$, отримуємо

$$\int_{x_0}^{x} k(\xi) d\,\xi = k_1(x - x_0) + \frac{1}{2}\sum_{i=1}^{n-1} (k_{i+1} - k_i) \cdot (x - x_i) + \frac{1}{2}\sum_{i=1}^{n-1} (k_{i+1} - k_i) \cdot |\ x - x_i\ | =$$

$$= \frac{1}{2}\left(k_1(x - x_0) + \sum_{i=1}^{n-1} k_i(x_i - x_{i-1}) + k_n(x - x_{n-1}) + \sum_{i=1}^{n-1} (k_{i+1} - k_i) \cdot |\ x - x_i\ | \right) \quad (7)$$

Зручніше цю формулу записати у вигляді

$$I(x) = \int_{x_0}^{x} k(\xi)\, d\,\xi = a_0 + a_1 \cdot x + \frac{1}{2}\sum_{i=1}^{n-1} (k_{i+1} - k_i) \cdot |\ x - x_i\ |\,, \qquad (8)$$

де $a_0 = \frac{1}{2}\left(\sum_{i=1}^{n-1} k_i(x_i - x_{i-1}) - k_1\, x_0 - k_n\, x_{n-1} \right), a_1 = \frac{1}{2}(k_1 + k_n)\,.$

Якщо в формулу (6) замість довільного x підставити значення x_n, яке більше за будь-яке x_i $(i = 0,1, \dots, n-1)$ то, з огляду на те, що $Q_r(x_n, x_i) = x_n - x_i$, вона набуде вигляду

$$I(x_n) = \int_{x_0}^{x_n} k(\xi) d\xi = k_1(x_n - x_0) + \sum_{i=1}^{n-1} (k_{i+1} - k_i) \cdot (x_n - x_i) = \sum_{i=1}^{n} k_i(x_i - x_{i-1}) \qquad (9)$$

З останнього випливатиме $\sum_{i=1}^{n-1} k_i(x_i - x_{i-1}) = I(x_n) - k_n(x_n - x_{n-1})$. Підставивши це в (7), отримуємо

$$\int_{x_0}^{x} k(\xi) d\,\xi = \frac{1}{2}\left(k_1(x - x_0) + I(x_n) + k_n(x - x_n) + \sum_{i=1}^{n-1} (k_{i+1} - k_i) \cdot |\ x - x_i\ | \right) \quad (10)$$

Нагадуємо, що в останній формулі $x_n > x_i\ \forall\ i$. В різних випадках інтегрування кусково-сталої функції $k(x)$ зручною може виявитися будь-яка з формул (6), (8) або (10).

Прогини струни. Диференціальне рівняння рівноваги струни довжини L (в припущенні малих прогинів) має вигляд

$$u''_{xx} = -f(x)\,, \qquad (11)$$

де $f(x) = F(x)/T$, T – натяг струни, $F(x)$ – зовнішня (поперечна) сила обчислена на одиницю довжини. Ось X збігається з горизонтальним напрямком вздовж натягнутої ненавантаженої струни, а початок $x = 0$ розташовано на її лівому кінці. Функція $u(x)$ являє вертикальне зміщення точок струни, які мали до навантаження горизонтальну координату x. До рівняння рівноваги (11) додаються граничні умови закріплення кінців струни $u(0) = 0,\ u(L) = 0$. В подальшому вважатимемо також, що $T = 1$.

Приклад 1. Розглянемо струну, яка закріплена в кінцевих точках $x_0 = 0, x_n = L$, а в точках $\{x_i\}_{i=1}^{n-1}$, $x_0 < x_1 < x_2 < \ldots < x_{n-1} < x_n$ піддається дії поперечно прикладених точкових навантажень f_i. Одиничну силу, зосереджену в нескінченно малій зоні навкруги точки $x = a$, слід представляти як дельта-функцію Дірака $\delta(x, a)$. Функцію $f(x)$, що представляє набір прикладених зосереджених навантажень в точках $x_i \in (0, L)$, можна задати у вигляді $f(x) = \sum_{i=1}^{n-1} f_i \cdot \delta(x, x_i)$. Тоді диференціальне рівняння рівноваги струни набуде вигляду $u''_{xx} = -\sum_{i=1}^{n-1} f_i \cdot \delta(x, x_i)$. Обчислюючи інтеграл від 0 до x лівої і правої частин, отримуємо

$$u'_x(x) = C_1 - \sum_{i=1}^{n-1} f_i \int_0^x \delta(\xi, x_i) d\xi = C_1 - \sum_{i=1}^{n-1} f_i \cdot H(x, x_i) \ ,$$

де $C_1 = u'_x(0)$ поки що невизначена стала. Інтегруючи ще раз обидві частини останнього рівняння, отримуємо

$$u(x) = C_1 x + C_2 - \sum_{i=1}^{n-1} f_i \cdot \int_0^x H(\xi, x_i) d\xi \ ,$$

де $C_2 = u(0)$ – інша стала. Оскільки $\int_0^x H(\xi, x_i) d\xi = Q_r(x, x_i)$, то матимемо

$$u(x) = C_1 x + C_2 - \sum_{i=1}^{n-1} f_i \cdot Q_r(x, x_i)$$

Дві сталі C_1, C_2 можуть бути знайдені з граничних умов $u(0) = u(L) = 0$. Враховуючи, що $Q_r(0, x_i) = 0$, $Q_r(L, x_i) = L - x_i$ для будь-якого $0 < x_i < L$, знаходимо $C_2 = 0$, $C_1 = \frac{1}{L}\sum_{i=1}^{n-1} f_i(L - x_i)$. Тому

$$u(x) = \frac{x}{L}\sum_{i=1}^{n-1} f_i \ (L - x_i) - \sum_{i=1}^{n-1} f_i \cdot Q_r(x, x_i) = \sum_{i=1}^{n-1} f_i \left(\frac{x(L - x_i)}{L} - Q_r(x, x_i)\right) =$$

$$= \sum_{i=1}^{n-1} f_i \left(\frac{x(L - x_i) + x_i(L - x)}{2L} - \frac{1}{2}|x - x_i|\right).$$

Тут було використано визначення функції $Q_r(x, a)$, наведене в п. 10.3. Таким чином, остаточно отримуємо

$$u(x) = A \cdot x + B - \frac{1}{2}\sum_{i=1}^{n-1} f_i \cdot |x - x_i| \ , \text{де } A = \frac{1}{2L}\sum_{i=1}^{n-1} f_i(L - 2\,x_i) \ , B = \frac{1}{2}\sum_{i=1}^{n-1} f_i \cdot x_i \quad (12)$$

Для реалізації формули (12) створимо функцію `ctl_string(x,X,F,L)`, яка повертатиме символьний вираз $u(x)$ від символьної змінної `x`, що передається їй першим аргументом. В масивах `X` і `F` передаватимуться абсциси прикладання і величини зосереджених зусиль, `L` – довжина струни.

```
import numpy as np
from sympy import symbols, lambdify, Abs, nsimplify
import matplotlib.pyplot as plt
import sys
# ConcenTrated_Loaded_string
def ctl_string (x,X,F,L):
    """ Прогин струни під дією зосереджених в точках X[i]
```

```
        сил F[i] (0<X[i]<L)
        L - довжина струни, x - ім'я символьної змінної"""
    try:
        dx=np.diff(X)
        if np.any(dx<=0): raise           # перевірка монотонності
        if np.any(X<=0) | np.any(X>=L): raise   # точки всередині
        A=np.sum(F*(L-2*X))/(2*L)
        B=np.sum(F*X)/2
        sabs=sum([F[i]*Abs(x-X[i]) for i in range(len(X))])/2
        return nsimplify(B)+nsimplify(A)*x-sabs
    except:
        print("Помилка входових даних!")
        return None
```

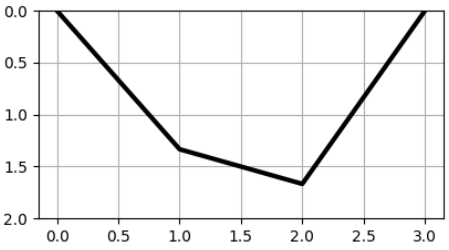
Нехай $L = 3, x_1 = 1\,, x_2 = 2, f_1 = 1\,, f_2 = 2$. Згенеруємо рівняння ламаної, що представлятиме форму струни.

```
X, F, L=np.array([1,2]), np.array([1,2]), 3
x = symbols('x')
dfl=ctl_string(x,X,F,L);
if dfl is None: sys.exit()
else:  print(dfl)
```

```
-x/6-Abs(x-2)-Abs(x-1)/2+5/2
```

Побудуємо зображення деформованої струни.

```
dfline=lambdify(x,dfl,"numpy")
fig,ax = plt.subplots(1,1)
x_=np.linspace(0,L,61)
y_=dfline(x_)
ax.plot(x_,y_,'-k',lw=3)
ax.set_ylim(y_.max(), y_.min())        # розворот донизу осі Y
ax.grid(True)
```



■

Часто зовнішні впливи моделюються кусково-сталими функціями.

