Міністерство освіти і науки України
Львівський національний університет імені Івана Франка
Факультет прикладної математики та інформатики
Кафедра програмування

Лабораторна робота
Функції Лаґерра
з курсу “Виробнича практика”

Виконав:
студент групи ПМІ-21
Дудинець Олександр Іванович

Львів – 2023

Мета роботи: здобути практичні навички використання многочленів Лаґерра, їхніх прямих та обернених перетворень.

Перетворення Лагера

[1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import ipywidgets as widgets
from typing import Callable
```

1.5.1 Многочлени Лаґерра

Функція, що обчислює многочлен Лаґерра порядку n, для заданого значення аргументу t та параметрів beta та sigma

[2]:
```
def laguerre_polymonials(
        t: float,
        n: int,
        beta: float = 2.0,
        sigma: float = 4.0
) -> float:
    """
    Функція для обчислення многочлену Лаґерра

    :param t:       Значення аргументу
    :param n:       Степінь многочлена Лаґерра
    :param beta:    Параметр бета
    :param sigma:   Параметр сигма

    :return:        Значення многочлена Лаґерра
    """

    # Валідація вхідних даних
    if beta < 0:
        raise ValueError('Value "beta" must be positive')

    if sigma < beta:
        raise ValueError('Value "sigma" must be greater than beta')

    if n < 0:
        raise ValueError('Value "n" must be positive')

    # Найкращі випадки
    l_prev_prev = np.sqrt(sigma) * np.exp(-beta * t / 2)
    l_prev = np.sqrt(sigma) * (1 - sigma * t) * np.exp(-beta * t / 2)
    if n == 0:
```

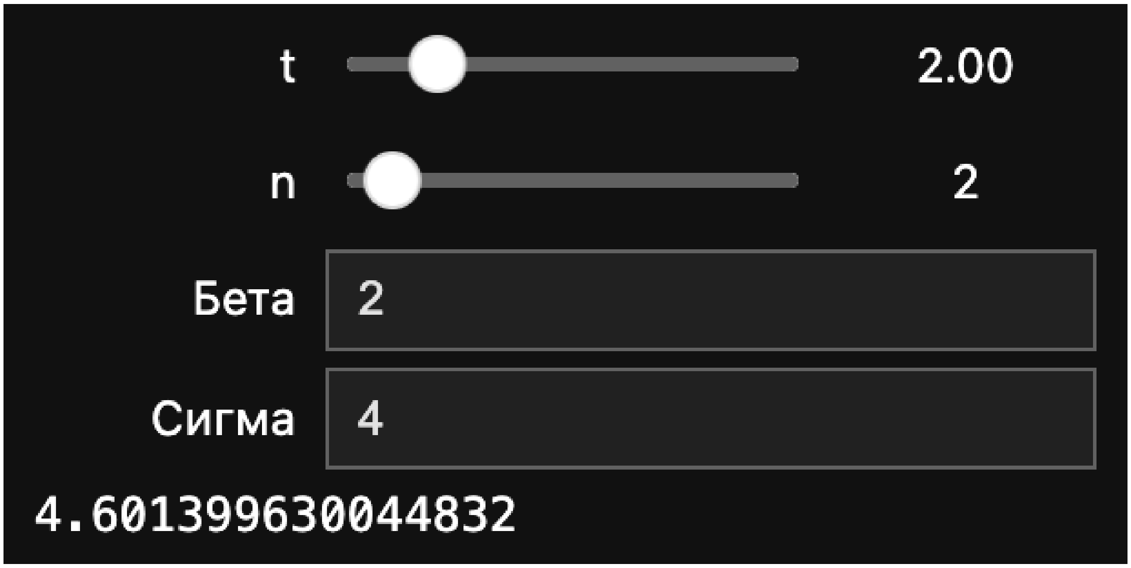
```python
        return l_prev_prev
    if n == 1:
        return l_prev

    # Обчислення
    for i in range(2, n + 1):
        temp = l_prev
        l_prev = (2 * i - 1 - sigma * t) * l_prev / i - (i - 1) * l_prev_prev / i
        l_prev_prev = temp

    return l_prev
```

[3]:

```python
widgets.interact(
    laguerre_polymonials,
    t=widgets.FloatSlider(
        value=2,
        min=0,
        max=10,
        step=0.1,
        description='t'
    ),
    n=widgets.IntSlider(
        value=2,
        min=0,
        max=20,
        step=1,
        description='n'
    ),
    beta=widgets.FloatText(
        value=2,
        min=0,
        max=10,
        step=0.1,
        description='Бета'
    ),
    sigma=widgets.FloatText(
        value=4,
        min=0,
        max=10,
        step=0.1,
        description='Сигма'
    )
)
```

```
interactive(children=(FloatSlider(value=2.0, description='t', max=10.0),␣
↪IntSlider(value=2, description='n', m...
```

[3]:
```
<function __main__.laguerre_polymonials(t: float, n: int, beta: float = 2.0,
sigma: float = 4.0) -> float>
```



1.5.2 Табуляція многочленів Лаґерра

Функція, що табулює многочлен Лаґерра порядку n від 0 до t_max

[4]:
```
def laguerre_polynomials_tabulation(
        n: int,
        t_max: float,
        t_step: float = 0.1,
        beta: float = 2.0,
        sigma: float = 4.0
) -> pd.DataFrame:
    """
    Функція для табуляції многочленів Лаґерра

