

```
In [48]: import numpy as np
import matplotlib.pyplot as plt
```

## 4.1. Визначення обсягу можливого фінансування інвестиційного проекту.

Відомо, що у фінансуванні проекту приймають участь п'ять фінансових установ, про які відомо таке:

1. Установа А надійна і стабільна, сума фінансування складе 300 у.о.
2. Установа Б надійна, але сума фінансування залежить від часу надання коштів. Так, з повною впевненістю можна стверджувати про те, що сума фінансування складе від 250 до 400 у.о., причому найбільш можливо одержати суму від 300 до 350 у.о.
3. Установа В планує фінансування проекту в сумі 200-300 у.о. із збільшенням впевненості в наданні суми з ростом обсягу виплати.
4. Установу Г визначають як нестабільну, але із можливістю надання значних сум коштів. Так, більша впевненість в тому, що проект буде профінансовано, зокрема в обсязі 2000-2400 у.о., але більш надійно одержати 2100-2200 у.о.
5. Установа Д ненадійна і нестабільна, проект напевне не фінансуватиме, але якщо надасть кошти, то в обсязі 300-500 у.о. із зменшенням упевненості в одержанні коштів із зростанням їх суми.

Визначити, найбільш можливу суму фінансування, неможливі обсяги фінансування, тощо.

```
In [51]: x = np.arange(0, 2601, 1, dtype=float)
```

### 1. Можливість отримання фінансування від будь-якої окремої установи

Задано п'ять фінансових установ із нечітким описом можливих сум фінансування. Нехай  $x$  — сума фінансування (у.о.). Для кожної установи  $i$  задаємо нечітку множину з функцією належності  $\mu_i(x)$ .

Використаємо форми функцій належності такі як трикутнаб трапецієподібнаб вироджений випадок - для точно заданого значення.

#### Установа А

Сума точно 300:

$$\mu_A(x) = \begin{cases} 1, & x = 300, \\ 0, & x \neq 300. \end{cases}$$

#### Установа Б

«250–400, найімовірніше 300–350» — трапецієподібна функція:

$$\tilde{B} = (250, 300, 350, 400).$$

#### Установа В

«200–300, впевненість зростає з ростом суми» — зростаюча обмежена функція:

$$\mu_V(x) = \begin{cases} 0, & x \leq 200, \\ \frac{x-200}{300-200}, & 200 < x < 300, \\ 0, & x \geq 300. \end{cases}$$

Поза інтервалом  $[200; 300]$  значення буде дорівнювати нулю, оскільки задано саме діапазон можливого фінансування.

#### Установа Г

«2000–2400, найнадійніше 2100–2200» — трапецієподібна функція:

$$\tilde{G} = (2000, 2100, 2200, 2400).$$

#### Установа Д

«Практично не фінансуватиме, але якщо видасть — 300–500, впевненість зменшується зі зростанням суми» — спадна обмежена функція:

$$\mu_D(x) = \begin{cases} 0, & x \leq 300, \\ \frac{500-x}{500-300}, & 300 < x < 500, \\ 0, & x \geq 500. \end{cases}$$

Оскільки розглядається можливість отримання фінансування **від будь-якої установи**, агрегування будемо виконувати через операцію об'єднання (OR):

$$\mu_{\text{total}}(x) = \max \{\mu_A(x), \mu_B(x), \mu_V(x), \mu_G(x), \mu_D(x)\}.$$

Функція  $\mu_{\text{total}}(x)$  є нечіткою оцінкою можливості отримання фінансування у розмірі  $x$ .

Найбільш можлива сума (інтервал):

$$x \in \arg \max_x \mu_{\text{total}}(x).$$

Неможливими є всі значення  $x$ , для яких:

$$\mu_{\text{total}}(x) = 0.$$

- обсяги, які не підтримуються жодною з фінансових установ.