Приклад 2. Нехай на ділянках струни $(x_{i-1}, x_i]\,, i = 1, 2, \ldots, n$ діють сталі зовнішні сили f_i, обчислювані на одиницю довжини. Тоді зовнішню силу в рівнянні (11) можна представити у вигляді (5)

$$f(x) = f_1 + \sum_{i=1}^{n-1}(f_{i+1} - f_i) \cdot H(x, x_i).$$

Інтегруючи двічі від 0 до x рівняння (1) з такою правою частиною, і враховуючи, що $\int_0^x Q_r(\xi, x_i)d\xi = \frac{1}{2}(x - x_i) \cdot Q_r(x, x_i)$, отримуємо

$$u(x) = C_1 x + C_2 - f_1 \cdot \frac{x^2}{2} - \frac{1}{2}\sum_{i=1}^{n-1}(f_{i+1} - f_i)(x - x_i)Q_r(x, x_i)\,,$$

З граничної умови $u(0)=0$ відразу випливає, що $C_2=0$. З іншої умови $u(L)=0$ отримуємо, що $C_1=\frac{1}{2}\left[f_1\,L+\frac{1}{L}\sum_{i=1}^{n-1}(f_{i+1}-f_i)(L-x_i)^2\right]$. Після підстановки і спрощень отримуємо

$$u(x)=a\,x^2+b\,x+c-\frac{1}{4}\sum_{i=1}^{n-1}(f_{i+1}-f_i)(x-x_i)|\,x-x_i|\,,\ \text{де} \qquad (13)$$

$$a=-\frac{1}{2}f_1-\frac{1}{4}\sum_{i=1}^{n-1}(f_{i+1}-f_i)\,,$$

$$b=\frac{1}{2}f_1\,L+\frac{1}{2L}\sum_{i=1}^{n-1}(f_{i+1}-f_i)(L^2-L\,x_i+x_i^2)\,,$$

$$c=-\frac{1}{4}\sum_{i=1}^{n-1}(f_{i+1}-f_i)\,x_i^2\,.$$

Для реалізації формули (13) створимо функцію `pwl_string(x,X,F,L)`, яка повертатиме символьний вираз $u(x)$ від символьної змінної `x`, що передається їй першим аргументом. Масив X містить координати внутрішніх точок розбиття, F – масив зовнішніх сталих сил на відповідних ділянках, `L` – довжина струни.

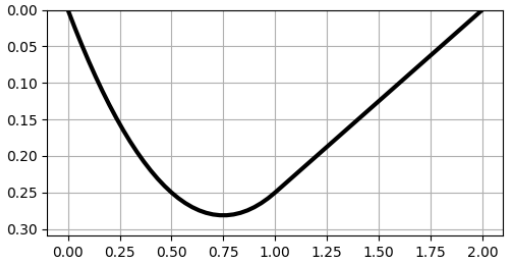
```
import numpy as np
from sympy import symbols,lambdify,Abs,nsimplify
import matplotlib.pyplot as plt
import sys
plt.close('all')
def pwl_string(x,X,F,L):
    """ Прогин струни під дією кусково-сталих сил F[i],
        L - довжина струни, масив X не містить координати
        кінців (0<X[i]<L), len(X)+1==len(F)    """
    try:
        dx=np.diff(X)
        if np.any(dx<=0): raise  # перевірка монотонності
        if np.any(X<=0) | np.any(X>=L): raise  # точки всередині
        a=-nsimplify(F[0]/2+np.sum(F[1:]-F[:-1])/4)
        b=nsimplify(F[0]*L/2+
                    np.sum((F[1:]-F[:-1])*(L**2-L*X+X**2))/(2*L))
        c=-nsimplify(np.sum((F[1:]-F[:-1])*X**2)/4)
        s1=a*x**2+b*x+c
        s2=-sum([(F[i+1]-F[i])*(x-X[i])*Abs(x-X[i])
                                        for i in range(len(X))])/4
        return s1+s2
    except:
        print("Помилка входових даних!")
        return None
```

Нехай $L=2, x_1=1\,, f_1=1\,, f_2=0$. Згенеруємо рівняння функції, графік якої представлятиме форму струни.

```
X, F, L=np.array([1]), np.array([1,0]), 2
x = symbols('x')
dfl=pwl_string(x,X,F,L);
if dfl is None: sys.exit()
else: print(dfl)
-x**2/4+x/4-(1-x)*Abs(x-1)/4+1/4
```

Побудуємо зображення деформованої струни.

```
dfline=lambdify(x,dfl,"numpy")
fig,ax = plt.subplots(1,1)
x_=np.linspace(0,L,61)
y_=dfline(x_)
ymin,ymax=y_.min(),y_.max()
ax.plot(x_,y_,'-k',lw=3)
ax.set_ylim(ymax*1.1, ymin)
ax.grid(True)
```



Зауваження 1. Функція `pwl_string(x,X,F,L)` повертатиме правильний вираз прогинів, навіть, якщо внутрішніх точок розбиття немає, тобто зовнішнє навантаження стале. Для цього входові дані можна підготувати наступною інструкцією `X,F,L=np.array([]),np.array([1]),1`.

Зауваження 2. Якщо одночасно діють зосереджені F_{ctl} і кусково-сталі F_{pwl} навантаження то, в силу лінійності, рівняння (11) можна розв'язати окремо для правих частин F_{ctl} та F_{pwl}, а потім скласти результівні прогини $u_{ctl}(x)$ і $u_{pwl}(x)$. ■

Якщо досліджується диференціальне рівняння зі сталими коефіцієнтами, розв'язок якого можна виразити через інтеграли, то воно схожим з попередніми міркуваннями способом може бути розв'язано з кусково-сталими коефіцієнтами. Кілька наступних прикладів це демонструють.

Прогини балки. Рівняння рівноваги балки, що знаходиться під дією зовнішніх сил, має вигляд [9]

$$E\,J\,u''''_{xxxx} = q(x)\,, \tag{14}$$

де q - зовнішнє зусилля, розраховане на одиницю довжини балки, E і J - модуль пружності матеріалу балки і момент інерції її перерізу. Ось X збігається з віссю балки. Функція $u(x)$ являє вертикальне зміщення точок осі балки, які мали до навантаження горизонтальну координату x.

Приклад 3. Нехай дано балку, яка вільно спирається в кінцевих точках $x_0 = 0$, $x_n = L$, а у внутрішніх точках $\{x_i\}_{i=1}^{n-1}$, $x_0 < x_1 < x_2 < \ldots < x_{n-1} < x_n$ піддається дії поперечно прикладених зосереджених навантажень f_i. Зовнішню силу $q(x)$ в цьому випадку слід задавати виразом $q(x) = \sum_{i=1}^{n-1} f_i \cdot \delta(x, x_i)$, де $\delta(x, x_i)$ - дельта-функція Дірака. Для вільно опертої на краях балки граничні умови мають вигляд $u(0) = 0, u(L) = 0, u''_{xx}(0) = 0, u''_{xx}(L) = 0$, де L - довжина балки.