    :param n:        Степінь многочлена Лаґерра
    :param t_max:    Максимальне значення аргументу
    :param t_step:   Крок аргументу
    :param beta:     Параметр бета
    :param sigma:    Параметр сигма

    :return:         DataFrame з табульованими значеннями
    """

    # Валідація вхідних даних
    if beta < 0:
        raise ValueError('Value "beta" must be positive')

    if sigma < beta:
        raise ValueError('Value "omega" must be greater than beta')

    if n < 0:
        raise ValueError('Value "n" must be positive')

    if t_max < 0:
        raise ValueError('Value "t_max" must be positive')

    if t_step < 0:
        raise ValueError('Value "t_step" must be positive')

    # Табуляція
    t = np.arange(0, t_max, t_step)
    return pd.DataFrame(
        data={
            't': t,
            f'L_{n}': [laguerre_polymonials(t=i, n=n, beta=beta, sigma=sigma)␣
 ↪for i in t]
        }
    ).set_index('t')
```

[5]:
```python
widgets.interact(
    laguerre_polynomials_tabulation,
    n=widgets.IntSlider(
        value=2,
        min=0,
        max=20,
        step=1,
        description='n'
    ),
    t_max=widgets.FloatSlider(
        value=5,
        min=0,
        max=10,
        step=0.1,
        description='Макс. t'
    ),
    t_step=widgets.FloatSlider(
        value=0.1,
        min=0.1,
        max=1,
        step=0.1,
        description='Крок t'
    ),
    beta=widgets.FloatText(
        value=2,
        min=0,
        max=10,
        step=0.1,
        description='Бета'
    ),
    sigma=widgets.FloatText(
        value=4,
        min=0,
        max=10,
        step=0.1,
        description='Сигма'
    )
)
```

```
interactive(children=(IntSlider(value=2, description='n', max=20),␣
↪FloatSlider(value=5.0, description='Макс. t...
```

[5]:
```
<function __main__.laguerre_polynomials_tabulation(n: int, t_max: float, t_step:
float = 0.1, beta: float = 2.0, sigma: float = 4.0) ->
pandas.core.frame.DataFrame>
```

n 2

Макс. t 5.00

Крок t 0.10

Бета 2

Сигма 4

t	L_2
0.0	2.000000
0.1	0.506709
0.2	-0.458489
0.3	-1.007513
0.4	-1.233389
0.5	-1.213061
0.6	-1.009813
0.7	-0.675356
0.8	-0.251624
0.9	0.227679
1.0	0.735759
1.1	1.251595
1.2	1.758974
1.3	2.245662
1.4	2.702703
1.5	3.123822
1.6	3.504924
1.7	3.843661
1.8	4.139084
1.9	4.391335
2.0	4.601400
2.1	4.770902

1.5.3 Обчислювальний експеримент

Функція, що визначає найменше значення аргументу t, при якому всі поліноми Лаґерра порядку від 0 до n_max — менші за epsilon

[6]:
```
def experiment(
        n_max: int = 20,
        epsilon: float = 1e-3,
        t_max: float = 100,
        t_points: int = 1000,
        beta: float = 2.0,
        sigma: float = 4.0
) -> tuple[float, pd.DataFrame]:
    """
    Функція для проведення обчислювального експерименту. Пошук такого t, що␣
 ↪|laguerre_polymonials(n, t)| < epsilon для усіх n Є [0, N]

    :param n_max:       Верхня межа степеня многочлена Лаґерра
    :param epsilon:     Точність
    :param t_max:       Максимальне значення аргументу
    :param t_points:    Кількість точок для від 0 до t_max
    :param beta:        Параметр бета
    :param sigma:       Параметр сигма

    :return:            Кортеж з t та DataFrame з табульованими значеннями
    """

    # Валідація вхідних даних
    if n_max < 0:
        raise ValueError('Value "N" must be positive')

    if epsilon < 0:
        raise ValueError('Value "epsilon" must be positive')

    if t_max < 0:
        raise ValueError('Value "t_max" must be positive')

    if t_points < 0:
        raise ValueError('Value "t_points" must be positive')

    # Пошук t
    T = np.linspace(0, t_max, t_points)
    N = range(0, n_max + 1)
    suitable_t = None
    for t in T:
        is_t_suitable = True
        for n in N:
```

```
            if abs(laguerre_polymonials(t=t, n=n, beta=beta, sigma=sigma)) >␣
↪epsilon:
                is_t_suitable = False
                break
        if is_t_suitable and suitable_t is None:
            suitable_t = t
            break

    # Табуляція
    return suitable_t, pd.DataFrame(
        data={
            'n': N,
            'L_n': [laguerre_polymonials(t=suitable_t, n=n, beta=beta,␣
↪sigma=sigma) for n in N]
        }
    ).set_index('n')
```

[7]:
```
t, df = experiment()
print(f't = {t}')
df
```