Функції належності

```
In [55]: def mu_trapezoid(x, a, b, c, d):
    x = np.asarray(x, dtype=float)
    mu = np.zeros_like(x)

    m1 = (a < x) & (x < b)    # зростання
    if b != a:
        mu[m1] = (x[m1] - a) / (b - a)

    m2 = (b <= x) & (x <= c) # плато
    mu[m2] = 1.0

    m3 = (c < x) & (x < d)    # спадання
    if d != c:
        mu[m3] = (d - x[m3]) / (d - c)

    return np.clip(mu, 0.0, 1.0)

def mu_rising_linear(x, left, right):
    x = np.asarray(x, dtype=float)
    mu = np.zeros_like(x)

    m = (left < x) & (x < right)
    if right != left:
        mu[m] = (x[m] - left) / (right - left)

    mu[x >= right] = 1.0
    return np.clip(mu, 0.0, 1.0)

def mu_falling_linear(x, left, right):
    x = np.asarray(x, dtype=float)
    mu = np.zeros_like(x)

    mu[x <= left] = 1.0
    m = (left < x) & (x < right)
    if right != left:
        mu[m] = (right - x[m]) / (right - left)

    return np.clip(mu, 0.0, 1.0)

def mu_rising_segment(x, left, right):
    x = np.asarray(x, dtype=float)
    mu = np.zeros_like(x)

    m = (left < x) & (x < right)
    mu[m] = (x[m] - left) / (right - left) if right != left else 1.0

    mu[x == right] = 1.0
    return np.clip(mu, 0.0, 1.0)

def mu_falling_segment(x, left, right):
    x = np.asarray(x, dtype=float)
    mu = np.zeros_like(x)

    mu[x == left] = 1.0
    m = (left < x) & (x < right)
    mu[m] = (right - x[m]) / (right - left) if right != left else 0.0

    return np.clip(mu, 0.0, 1.0)
```

Установа А

```
In [57]: def mu_crisp_peak(x, x0=300.0, eps=0.5):
    return (np.abs(x - x0) <= eps).astype(float)

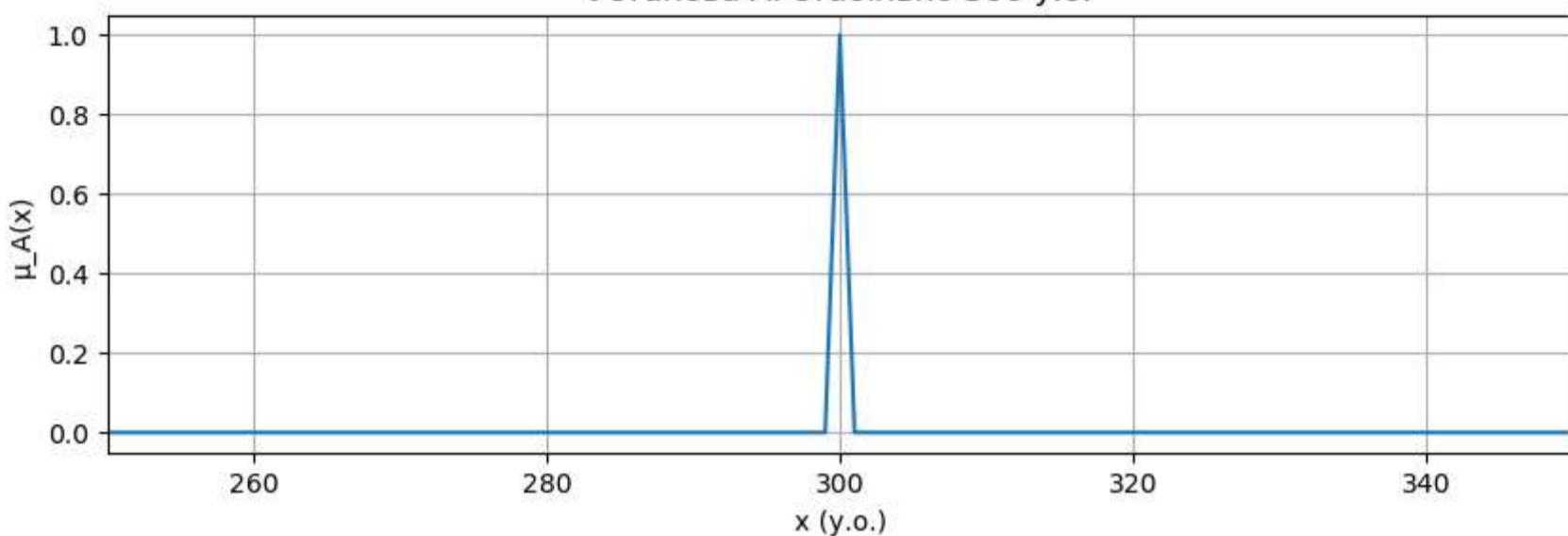
mu_A = mu_crisp_peak(x, 300.0, eps=0.5)
```

```

plt.figure(figsize=(10, 3))
plt.plot(x, mu_A)
plt.xlim(250, 350)
plt.ylim(-0.05, 1.05)
plt.grid(True)
plt.title("Установа А: стабільно 300 у.о.")
plt.xlabel("x (у.о.)")
plt.ylabel("μ_A(x)")
plt.show()