Аналогічно попереднім прикладам, інтегруючи рівняння (14) з правою частиною $q(x)$ чотири рази від 0 до x і враховуючи (3), отримуємо

$$u(x) = C_0\,x^3 + C_1\,x^2 + C_2\,x + C_3 + \frac{1}{6\,E\,J}\sum_{i=1}^{n-1} f_i\,(x-x_i)^2\,Q_r(x,x_i).$$

З граничної умови $u(0)=0$ випливає $C_3=0$. А з умови $u''_{xx}(0)=0$, враховуючи що $\frac{d^2}{dx^2}(x-x_i)^2\,Q_r(x,x_i) = 6\,Q_r(x,x_i)$, випливає $C_1=0$. З умови $u''_{xx}(L)=0$ маємо $C_0 = -\frac{1}{6\,E\,J\,L}\sum_{i=1}^{n-1} f_i\,(L-x_i)$, а з граничної умови $u(L)=0$ отримуємо $C_2 = \frac{1}{6\,E\,J\,L}\sum_{i=1}^{n-1} f_i\cdot x_i\,(L-x_i)\,(2L-x_i)$. Після підстановки в останню формулу сталих C_i, заміни $(x-x_i)^2\,Q_r(x,x_i) \equiv \frac{1}{2}\,((x-x_i)^3 + |\,x-x_i\,|^3\,)$, і спрощення, отримуємо

$$u(x) = a\,x^3 + b\,x^2 + c\,x + d + \frac{1}{12\,E\,J}\sum_{i=1}^{n-1} f_i\;|\,x-x_i\,|^3\;,\quad \text{де} \qquad (15)$$

$$a = -\frac{1}{12\,E\,J\,L}\sum_{i=1}^{n-1} f_i\,(L-2x_i)\;,\qquad b = -\frac{1}{4\,E\,J}\sum_{i=1}^{n-1} f_i\,x_i,$$

$$c = \frac{1}{12\,E\,J\,L}\sum_{i=1}^{n-1} f_i\,x_i\,(4L^2 - 3\,L\,x_i + 2\,x_i^2)\;,\quad d = -\frac{1}{12\,E\,J}\sum_{i=1}^{n-1} f_i\,x_i^3\,.$$

Для реалізації формули (15) створимо функцію `ctl_beam(x,X,F,ejl)`, яка повертатиме символьний вираз $u(x)$ від символьної змінної `x`, що передається їй першим аргументом. В масивах `X` і `F` передаватимуться абсциси прикладання і величини зосереджених зусиль. Аргумент `ejl` містить трійку параметрів: модуль пружності E матеріалу балки, момент інерції J її перерізу та довжину L.

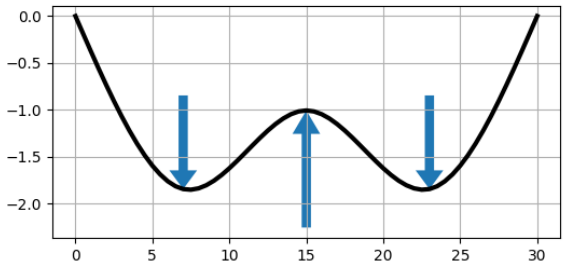
```
import numpy as np
from sympy import symbols,lambdify,Abs,nsimplify
import matplotlib.pyplot as plt
from matplotlib.patches import Arrow
import sys
plt.close('all')
def ctl_beam(x,X,F,ejl):
    """ Прогин балки під дією зосереджених в точках X[i]
        сил F[i] (0<X[i]<L); L - довжина струни, ejl=(E,J,L)
    """
    try:
        E, J, L=ejl
        dx=np.diff(X)
        if np.any(dx<=0): raise                  # перевірка монотонності
        if np.any(X<=0) | np.any(X>=L): raise    # точки всередині
        a=-nsimplify(np.sum(F*(L-2*X))/(12*E*J*L))
        b=-nsimplify(np.sum(F*X)/(4*E*J))
        c=nsimplify(np.sum(F*X*(4*L**2-3*L*X+2*X**2))/(12*E*J*L))
        d=-nsimplify(np.sum(F*X**3))/(12*E*J)
        ss=sum([nsimplify(F[i]/(12*E*J))*Abs(x-nsimplify(X[i]))**3
                                          for i in range(len(X))])
        return a*x**3+b*x**2+c*x+d+ss
    except:
        print("Помилка входових даних!")
        return None
```

Нехай $L=30, E=2, J=54,\ x_1=7, x_2=15, x_3=23,\ f_1=-4\,, f_2=5, f_3=-4.$ Згенеруємо рівняння функції, графік якої представлятиме форму осі балки.

```
X, F, E, J, L=np.array([7,15,23]), np.array([-4,5,-4]), 2, 54, 30
x = symbols('x')
dfl=ctl_beam(x,X,F,(E,J,L));
if dfl is None: sys.exit()
else:  print(dfl)
5*x**2/48-25*x/8-Abs(x-23)**3/324+5*Abs(x-15)**3/1296-
                                        Abs(x-7)**3/324+3685/144
```

Побудуємо зображення деформованої балки і схему навантажень.

```
dfline=lambdify(x,dfl,"numpy")
fig, ax = plt.subplots(1,1)
x_=np.linspace(0,L,61)
ax.plot(x_,dfline(x_),'-k',lw=3)        # графік деформованої осі балки
ax.grid(True)
arrows=[[X[i],dfline(X[i])-F[i]/4,0,F[i]/4] for i in range(len(F))]
for i in range(len(F)):                 # схема навантаження
    ax.add_patch(Arrow(*arrows[i],width=3))
```



Зауваження. Інші граничні умови змінять в формулі (15) значення коефіцієнтів a, b, c, d, залишаючи незмінним член $\frac{1}{12\,E\,J}\sum_{i=1}^{n-1} f_i \mid x-x_i \mid^3$.

Зауваження. Аналогічно можна отримати формулу для обчислення прогинів балки під дією кусково-сталої зовнішньої сили $f(x)=f_1+\sum_{i=1}^{n-1}(f_{i+1}-f_i)\cdot H(x,x_i)$. Результівна формула міститиме суму кубічного полінома $a\,x^3+b\,x^2+c\,x+d$ і доданка, на кшталт, $K\cdot\sum_{i=1}^{n-1} f_i\cdot(x-x_i)^3 \mid x-x_i \mid$ з деяким скалярним множником K.

■

Стаціонарна теплопровідність в багатошаровій плоскій стінці. Загальне однорідне рівняння теплопровідності має вигляд [7]

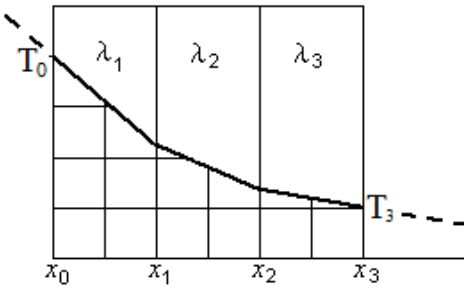
$$c\,\rho\,\frac{\partial T}{\partial t}=\operatorname{div}(\lambda\cdot\operatorname{grad}T)\,, \tag{16}$$

де c – теплоємність речовини, ρ – густина, λ – коефіцієнт внутрішньої теплопровідності. В стаціонарних задачах ліва частина дорівнює нулю і в одновимірному випадку рівняння набуває вигляду $\frac{d}{dx}\left(\lambda\frac{dT}{dx}\right)=0$, де $T(x)$ – температура середовища в точці з координатою x. Якщо коефіцієнт теплопровідності $\lambda(x)$ є кусково-сталим, то в наведеній формі рівняння не можна застосовувати, бо в дужках може стояти розривна величини, і його слід переписати у вигляді

$$-\lambda(x)\frac{dT}{dx}=q\,, \tag{17}$$

а невідома стала q являтиме густину теплового потоку вздовж осі X, яку слід знаходити з граничних умов. Знак в (17) можна взяти будь-яким, але при зробленому виборі потік тепла від гарячіших точок до холодніших буде позитивним, тобто $q > 0$.