```
t = 79.07907907907908
```

[7]:

n	L_n
0	9.066138e-35
1	-2.858701e-32
2	4.478343e-30
3	-4.647081e-28
4	3.593209e-26
5	-2.208132e-24
6	1.123332e-22
7	-4.865604e-21
8	1.831625e-19
9	-6.087176e-18
10	1.808168e-16
11	-4.848845e-15
12	1.183547e-13
13	-2.647728e-12
14	5.460659e-11
15	-1.043487e-09
16	1.855654e-08
17	-3.082750e-07
18	4.800407e-06
19	-7.027805e-05
20	9.699021e-04

1.5.4 Обчислення значень інтегралів

Функція для обчислення прямого перетворення Лаґерра за допомогою апроксимації інтегралу методом прямокутників

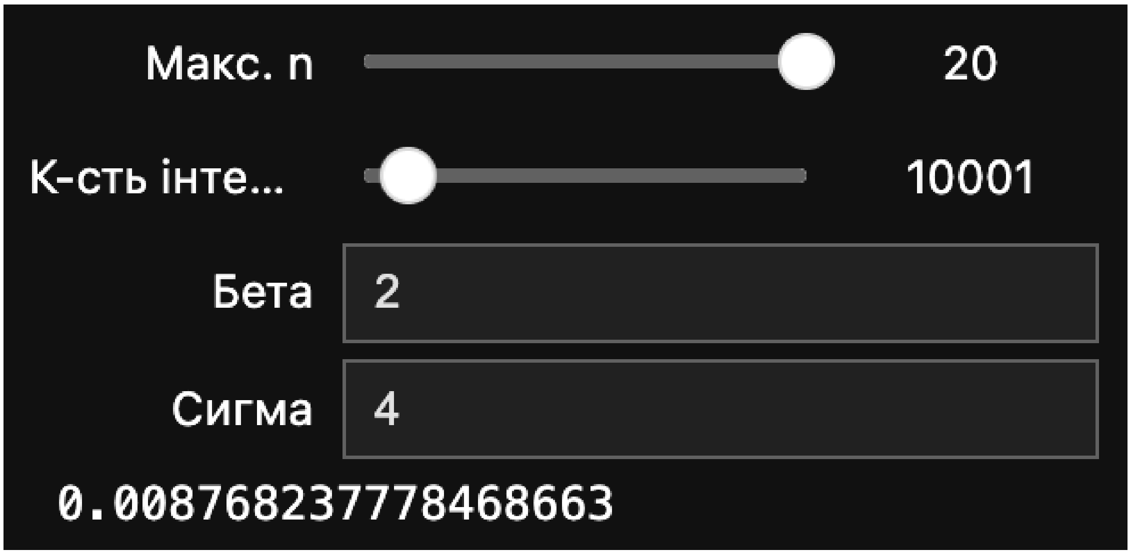
[8]:
```
def quad(
        f: Callable[[float], float],
        a: float,
        b: float,
        int_points: int = 10000
) -> float:
    """
    Функція для обчислення наближеного значення інтегралу методом прямокутників

    :param f:           Функція, яку інтегруємо
    :param a:           Початок інтервалу
    :param b:           Кінець інтервалу
    :param int_points:  Кількість точок для інтегрування

    :return: Значення інтегралу
    """

    x = np.linspace(a, b, int_points)
    s = sum([f(i) for i in x])
    return s * abs(b - a) / int_points


def laguerre_transform(
        f: Callable[[float], float],
        n_max: int,
        int_points: int = 10000,
        beta: float = 2.0,
        sigma: float = 4.0
) -> float:
    """
    Функція для обчислення перетворення Лаґерра

    :param f:           Функція, яку перетворюємо
    :param n_max:       Верхня межа степеня многочлена Лаґерра (N)
    :param int_points:  Кількість точок для інтегрування
    :param beta:        Параметр бета
    :param sigma:       Параметр сигма

    :return:            Значення перетворення Лаґерра
    """

    if n_max < 0:
        raise ValueError('Value "n_max" must be positive')
```

```
    if int_points < 0:
        raise ValueError('Value "t_step" must be positive')

    # Функція для інтегрування
    def integrand(t):
        alpha = sigma - beta
        return f(t) * laguerre_polymonials(t=t, n=n_max, beta=beta, sigma=sigma)␣
↪* np.exp(-alpha * t)

    # Верхня межа інтегрування
    t_max = experiment(n_max=n_max, beta=beta, sigma=sigma)[0]

    return quad(integrand, 0, t_max, int_points)


widgets.interact(
    laguerre_transform,
    f=widgets.fixed(lambda t: np.exp(-t ** 2)),
    n_max=widgets.IntSlider(
        value=20,
        min=0,
        max=20,
        step=1,
        description='Макс. n'
    ),
    int_points=widgets.IntSlider(
        value=10000,
        min=1,
        max=100000,
        step=1000,
        description='К-сть інтервалів розбиття для інтегрування'
    ),
    beta=widgets.FloatText(
        value=2,
        min=0,
        max=10,
        step=0.1,
        description='Бета'
    ),
    sigma=widgets.FloatText(
        value=4,
        min=0,
        max=10,
        step=0.1,
        description='Сигма'
    )
```

```
)
```

```
interactive(children=(IntSlider(value=20, description='Макс. n', max=20),␣
↪IntSlider(value=10000, description='...
```

[8]:
```
<function __main__.laguerre_transform(f: Callable[[float], float], n_max: int,
int_points: int = 10000, beta: float = 2.0, sigma: float = 4.0) -> float>
```



1.5.5 Перетворення Лаґерра

Функція для табулювання прямого перетворення Лаґерра порядку від 0 до n_max

[9]:
```
def laguerre_transform_tabulation(
        f: Callable[[float], float],
        n_max: int,
        int_points: int = 10000,
        beta: float = 2.0,
        sigma: float = 4.0
) -> pd.DataFrame:
    """
    Функція для табулювання перетворення Лаґерра