Приклад 4. Дослідити стаціонарний розподіл температури в багатошаровій плоскій плиті, яка складається з n плоских контактуючих необмежених пластин. Межі шарів мають координати $0 = x_0 < x_1 < \cdots < x_n$, їх товщини $d_i = x_i - x_{i-1}$, а коефіцієнти теплопровідності λ_i $(i = 1, \dots, n)$. Тепло передається через пластини нормально до обмежуючих їх площин. Температура зовнішніх поверхонь незмінна і відома $T(x_0) = T_0,\ T(x_n) = T_n$.



В нашому випадку $\lambda(x) = \begin{cases} \lambda_1,\ x_0 \le x < x_1 \\ \cdots \\ \lambda_n,\ x_{n-1} \le x \le x_n \end{cases}$ - кусково-стала функція, і рівняння (17) має вигляд $\frac{dT}{dx} = -\frac{q}{\lambda(x)}$, де кусково-стала права частина може бути переписана з використанням функції Гевісайда (див. ф. (5)) у вигляді

$$\frac{dT}{dx} = -q \cdot k(x)\,, \text{ де } k(x) = \frac{1}{\lambda_1} + \sum_{i=1}^{n-1} \left(\frac{1}{\lambda_{i+1}} - \frac{1}{\lambda_i}\right) H(x, x_i)\,.$$

Інтегруючи від $x_0 = 0$ до x ліву і праву частини останнього рівняння, матимемо

$$T(x) = T_0 - q \cdot \int_{x_0}^{x} k(\xi)\, d\xi = T_0 - q \cdot I(x), \tag{18}$$

$$\text{де } T_0 = T(x_0) \text{ і } I(x) = \int_{x_0}^{x} k(\xi)\, d\xi = \frac{x - x_0}{\lambda_1} + \sum_{i=1}^{n-1} \left(\frac{1}{\lambda_{i+1}} - \frac{1}{\lambda_i}\right) Q_r(x, x_i)\,.$$

Для визначення невідомої сталої q підставимо в (18) граничну умову $T(x_n) = T_n$.

$$T_n = T_0 - q \cdot I(x_n)\,, \text{ де } I(x_n) = \frac{x_n - x_0}{\lambda_1} + \sum_{i=1}^{n-1} \left(\frac{1}{\lambda_{i+1}} - \frac{1}{\lambda_i}\right)(x_n - x_i)$$

Згадуючи формулу (9), матимемо $I(x_n) = \sum_{i=1}^{n} \frac{x_i - x_{i-1}}{\lambda_i}$. Тому

$$q = \frac{T_0 - T_n}{I(x_n)} = \frac{T_0 - T_n}{\sum_{i=1}^{n}(x_i - x_{i-1})/\lambda_i} = \frac{T_0 - T_n}{\sum_{i=1}^{n} d_i/\lambda_i}\,. \tag{19}$$

З (18), враховуючи (10), матимемо

$$T(x) = T_0 - \frac{q}{2} \cdot I(x_n) - \frac{q}{2}\left(\frac{1}{\lambda_1}(x - x_0) + \frac{1}{\lambda_n}(x - x_n) + \sum_{i=1}^{n-1}\left(\frac{1}{\lambda_{i+1}} - \frac{1}{\lambda_i}\right) \cdot |\, x - x_i|\right) =$$

$$= \frac{T_0+T_n}{2} + \frac{T_n-T_0}{2\sum_{i=1}^{n} d_i/\lambda_i}\left(\frac{x-x_0}{\lambda_1} + \frac{x-x_n}{\lambda_n} + \sum_{i=1}^{n-1}\left(\frac{1}{\lambda_{i+1}} - \frac{1}{\lambda_i}\right)|\,x-x_i\,|\right)$$

Для символьних обчислень цей вираз зручніше записати у вигляді

$$T = a_0 + a_1 x + \frac{T_n-T_0}{2\sum_{i=1}^{n} d_i/\lambda_i}\sum_{i=1}^{n-1}\left(\frac{1}{\lambda_{i+1}} - \frac{1}{\lambda_i}\right)|x-x_i|\,, \qquad (20)$$

де $a_0 = \frac{T_0+T_n}{2} - \frac{T_n-T_0}{2\sum_{i=1}^{n} d_i/\lambda_i}\cdot\left(\frac{x_0}{\lambda_1} + \frac{x_n}{\lambda_n}\right)$, $a_1 = \frac{T_n-T_0}{2\sum_{i=1}^{n} d_i/\lambda_i}\cdot\left(\frac{1}{\lambda_1} + \frac{1}{\lambda_n}\right)$.

Для реалізації ф. (20) створимо функцію `ml_plate((x,X,Lmb,Tb)`, яка повертатиме символьний вираз $T(x)$ від символьної змінної `x`, що передається їй першим аргументом. В масиві `X` передаватимуться абсциси $x_0, x_1, x_2, \ldots, x_n$, а в масиві `Lmb` – величини $\lambda_1, \lambda_2, \ldots, \lambda_n$, `Tb` – кортеж/список значень лівої і правої граничної температури.

```
import numpy as np
from sympy import symbols, lambdify, Abs, nsimplify, simplify
import matplotlib.pyplot as plt
import sys
def ml_plate(x, X, Lmb, Tb):
     """  Температура в багатошаровій плоскій плиті,
          X[i] абсциси стінок, Lmb - масив коефіцієнтів
          теплопровідності, len(Lmb)=len(X)-1   """
    try:
        dx=np.diff(X)
        if np.any(dx<=0): raise           # перевірка монотонності
        if np.any(X[:-1]>=X[-1]): raise   # точки всередині
        T0,Tn=Tb
        x0,xn=X[0],X[-1]
        l1,ln=Lmb[0],Lmb[-1]
        dl=(dx/Lmb).sum()
        k=nsimplify((Tn-T0)/2/dl)
        a0=nsimplify((T0+Tn)/2-k*(x0/l1+xn/ln))
        a1=nsimplify(k*(1/l1+1/ln))
        ss=sum([nsimplify(k*(1/Lmb[i]-1/Lmb[i-1]))*
                    Abs(simplify(x-nsimplify(X[i])))
                                    for i in range(1,len(X)-1)])
        return a0+a1*x+ss
    except:
        print("Помилка входових даних!")
        return None
```

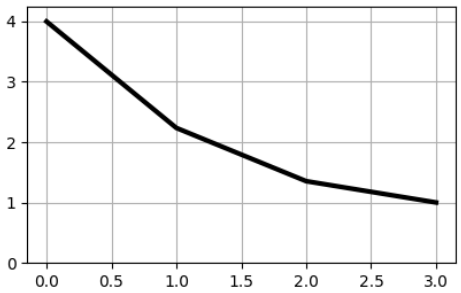
Нехай $\{x_i\}_{i=0}^{3} = \{0,1,2,3\}$, $\{\lambda_i\}_{i=1}^{3} = \{1,2,5\}$, $T_0 = 4, T_3 = 1$. Згенеруємо рівняння функції, яка представлятиме температуру в плиті.

```
if __name__ == '__main__':
  X, Lmbda,T0,Tn=np.array([0,1,2,3]), np.array([1,2,5]), 4, 1
  x = symbols('x')
  tmp= ml_plate(x, X, Lmbda, (T0,Tn));
  if tmp is None: sys.exit()
  else: print(tmp)
```

```
-18*x/17+9*Abs(x-2)/34+15*Abs(x-1)/34+103/34
```

Побудуємо графік температури.

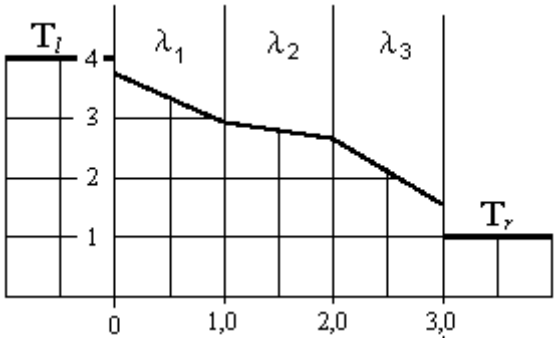
```
plt.close('all')
Temp=lambdify(x, tmp, "numpy")
fig, ax = plt.subplots(1,1)
x_=np.linspace(0,X[-1],61)
ax.plot(x_,Temp(x_),'-k',lw=3)
ax.set_ylim(0, 4.1)
ax.grid(True)
```



Збережіть файл з ім'ям `multilayer_plate.py`. В наступному прикладі з нього ми імпортуватимемо функцію `ml_plate()`.

■

Вправа. Дослідити стаціонарний розподіл температури в багатошаровій плоскій плиті, якщо на зовнішніх границях $x = x_0 = 0$, $x = x_n$ здійснюється теплообмін з навколишнім середовищем по закону Ньютона. В одновимірному випадку він записується у вигляді $T'_x(x_0) = h_l(T_0 - T_l)$, $T'_x(x_n) = h_r(T_r - T_n)$, де $h_l = \frac{\alpha_l}{\lambda_1}$, $h_r = \frac{\alpha_r}{\lambda_n}$ - коефіцієнти теплообміну, α_l, α_r – коефіцієнти тепловіддачі до лівого і правого навколишніх середовищ з температурами T_l, T_r, а T_0, T_n – температури лівої і правої стінок плити (спочатку невідомі).



Вказівка. Рівняння теплопровідності для цієї задачі має вигляд $\frac{dT}{dx} = -q \cdot \frac{1}{\lambda(x)}$. Розв'язок можна побудувати за ф. (19), якщо знати граничні температури T_0, T_n, зокрема буде $q = \frac{T_0 - T_n}{\sum_{i=1}^{n}(x_i - x_{i-1})/\lambda_i} = \gamma \cdot (T_0 - T_n)$. Для визначення T_0, T_n є дві граничні умови. На ділянці $x < x_1$ і, зокрема, в точці $x = x_0$, температура задовольняє рівнянню $\lambda_1 \frac{dT}{dx} = -q$, де виконується також гранична умова $T'_x(x_0) = h_l(T_0 - T_l)$. Тому $\lambda_1 h_l(T_0 - T_l) = -q$. Аналогічно, в точці $x = x_n$ маємо $\lambda_n \frac{dT}{dx} = -q$ і другу граничну умову, що дає рівність $\lambda_n h_r(T_r - T_n) = -q$. Замінюючи q його поданням через T_0, T_n, отримуємо систему рівнянь щодо невідомих температур T_0, T_n.

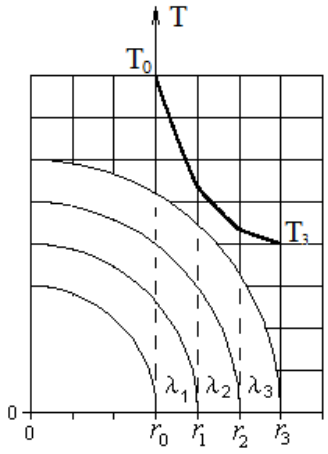
$$\begin{cases}\lambda_1 h_l(T_0 - T_l) = -(T_0 - T_n)\,\gamma \\ \lambda_n h_r(T_r - T_n) = -(T_0 - T_n)\,\gamma\end{cases},$$

де $\gamma = 1/\sum_{i=1}^{n}(x_i - x_{i-1})/\lambda_i$.

На попередньому рисунку показано графік температури в тришаровій плиті при $T_l = 4, T_r = 1$, $\{x_i\}_{i=0}^{3} = \{0,1,2,3\}$, $\{\lambda_i\}_{i=1}^{3} = \left\{\frac{2}{3}, 2, \frac{1}{2}\right\}$, $\alpha_l = 2, \alpha_r = 1$.

Стаціонарна теплопровідність в багатошаровій циліндричній трубі. Однорідне рівняння теплопровідності (16) в циліндричній системі координат [8], коли температура T і коефіцієнт внутрішньої теплопровідності λ залежать лише від радіуса r, набуває вигляду $\frac{1}{r}\frac{\partial}{\partial r}(r\,\lambda\,\frac{\partial u}{\partial r}) = 0$, або $\lambda(r) r \frac{dT}{dr} = -q$, де q – поки що невідома стала.

Приклад 5. Дослідити розподіл температури в стінці багатошарової труби зі сталими в кожному шарі коефіцієнтами внутрішньої теплопровідності $\lambda(r) = \lambda_i$ $(r_{i-1} < r < r_i\,, i = 1,2,\dots,n)$.



Для цієї задачі рівняння зручно переписати у вигляді $\frac{d\,T}{d\,r} = -\frac{q}{r\,\lambda(r)}$, і його права частина буде кусковою функцією (але не кусково-сталою). Для подальшого нам буде зручно розв'язувати трохи узагальнене рівняння

$$\frac{d\,T}{d\,r} = -q \cdot k(r) f(r)\,, \tag{21}$$

де $f(r)$ – довільна функція, а $k(r)$–кусково-стала. Проінтегруємо (21) від r_0 до r:

$$T(r) = T_0 - q \cdot \int_{r_0}^{r} k(\xi)\, f(\xi)\, d\,\xi \;\text{, де } T_0 = T(r_0)\,. \tag{22}$$

Тоді для інтеграла в (22) матимемо

$$I(r) = \int_{r_0}^{r} k(\xi)\, f(\xi)\, d\,\xi = \int_{r_0}^{r} k(\xi)\, d\,F(\xi)\,,$$

де $F(r)$ – первісна функції $f(r)$. Зробимо заміну $F(r) = t$, або $r = F^{-1}(t)$, де F^{-1} – обернена до F функція (припускаємо, що на ділянці $r_0 \le r \le r_n$ вона існує і є монотонно зростаючою). Тоді кусково-стала функція $k(r)$ перетворюється на іншу кусково-сталу функцію $\tilde{k}(t)$ з тими же заначеннями k_i, але іншими точками розриву $t_i = F(r_i)$. Дійсно, в припущенні монотонності F, маємо

$$k(r)=\begin{cases}k_1,\ r<r_1\\ k_2,\ r_1\le r<r_2\\ \dots\\ k_n,\ r\ge r_{n-1}\end{cases}=\begin{cases}k_1,\ F^{-1}(t)<F^{-1}(t_1)\\ k_2,\ F^{-1}(t_1)\le F^{-1}(t)<F^{-1}(t_2)\\ \dots\\ k_n,\ F^{-1}(t)\ge F^{-1}(t_{n-1})\end{cases}=\begin{cases}k_1,\ t<t_1\\ k_2,\ t_1\le t<t_2\\ \dots\\ k_n,\ t\ge t_{n-1}\end{cases}=$$

$$=k_1+\sum_{i=1}^{n-1}(k_{i+1}-k_i)H(t,t_i)\ =\tilde{k}(t)\,.$$

Тоді, позначаючи $t_0=F(r_0)$, що менше будь-якого $t_i=F(r_i)$, матимемо

$$I(r)=\int_{r_0}^{r}k(\xi)d\,F(\xi)=\int_{t_0}^{t}\tilde{k}(\tau)d\tau=\tilde{I}(t)\,,$$

де $\tilde{I}(t)$ є первісною кусково-сталої функції $\tilde{k}(t)$,

Покладемо в рівнянні (21) $k(r)=1/\lambda(r),\ f(r)=\frac{1}{r},\ F(r)=\ln r$. Тоді, відповідно до (22), його розв'язком буде $T(r)=T_0-q\,I(r)=T_0-q\,\tilde{I}(t)$. Використовуючи граничну умову $T(r_n)=T_n$, отримуємо $q=(T_0-T_n)/\tilde{I}(t_n)$, де $t_n=F(r_n)$. Враховуючи (9) і позначення $t_i=F(r_i)$, матимемо

$$q=\frac{T_0-T_n}{\tilde{I}(t_n)}=\frac{T_0-T_n}{\sum_{i=1}^{n}k_i(t_i-t_{i-1})}=\frac{T_0-T_n}{\sum_{i=1}^{n}\frac{1}{\lambda_i}\ln\frac{r_i}{r_{i-1}}}\,.\qquad(23)$$