    :param f:           Функція, яку перетворюємо
    :param n_max:       Верхня межа степеня многочлена Лаґерра (N)
    :param int_points:  Кількість точок для інтегрування
    :param beta:        Параметр бета
    :param sigma:       Параметр сигма

    :return:            DataFrame з табульованими значеннями
    """

    # Валідація вхідних даних
    if n_max < 0:
        raise ValueError('Value "n_max" must be positive')

    if int_points < 0:
        raise ValueError('Value "t_step" must be positive')

    # Табулювання
    N = range(0, n_max)
    return pd.DataFrame(
        data={
            'n': N,
            f'L_n': [laguerre_transform(f=f, n_max=n, beta=beta,␣
 ↪int_points=int_points, sigma=sigma) for n in N]
        }
    ).set_index('n')
```

[10]:
```
# Визначення функції
def f(t):
    if t >= 2 * np.pi:
        return 0
    return np.sin(t - np.pi / 2) + 1


# Табулювання
```

```python
widgets.interact(
    laguerre_transform_tabulation,
    f=widgets.fixed(f),
    n_max=widgets.IntSlider(
        value=20,
        min=0,
        max=20,
        step=1,
        description='Макс. n'
    ),
    int_points=widgets.IntSlider(
        value=10000,
        min=1,
        max=100000,
        step=1000,
        description='К-сть інтервалів розбиття для інтегрування'
    ),
    beta=widgets.FloatText(
        value=2,
        min=0,
        max=10,
        step=0.1,
        description='Бета'
    ),
    sigma=widgets.FloatText(
        value=4,
        min=0,
        max=10,
        step=0.1,
        description='Сигма'
    )
)
```

```
interactive(children=(IntSlider(value=20, description='Макс. n', max=20),␣
↪IntSlider(value=10000, description='...
```

[10]:

```
<function __main__.laguerre_transform_tabulation(f: Callable[[float], float],
n_max: int, int_points: int = 10000, beta: float = 2.0, sigma: float = 4.0) ->
pandas.core.frame.DataFrame>
```

Макс. n 20

К-сть інте... 10001

Бета 2

Сигма 4

n	L_n
0	0.066660
1	-0.182204
2	0.178056
3	-0.074283
4	0.007263
5	0.007587
6	-0.003096
7	-0.000615
8	0.000799
9	-0.000026
10	-0.000236
11	0.000053
12	0.000094
13	-0.000031
14	-0.000053
15	0.000011
16	0.000036
17	0.000005
18	-0.000023
19	-0.000015

1.5.6 Обернене перетворення Лаґерра

Функція для обчислення оберненого перетворення Лаґерра. Приймає послідовність коефіцієнтів h, які можна отримати з табуляції перетворення Лаґерра

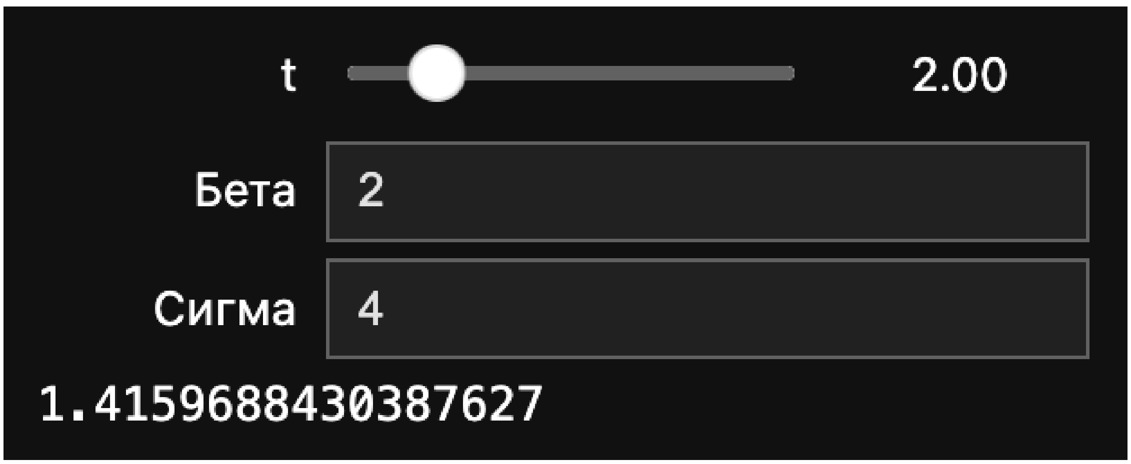
[11]:
```
def inverse_laguerre_transform(
        h: list[float],
        t: float,
        beta: float = 2.0,
        sigma: float = 4.0
) -> float:
    """
    Функція для обчислення оберненого перетворення Лаґерра

    :param h:       Довільна послідовність дійсних чисел
    :param t:       Значення аргументу
    :param beta:    Параметр бета
    :param sigma:   Параметр сигма

    :return:        Значення оберненого перетворення Лаґерра
    """

    # Обчислення
    return sum([h[k] * laguerre_polymonials(t=t, n=k, beta=beta, sigma=sigma)␣
 ↪for k in range(0, len(h))])
```

[12]:
```
# Отримуємо послідовність h з табуляції перетворення Лаґерра
h = laguerre_transform_tabulation(
    f=f,
    n_max=20,
    int_points=10000
)['L_n'].tolist()

widgets.interact(
    inverse_laguerre_transform,
    h=widgets.fixed(h),
    t=widgets.FloatSlider(
        value=2,
        min=0,
        max=10,
        step=0.1,
        description='t'
    ),
    beta=widgets.FloatText(
        value=2,
        min=0,
        max=10,
        step=0.1,
```