Підставляючи в розв'язок $T(r)=T_0-q\,I(r)=T_0-q\,\tilde{I}(t)$ замість $\tilde{I}(t)=\int_{t_0}^{t}\tilde{k}(\tau)\,d\,\tau$ його представлення у вигляді (10), отримуємо

$$T(r)=T_0-\frac{q}{2}I(t_n)-\frac{q}{2}\left(k_1(t-t_0)+k_n(t-t_n)+\sum_{i=1}^{n-1}(k_{i+1}-k_i)\cdot|\,t-t_i\,|\right)=$$

$$=\frac{T_0+T_n}{2}-\frac{q}{2}\left(k_1(t-t_0)+k_n(t-t_n)+\sum_{i=1}^{n-1}(k_{i+1}-k_i)\cdot|\,t-t_i\,|\right).$$

Тут було використано рівність $q\cdot I(t_n)=T_0-T_n$. Враховуючи, що $t=\ln r$ та $t_i=F(r_i)$, остаточно отримуємо

$$T(r)=\frac{T_0+T_n}{2}-\frac{q}{2}\left(\frac{1}{\lambda_1}\ln\frac{r}{r_0}+\frac{1}{\lambda_n}\ln\frac{r}{r_n}+\sum_{i=1}^{n-1}\left(\frac{1}{\lambda_{i+1}}-\frac{1}{\lambda_i}\right)\cdot\left|\ln\frac{r}{r_i}\right|\right),\qquad(24)$$

де q визначається в (23). Праву частину (24) можна також записати у вигляді

$$T(r)=a_0+a_1\ln r-\frac{q}{2}\sum_{i=1}^{n-1}\left(\frac{1}{\lambda_{i+1}}-\frac{1}{\lambda_i}\right)\cdot\left|\ln\frac{r}{r_i}\right|\,,\ \text{де}\qquad(25)$$

$$a_0=\frac{T_0+T_n}{2}-\frac{T_n-T_0}{2\sum_{i=1}^{n}\frac{1}{\lambda_i}\ln\frac{r_i}{r_{i-1}}}\left(\frac{\ln r_0}{\lambda_1}+\frac{\ln r_{n-1}}{\lambda_n}\right),\ a_1=\frac{T_n-T_0}{2\sum_{i=1}^{n}\frac{1}{\lambda_i}\ln\frac{r_i}{r_{i-1}}}\left(\frac{1}{\lambda_1}+\frac{1}{\lambda_n}\right).$$

Формули (24) або (25) при $r_0\le r\le r_n$ надають розподіл температури в багатошаровій циліндричній трубі, коли температура T_0 внутрішньої і T_n зовнішньої поверхонь відомі.

Зауваження. Порівняйте розв'язки (20) і (25). Другий, фактично, отримано з (20) заміною абсциси x на $\ln r$, а величин x_i на $\ln r_i$. Інакше кажучи, в рівнянні ламаної $y = l(x)$ зроблено заміну незалежного аргументу на монотонно зростаючу функцію $\ln r$. Взагалі-то, у нас існує дві можливості суперпозиції функції $l(x)$, що являє ламану, з монотонно-зростаючою функцією $f(r)$: $l \circ f = l(f(r))$ і $f \circ l = f(l(x))$. Комбінація $l(f(r))$ зустрічалася в поточному прикладі, а також в п. 10.3, наприклад, в формулах «обрізання» функцій ($mn(f(x), y_0$) і інших). Суперпозиція $f(l(x))$ використовувалася там же, наприклад, при побудові фінітної функції $\sin(\pi\, l(x))$.

Для реалізації ф. (25) створимо функцію `ml_pipe(r,R,Lmb,Tb)`, яка повертатиме символьний вираз $T(r)$ від символьної змінної `r`, що передається їй першим аргументом. В масиві `R` передаватимуться радіуси $r_0, r_1, r_2, \ldots, r_n$, а в масиві `Lmb` – величини $\lambda_1, \lambda_2, \ldots, \lambda_n$, `Tb` – кортеж/список значень лівої і правої граничної температури. Враховуючи наведене зауваження, функція `ml_pipe()` повинна перетворювати масив радіусів `R` на масив `X=np.log(R)`, і викликати функцію `ml_plate()` з першим (символьним) аргументом `log(r)`. Аргументи `Lmb` і `Tb` передаватимуться незмінними.

```
import numpy as np
from sympy import symbols, lambdify, log
import matplotlib.pyplot as plt
import matplotlib.colors as mc
from multilayer_plate import ml_plate
from surfaces02 import cut_out_cylinder
import sys
plt.close('all')
def ml_pipe(r,R,Lmb,Tb):      # обчислення за ф. (25), використовуючи ф. (20)
    X=np.log(R)
    x=log(r)
    return (ml_plate(x,X,Lmb,Tb)).evalf()
```

Нехай $\{r_i\}_{i=0}^3 = \{3,4,5,6\}$, $\{\lambda_i\}_{i=1}^3 = \{1,2,5\}$, $T_0 = 8, T_3 = 4$. Згенеруємо рівняння функції, яка представлятиме температуру в трубі.

```
R,Lmbda,T0,Tn=np.array([3,4,5,6]),np.array([1,2,5]),8,4
r = symbols('r')
tmp=ml_pipe(r,R,Lmbda,(T0,Tn));
if tmp is None: sys.exit()
else: print(tmp)
-5.50814775067469*log(r)+1.37703693766867*Abs(log(r/5))+
         2.29506156278112*Abs(log(r/4))+12.6876449872721
```

Тепер побудуємо графік температури в залежності від радіуса r.

```
Temp=lambdify(r,tmp,"numpy")
fig,ax = plt.subplots(1,1)
r0,rn=R[0],R[-1]
r_=np.linspace(r0,rn,91)
ax.plot(r_,Temp(r_),'-k',lw=3)          # наступний рисунок ліворуч
ax.grid(True)
```

Для тренування користувача, побудуємо кольорові графіки розподілу температури по перерізам труби: поперечному та поздовжньому. Графік температури по поперечному перерізу будується як зафарбований контурний графік.

```
u_=np.linspace(0,np.pi/2,31)[:,None]
X=r_*np.cos(u_)
Y=r_*np.sin(u_)
F=Temp(np.sqrt(X**2+Y**2))
vals=np.linspace(Tn,T0,41)
fig,ax = plt.subplots(1,1)
cp=ax.contourf(X,Y,F, vals, cmap='hot')
ax.contour(X,Y,np.sqrt(X**2+Y**2), R, linewidths=2,colors='b')
ax.set(xlim=(0,rn+0.5),ylim=(0,rn+0.5))    # наступний рис. всередині
ax.set_aspect(1)
plt.colorbar(cp)
```

Щоб зобразити температуру на вертикальних перерізах труби, використаємо параметричні рівняння поверхні «розрізаної» труби, які генерує функція `cut_out_cylinder()`, створена нами в файлі `surfaces.py` в прикладі 11 п. 10.3.

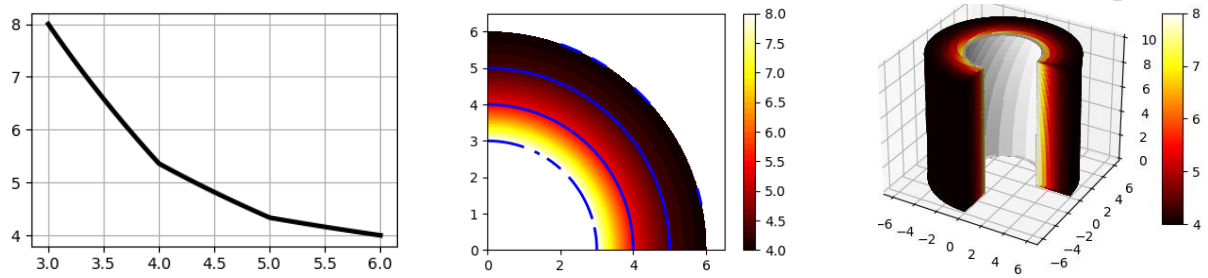
```
H,amax=10,3*np.pi/2                    # висота і кутова ширина труби
u_=np.linspace(0,2*np.pi,200)
v_ = np.linspace(-np.pi/2,np.pi/2,150)
U,V=np.meshgrid(u_,v_)
flist=cut_out_cylinder(r0,rn,H,amax)  # парам. рівняння поверхні
```

Створимо сітку точок `Xd,Yd,Zd` на поверхні, обчислимо для них температуру `Temp_r`, і з неї створимо масив кольорів, що їм відповідає.

```
Xd,Yd,Zd=[f(U,V) for f in flist]       # сітка точок на поверхні
Temp_r=Temp(np.sqrt(Xd**2+Yd**2))      # температура в точках сітки
cNorm=mc.Normalize()
Norm_Temp=cNorm(Temp_r)                # нормований масив температур
Clr=plt.cm.hot(Norm_Temp)              # масив hot кольорів
```

Тепер побудуємо тривимірне зображення поверхні «розрізаної» труби, розфарбованої відповідно до значень температури її точок (наступний рисунок праворуч).

```
fig= plt.figure()
ax = fig.add_subplot(111, projection='3d')
surf=ax.plot_surface(Xd,Yd,Zd,rstride=2,cstride=1,facecolors=Clr)
ax.set_box_aspect(aspect =(2*rn,2*rn,H))
sm=plt.cm.ScalarMappable(cmap=plt.cm.hot,norm=cNorm)
fig.colorbar(sm, shrink=0.75, aspect=10)          # шкала кольорів
```



Зауваження. Формулу (25) можна використовувати і для одношарової труби, розглядаючи її як багатошарову з однаковими значеннями $\lambda_i = \lambda$ всіх шарів. Тоді з (23) матимемо $q = \frac{\lambda\,(T_0 - T_1)}{\ln\frac{r_1}{r_0}}$, і з (24) отримуємо $T(r) = \frac{1}{2}\left(T_0 + T_1 - \frac{T_0 - T_1}{\ln\frac{r_1}{r_0}}\,\ln\frac{r^2}{r_0\,r_1}\right)$.

Приклад 6. Знайти стаціонарний сферично-симетричний розподіл температури в багатошаровій кулі з кульовою порожниною (внутрішні джерела тепла відсутні). Коефіцієнти внутрішньої теплопровідності шарів λ_i $(i = 1, \dots, n)$, внутрішній і зовнішний радіуси r_0, r_n і радіуси поверхонь контакту шарів $r_1 < \dots < r_{n-1}$ відомі. Температури внутрішньої $T(r_0) = T_0$ і зовнішньої $T(r_n) = T_n$ поверхонь задано.

Р о з в ' я з а н н я . Рівняння теплопровідності, що описує зазначену ситуацію, має вигляд $\frac{\partial}{\partial r}\left(-\lambda(r)\,r^2\,\frac{\partial T}{\partial r}\right) = 0$ або

$$\frac{\partial T}{\partial r} = -q\,\frac{1}{\lambda(r)}\cdot\frac{1}{r^2}. \qquad (26)$$

Тоді в рівнянні (21) слід покласти $f(r) = \frac{1}{r^2}$, $F(r) = -\frac{1}{r}$ і, по-аналогії з прикладом 5, зробити заміни $t = -\frac{1}{r}$, $t_i = -\frac{1}{r_i}$. З (23) випливатиме, що $q = \frac{T_0 - T_n}{\sum_{i=1}^{n}\frac{1}{\lambda_i}\left(\frac{1}{r_{i-1}} - \frac{1}{r_i}\right)}$, а з формули $T(r) = T_0 - q\,\tilde{I}(t)$ отримуємо

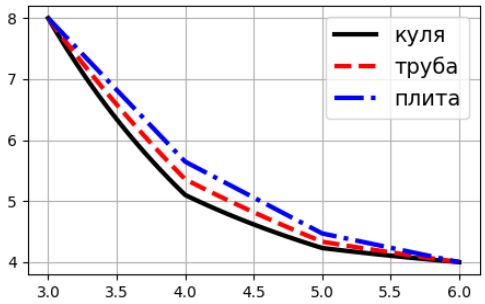
$$T(r) = \frac{T_0 + T_n}{2} - \frac{q}{2}\left[\frac{1}{\lambda_1}\left(\frac{1}{r_0} - \frac{1}{r}\right) + \frac{1}{\lambda_n}\left(\frac{1}{r_n} - \frac{1}{r}\right) + \sum_{i=1}^{n-1}\left(\frac{1}{\lambda_{i+1}} - \frac{1}{\lambda_i}\right)\cdot\left|\frac{1}{r_i} - \frac{1}{r}\right|\right]. \qquad (27)$$

Насправді, для побудови графіка температури формула (27) нам не знадобиться, бо достатньо функції `ml_plate()`, яка реалізує обчислення за формулою (20), і в якій слід зробити заміни $x_i \to -1/r_i$ та $x \to -1/r$. Це реалізується наступною функцією

```
def ml_globe(r,R,Lmb,Tb):
    X=-1/R
    x=-1/r
    return (ml_plate(x,X,Lmb,Tb)).evalf()
```

Всі інші інструкції повторюватимуть код прикладу 5 (за виключенням побудови графіків функціонального забарвлення перерізів).

На наступному рисунку наведено графік температури в тришаровій порожнистій кулі у випадку коли $\{r_i\}_{i=0}^{3} = \{3,4,5,6\}$, $\{\lambda_i\}_{i=1}^{3} = \{1,2,5\}$, $T_0 = 8$, $T_3 = 4$. Тобто це ті самі дані, що і в прикладі 5. Для порівняння червоною штриховою лінією показано графік температури в стінці тришарової труби, отриманий для тих це входових даних, а синім штрих-пунктиром – температуру в тришаровій плиті з такими же граничними температурами, товщинами шарів і коефіцієнтами теплопровідності.



Приклад 7. Поле двошарового сферичного конденсатора. Знайти потенціал $u(r)$ електростатичного поля сферичного конденсатора, заповненого неоднорідним діелектриком з діелектричною сталою

$$\varepsilon = \begin{cases} \varepsilon_1 \ , \ r_0 < r < r_1 \\ \varepsilon_2 \ , \ r_1 < r < r_2 \end{cases} \tag{28}$$

Розв'язання. Вважатимемо, що заряди всередині діелектрика відсутні, а на поверхні заданий електростатичний потенціал $u(r_0) = u_0,\ u(r_2) = u_2$. Фактично, слід розв'язувати крайову задачу

$$\begin{aligned} -\nabla \cdot (\varepsilon_1 \nabla u_l) = 0 \quad \text{при} \quad r_0 < r < r_1, \\ -\nabla \cdot (\varepsilon_2 \nabla u_r) = 0 \quad \text{при} \quad r_1 < r < r_2 \ , \end{aligned} \tag{29}$$

де $u_l(r)$ і $u_r(r)$ задовольняють при $r = r_0$ і $r = r_2$ граничним умовам

$$u_l(r_0) = u_0,\ u_r(r_2) = u_2 \ , \tag{30}$$

і при $r = r_1$ умовам спряження (неперервність потенціалу u і вектора електричної індукції $\boldsymbol{D} = -\varepsilon \ \mathrm{grad}\ u$), тобто

$$u_l(r_1) = u_r(r_1) \ , \varepsilon_1 \frac{\partial u_l}{\partial r}(r_1) = \varepsilon_2 \frac{\partial u_r}{\partial r}(r_1) \ . \tag{31}$$

Аналітичний розв'язок цієї задачі при $u_0 = 1 \ , u_2 = 0$ відомий [10]

$$u_l = 1 + A \cdot \left(\frac{1}{r} - \frac{1}{r_0}\right) \ , u_r = \frac{\varepsilon_1}{\varepsilon_2} A \cdot \left(\frac{1}{r} - \frac{1}{r_2}\right) \ , A = 1 \Big/ \left(\frac{1}{r_0} - \frac{1}{r_1} - \frac{\varepsilon_1}{\varepsilon_2}\left(\frac{1}{r_2} - \frac{1}{r_1}\right)\right). \tag{32}$$

Записуючи в (29) дивергенцію і градієнт в сферичних координатах, і враховуючи сферичну симетрію, отримуємо $\frac{\partial}{\partial r}\left(-\varepsilon_i r^2 \frac{\partial u}{\partial r}\right) = 0 \ \ (i = 1{,}2)$. Через виконання умов стикування (31), замість двох рівнянь можна використати одне

$$-\varepsilon(r)\, r^2 \frac{\partial u}{\partial r} = q \ , \tag{33}$$

де q – константа, $u = \begin{cases} u_l(r) \ , \ r_0 < r \le r_1 \\ u_r(r) \ , \ r_1 \le r < r_2 \end{cases}$, а $\varepsilon(r)$ – шматково-стала функція (28).