```
        description='Бета'
    ),
    sigma=widgets.FloatText(
        value=4,
        min=0,
        max=10,
        step=0.1,
        description='Сигма'
    )
)
```

```
interactive(children=(FloatSlider(value=2.0, description='t', max=10.0),␣
↪FloatText(value=2.0, description='Бет...
```

```
[12]: <function __main__.inverse_laguerre_transform(h: list[float], t: float, beta:
float = 2.0, sigma: float = 4.0) -> float>
```



1.5.7 Графік функції Лаґерра

Функція для побудови графіку многочленів Лаґерра порядку від 0 до n_max

[13]:
```
def laguerre_polynomials_plot(
        t_max: float,
        n_max: int,
        t_step: float = 0.01,
        beta: float = 2.0,
        sigma: float = 4.0
) -> None:
    """
    Функція для побудови графіку многочленів Лаґерра

    :param t_max:   Максимальне значення аргументу
    :param n_max:   Верхня межа степеня многочлена Лаґерра
    :param t_step:  Крок аргументу
    :param beta:    Параметр бета
    :param sigma:   Параметр сигма

    :return:        None
    """

    plt.figure(figsize=(12, 8))

    for n in range(0, n_max + 1):
        l_n_tabulation = laguerre_polynomials_tabulation(
            n=n,
            t_max=t_max,
            t_step=t_step
        )
        plt.plot(l_n_tabulation.index, l_n_tabulation[f'L_{n}'], label='$L_{' +␣
 ↪str(n) + '}$')

    plt.title('Многочлени Лагерра')
    plt.xlabel('t')
    plt.ylabel('$L_n(t)$')
    plt.grid()
    plt.legend(
        loc='upper left',
        bbox_to_anchor=(-0.2, 1)
    )

    plt.show()
```

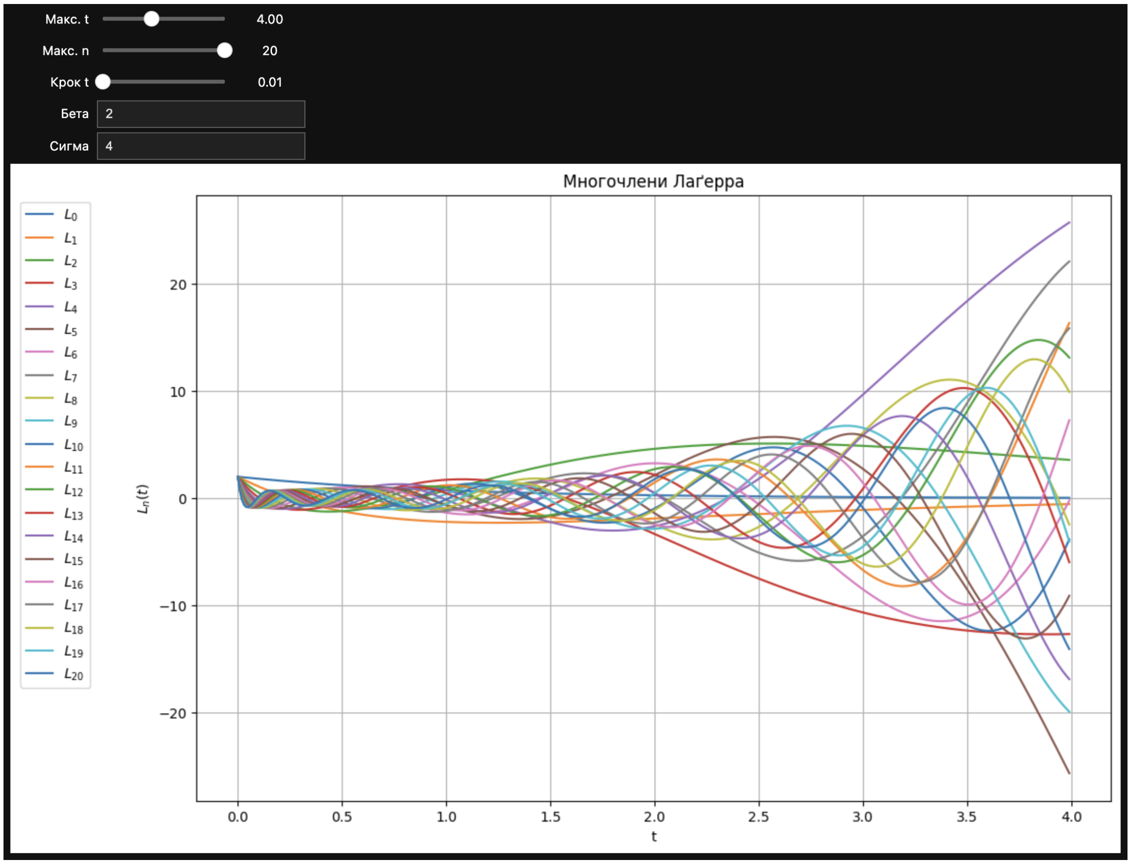
[14]:
```
widgets.interact(
    laguerre_polynomials_plot,
    n_max=widgets.IntSlider(
```

```
        value=20,
        min=0,
        max=20,
        step=1,
        description='Макс. n'
    ),
    t_max=widgets.FloatSlider(
        value=4,
        min=0,
        max=10,
        step=0.1,
        description='Макс. t'
    ),
    t_step=widgets.FloatSlider(
        value=0.01,
        min=0.01,
        max=1,
        step=0.01,
        description='Крок t'
    ),
    beta=widgets.FloatText(
        value=2,
        min=0,
        max=10,
        step=0.1,
        description='Бета'
    ),
    sigma=widgets.FloatText(
        value=4,
        min=0,
        max=10,
        step=0.1,
        description='Сигма'
    )
)
```

```
interactive(children=(FloatSlider(value=4.0, description='Макс. t', max=10.0),␣
↪IntSlider(value=20, description...
```

[14]:

```
<function __main__.laguerre_polynomials_plot(t_max: float, n_max: int, t_step:
float = 0.01, beta: float = 2.0, sigma: float = 4.0) -> None>
```


Макс. t
4.00
Макс. n
20
Крок t
0.01
Бета
2
Сигма
4
Многочлени Лаґерра
L_0 L_1 L_2 L_3 L_4 L_5 L_6 L_7 L_8 L_9 L_{10} L_{11} L_{12} L_{13} L_{14} L_{15} L_{16} L_{17} L_{18} L_{19} L_{20}
$L_n(t)$
t

1.5.8 Графік $\widetilde{f}^N(t), t \in [0, 2\pi]$

Функція для побудови графіків прямого, оберненого перетворення Лаґерра та початкової функції

[15]:
```
def laguerre_transform_plot(
        f: Callable[[float], float],
        n_max: int,
        t_max: float = np.pi * 2,
        t_step: float = 0.01,
        int_points: int = 10000,
        beta: float = 2.0,
        sigma: float = 4.0
) -> None:
    """
    Функція для побудови графіку перетворення Лаґерра

    :param f:           Функція, яку перетворюємо
    :param n_max:       Верхня межа степеня многочлена Лаґерра (N)
    :param t_max:       Максимальне значення аргументу
    :param t_step:      Крок аргументу
    :param int_points:  Кількість точок для інтегрування
    :param beta:        Параметр бета
    :param sigma:       Параметр сигма

    :return:            None
    """

    # Обчислення послідовності h
    laguerre_transform_tabulation_values = laguerre_transform_tabulation(
        f=f,
        n_max=n_max,
        int_points=int_points,
        beta=beta,
        sigma=sigma
    )
    h = laguerre_transform_tabulation_values['L_n'].tolist()

    # Табулювання
    T = np.arange(0, t_max, t_step)
    inverse_laguerre_transform_tabulation = pd.DataFrame(
        data={
            't': T,
            'h': [inverse_laguerre_transform(h=h, t=t, beta=beta, sigma=sigma)␣
↪for t in T]
        }
    ).set_index('t')
```

```python
# Побудова графіків
plt.figure(figsize=(12, 8))
plt.subplots_adjust(hspace=0.5)
plt.suptitle(f'Графік $\\widetilde{{f}}^N(t), t\\in[0, {t_max}]$')

# Перетворення Лаґерра
plt.subplot(2, 1, 1)
plt.plot(
    laguerre_transform_tabulation_values.index,
    laguerre_transform_tabulation_values['L_n'],
)
plt.title(f'Перетворення Лаґерра, N = {n_max}')
plt.xlabel('n')
plt.ylabel(r'$L_{N}$')
plt.grid()

# Обернене перетворення Лаґерра
plt.subplot(2, 1, 2)
initial_function_tabulation = pd.DataFrame(
    data={
        't': T,
        'f': [f(t) for t in T]
    }
).set_index('t')
plt.plot(
    initial_function_tabulation.index,
    initial_function_tabulation['f'],
    label='Початкова функція',
    linewidth=6,
    color='black',
    alpha=0.25
)

plt.plot(
    inverse_laguerre_transform_tabulation.index,
    inverse_laguerre_transform_tabulation['h'],
    label='Обернене перетворення Лаґерра',
    linewidth=1,
    color='green'
)