Рівняння (33) по формі співпадає з (26). З ф. (19), в якій слід зробити заміни $x_i \to 1/r_i$ (а також замінити позначення сталих λ_i і T_i), матимемо

$$q = \frac{u_0 - u_2}{\frac{1}{\varepsilon_1}\left(\frac{1}{r_0} - \frac{1}{r_1}\right) + \frac{1}{\varepsilon_2}\left(\frac{1}{r_1} - \frac{1}{r_2}\right)} = (u_0 - u_2)\ \varepsilon_1 A \ , \tag{34}$$

де А визначено в (32). З (18) маємо $u(r) = u_0 - q\ I(r)$, де $I(r)$ можна обчислити за формулою (6), в якій x слід замінити на $-1/r$, тобто

$$I(r) = \frac{1}{\varepsilon_1}\left(-\frac{1}{r} + \frac{1}{r_0}\right) + \left(\frac{1}{\varepsilon_2} - \frac{1}{\varepsilon_1}\right) \cdot Q_r(-\frac{1}{r}, -\frac{1}{r_1}).$$

Тоді $u(r)$ набуде вигляду

$$u = u_0 - (u_0 - u_2)\,\varepsilon_1 A\left[\frac{1}{\varepsilon_1}\left(-\frac{1}{r} + \frac{1}{r_0}\right) + \left(\frac{1}{\varepsilon_2} - \frac{1}{\varepsilon_1}\right) \cdot Q_r\left(-\frac{1}{r}, -\frac{1}{r_1}\right)\right]$$

При $r < r_1$ маємо $Q_r\left(-\frac{1}{r}, -\frac{1}{r_1}\right) = 0$, і тому

$$u(r) = u_l(r) = u_0 + A \cdot (u_0 - u_2) \cdot \left(\frac{1}{r} - \frac{1}{r_0}\right), \text{при } r_0 \le r \le r_1.$$

При $r > r_1$ маємо $Q_r\left(-\frac{1}{r}, -\frac{1}{r_1}\right) = \frac{1}{r_1} - \frac{1}{r}$. Тоді

$$u(r) = u_r(r) = u_0 - (u_0 - u_2)\,\varepsilon_1 A\left[\frac{1}{\varepsilon_1}\left(-\frac{1}{r} + \frac{1}{r_0}\right) + \left(\frac{1}{\varepsilon_2} - \frac{1}{\varepsilon_1}\right) \cdot \left(\frac{1}{r_1} - \frac{1}{r}\right)\right] =$$

$$= u_0 - \frac{u_0 - u_2}{\frac{1}{\varepsilon_1}\left(\frac{1}{r_0} - \frac{1}{r_1}\right) + \frac{1}{\varepsilon_2}\left(\frac{1}{r_1} - \frac{1}{r_2}\right)}\left[\frac{1}{\varepsilon_1}\left(\frac{1}{r_0} - \frac{1}{r_1}\right) + \frac{1}{\varepsilon_2}\left(\frac{1}{r_1} - \frac{1}{r_2}\right) + \frac{1}{\varepsilon_2}\left(\frac{1}{r_2} - \frac{1}{r}\right)\right] =$$

$$= u_0 - (u_0 - u_2)\left[1 + \frac{\varepsilon_1 A}{\varepsilon_2}\left(\frac{1}{r_2} - \frac{1}{r}\right)\right] = u_2 + (u_0 - u_2)\frac{\varepsilon_1 A}{\varepsilon_2}\left(\frac{1}{r} - \frac{1}{r_2}\right).$$

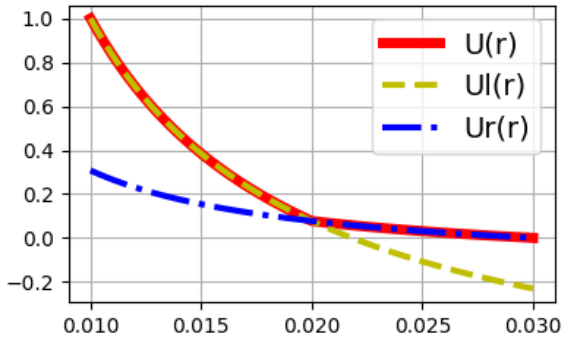
Бачимо, що отримані вирази $u_l(r)$ і $u_r(r)$ повністю співпадають з (32) при $u_0 = 1, u_2 = 0$.

По-аналогії з (27) розв'язок можна записати єдиним виразом

$$u(r) = \frac{u_0 + u_2}{2} - \frac{q}{2}\left[\frac{1}{\varepsilon_1}\left(\frac{1}{r_0} - \frac{1}{r}\right) + \frac{1}{\varepsilon_2}\left(\frac{1}{r_2} - \frac{1}{r}\right) + \left(\frac{1}{\varepsilon_2} - \frac{1}{\varepsilon_1}\right) \cdot \left|\frac{1}{r_1} - \frac{1}{r}\right|\right], \qquad (35)$$

де q обчислюється за формулою (34).

Для модельної задачі покладемо $r_0 = 0.01$, $r_1 = 0.02$, $r_2 = 0.03$, $\varepsilon_1 = 4$, $\varepsilon_2 = 16$, $u_0 = 1, u_2 = 0$. На наступному рисунку наведено графік неперервної кускової функції $u(r)$, обчислюваної за формулою (35), і окремо графіки виразів $u_l(r), u_r(r)$. Сподіваємося, що читач самостійно зможе реалізувати в Python відповідні побудови.



Література до глави.

1. Бернштейн С.Н. Об интерполировании. В кн. Собрание сочинений. Конструктивная теория функций. Т.1. – М.: АН СССР, 1952, – С.5 – 7.
2. Доля П.Г. Об одном способе представления кусочных полиномов в системах символьной математики. // Вестник Харьк. нац. ун-та., - 2008. - № 833. Сер. “Мате-матическое моделирование. Информационные технологии. Автоматизированные системы управления”, вып.10. – С.110-120.
3. Фокс Ф., Пратт М. Вычислительная геометрия. Применение в проектировании и на производстве. – М.: Мир, 1982.
4. Доля П.Г. Моделирование кусочно-гладких непрерывных функций и кривых. // Вестник Харьк. нац. ун-та., - 2005.- № 661. Сер. “Математическое моделирование. Информационные технологии. Автоматизированные системы управления”, вып.4. – С.97-103.
5. Рвачев В.Л. Геометрические приложения алгебры логики. – Киев: Техніка, 1967. – 212с.
6. Доля П.Г. Периодическое продолжение функций и решение уравнения колебаний струны в системах символьной математики. // Вестник Харьк. нац. ун-та., - 2006. - № 733. Сер. “Математическое моделирование. Информационные техно-логии. Автоматизированные системы управления”, вып.6. – С. 106-116.
7. Кошляков Н.С., Глинер Э.Б., Смирнов М.М. Основные дифференциальные уравнения математической физики. – М.: Гос. изд. физ.-мат. лит., 1962.
8. Тихонов А.Н., Самарский А.А. Уравнения математической физики. – М.: Наука, 1977. – 736с.
9. Тимошенко С.П. Сопротивление материалов. Т.1 – М.: Наука, 1965.
10. Будак Б.М., Самарский А.А., Тихонов А.Н. Сборник задач по математической физике: Учебное пособие. – 3-е изд. – М.: Наука, Главная редакция физико-математической литературы, 1980, 688 стр.
11. Dolya P. G. Solution to the homogeneous boundary value problems of free vibrations of a finite string. - Journal of Mathematical Physics, Analysis, Geometry, - 2008, vol.4, № 2, pp. 237 - 251