plt.title('Обернене перетворення Лаґерра')
plt.xlabel('t')
plt.ylabel(r'$\widetilde{f}^N(t)$')
plt.legend()
plt.grid()
```

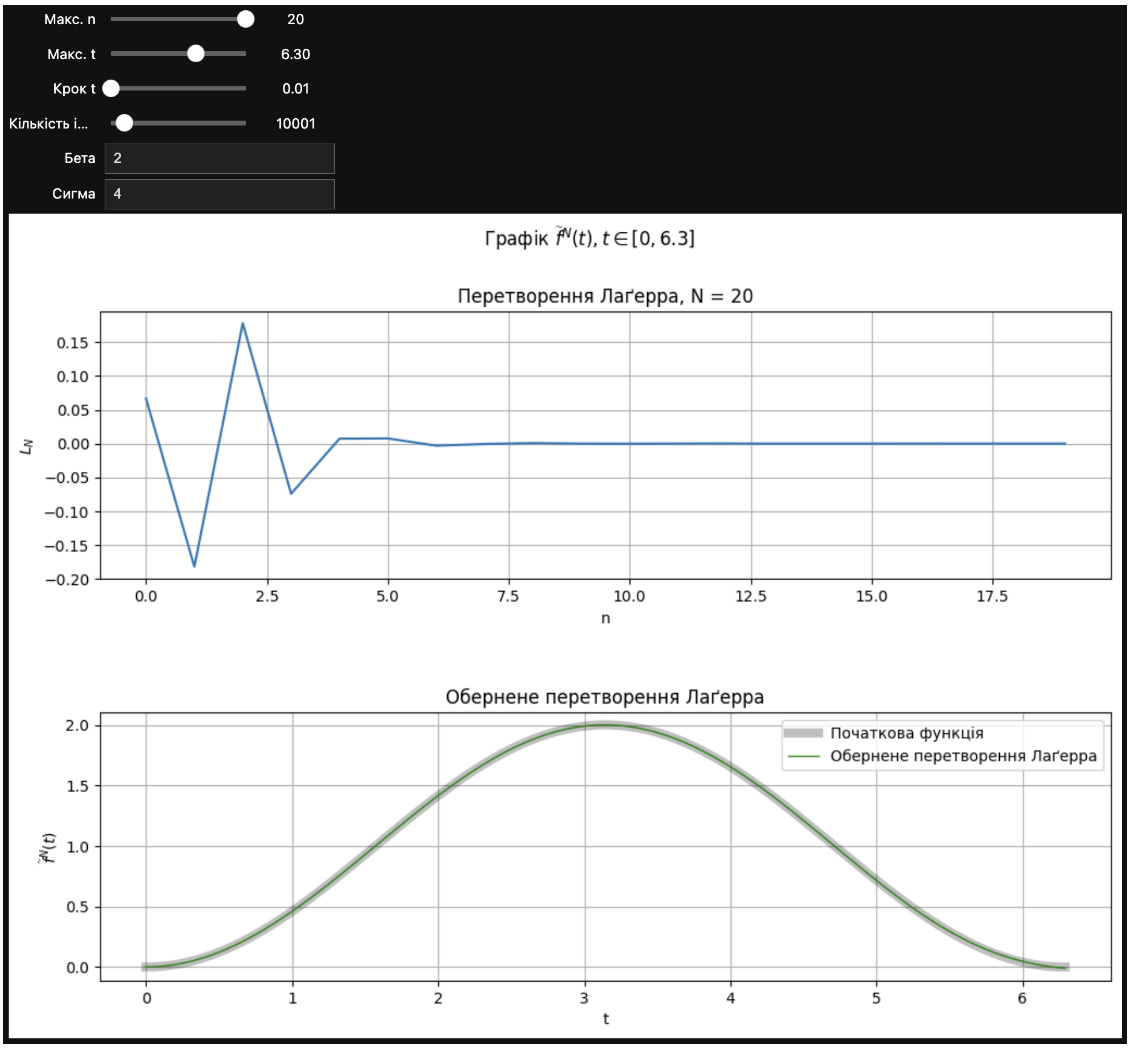
```
    plt.show()
```

[16]:
```
widgets.interact(
    laguerre_transform_plot,
    f=widgets.fixed(f),
    n_max=widgets.IntSlider(
        value=20,
        min=0,
        max=20,
        step=1,
        description='Макс. n'
    ),
    t_max=widgets.FloatSlider(
        value=2 * np.pi,
        min=0,
        max=10,
        step=0.1,
        description='Макс. t'
    ),
    t_step=widgets.FloatSlider(
        value=0.01,
        min=0.01,
        max=1,
        step=0.01,
        description='Крок t'
    ),
    int_points=widgets.IntSlider(
        value=10000,
        min=1,
        max=100000,
        step=1000,
        description='Кількість інтервалів розбиття для інтегрування'
    ),
    beta=widgets.FloatText(
        value=2,
        min=0,
        max=10,
        step=0.1,
        description='Бета'
    ),
    sigma=widgets.FloatText(
        value=4,
        min=0,
        max=10,
        step=0.1,
        description='Сигма'
    )
```

```
)
```

```
interactive(children=(IntSlider(value=20, description='Макс. n', max=20),␣
↪FloatSlider(value=6.283185307179586,...
```

[16]:
```
<function __main__.laguerre_transform_plot(f: Callable[[float], float], n_max:
int, t_max: float = 6.283185307179586, t_step: float = 0.01, int_points: int =
10000, beta: float = 2.0, sigma: float = 4.0) -> None>
```



Графік $\widetilde{g}^N(t), t \in [0,1]$

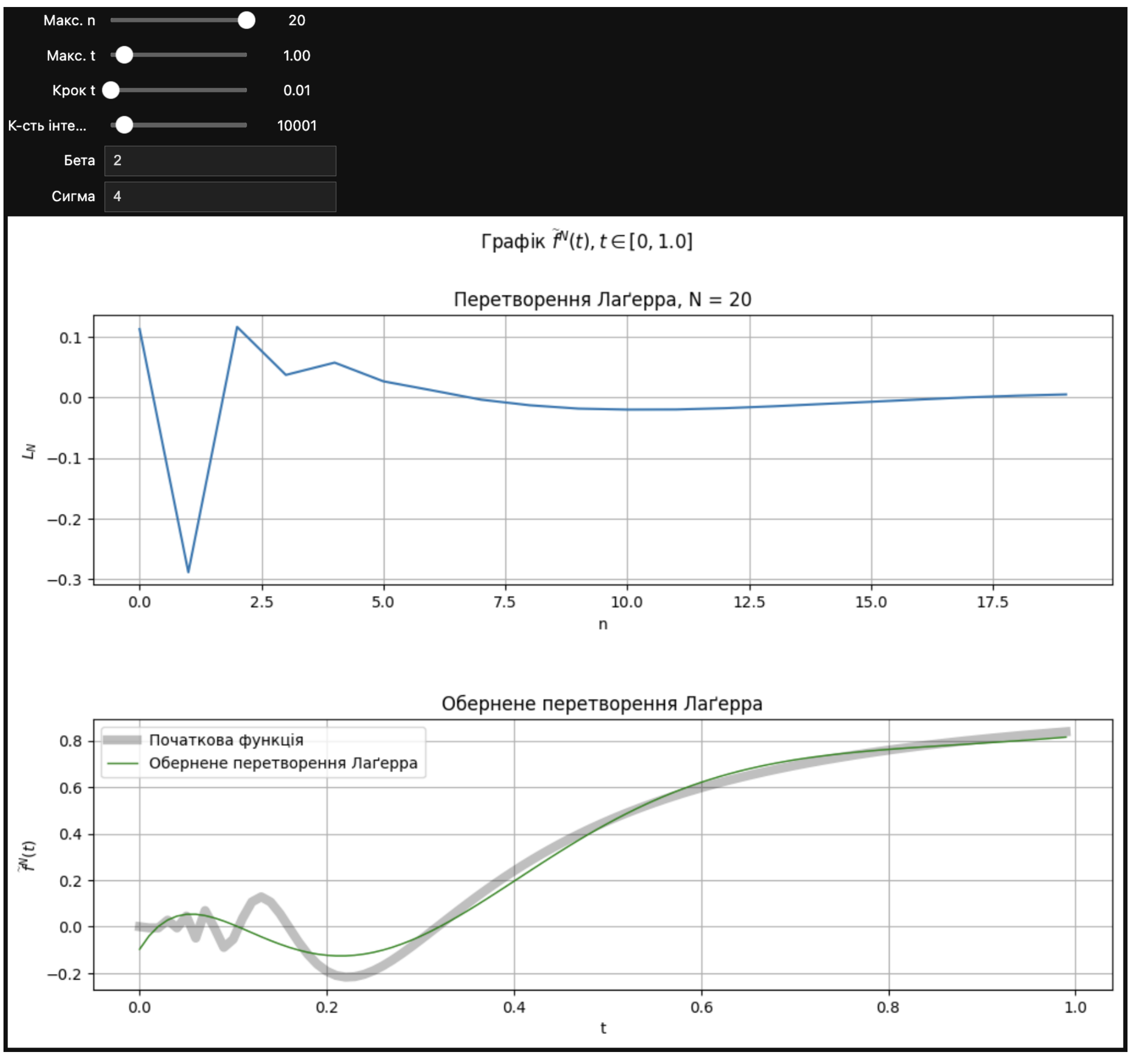
Побудова графіків прямого, оберненого перетворення Лаґерра для власної функції g

[17]:
```
def g(t):
    if t != 0:
        return t * np.sin(1 / t)
    return 0


widgets.interact(
    laguerre_transform_plot,
    f=widgets.fixed(g),
    n_max=widgets.IntSlider(
        value=20,
        min=0,
        max=20,
        step=1,
        description='Макс. n'
    ),
    t_max=widgets.FloatSlider(
        value=1,
        min=0,
        max=10,
        step=0.1,
        description='Макс. t'
    ),
    t_step=widgets.FloatSlider(
        value=0.01,
        min=0.01,
        max=1,
        step=0.01,
        description='Крок t'
    ),
    int_points=widgets.IntSlider(
        value=10000,
        min=1,
        max=100000,
        step=1000,
        description='К-сть інтервалів розбиття для інтегрування'
    ),
    beta=widgets.FloatText(
        value=2,
        min=0,
        max=10,
        step=0.1,
        description='Бета'
    ),
```

```
    sigma=widgets.FloatText(
        value=4,
        min=0,
        max=10,
        step=0.1,
        description='Сигма'
    )
)
```

```
interactive(children=(IntSlider(value=20, description='Макс. n', max=20),␣
↪FloatSlider(value=1.0, description='...
```

[17]:
```
<function __main__.laguerre_transform_plot(f: Callable[[float], float], n_max:
int, t_max: float = 6.283185307179586, t_step: float = 0.01, int_points: int =
10000, beta: float = 2.0, sigma: float = 4.0) -> None>
```



Висновок

У лабораторній роботі було розглянуто використання многочленів Лаґерра у контексті обчислення їхнього прямого та оберненого перетворення. Були реалізовані функції для обчислення многочленів, їхньої табуляції, проведення обчислювального експерименту для знаходження оптимального значення аргументу t та обчислення перетворень.

В результаті роботи було показано, як можна використовувати многочлени Лаґерра для перетворення функцій та як здійснювати обернене перетворення для відновлення початкових функцій. Проведено аналіз та вивчення залежностей між параметрами многочленів Лаґерра та їхніми властивостями.

Окремий акцент був зроблений на побудові графіків многочленів Лаґерра для різних значень степенів. Це дозволило візуально спостерігати їхні властивості та залежність від параметрів.

Також було проведено аналіз двох функцій, одна з яких була задана, а інша — власна. Для них було побудовано графіки перетворення Лаґерра та оберненого перетворення, що дозволило візуально спостерігати схожість графіку оберненого перетворення та початкової функції.

Отже, лабораторна робота надала можливість здобути практичні навички використання многочленів Лаґерра.