```

Установа А: стабільно 300 у.о.



Установа Б

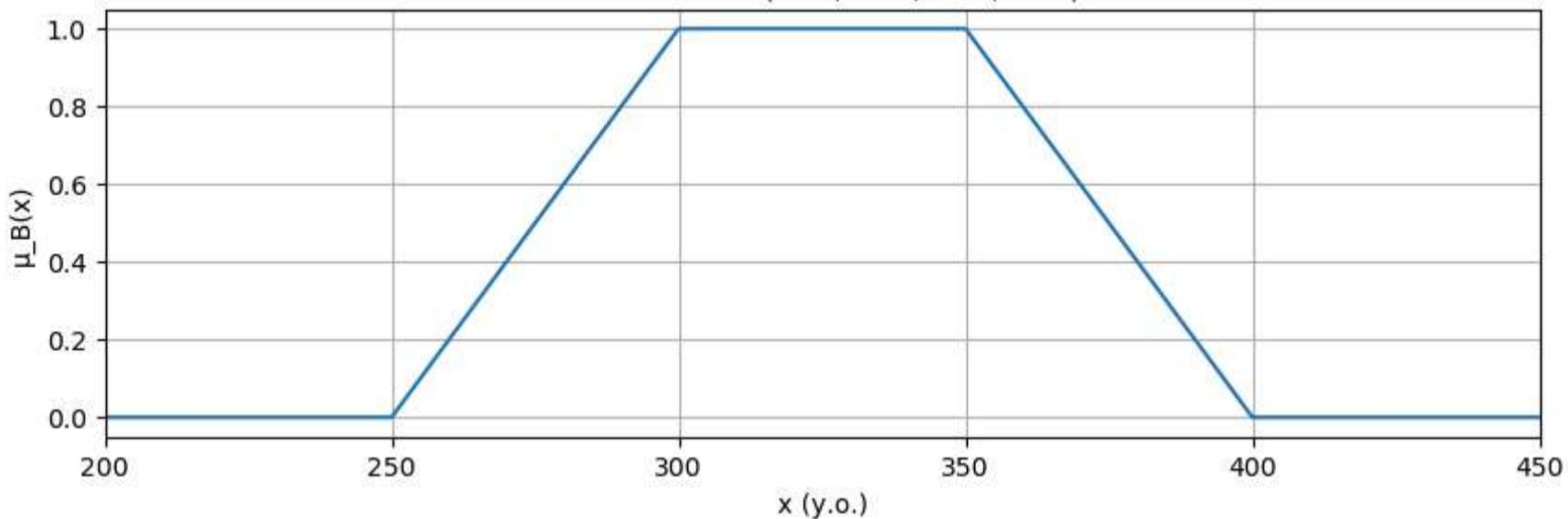
```

In [59]: mu_B = mu_trapezoid(x, 250, 300, 350, 400)

plt.figure(figsize=(10, 3))
plt.plot(x, mu_B)
plt.xlim(200, 450)
plt.ylim(-0.05, 1.05)
plt.grid(True)
plt.title("Установа Б: (250, 300, 350, 400)")
plt.xlabel("x (у.о.)")
plt.ylabel("μ_B(x)")
plt.show()

```

Установа Б: (250, 300, 350, 400)



Установа В

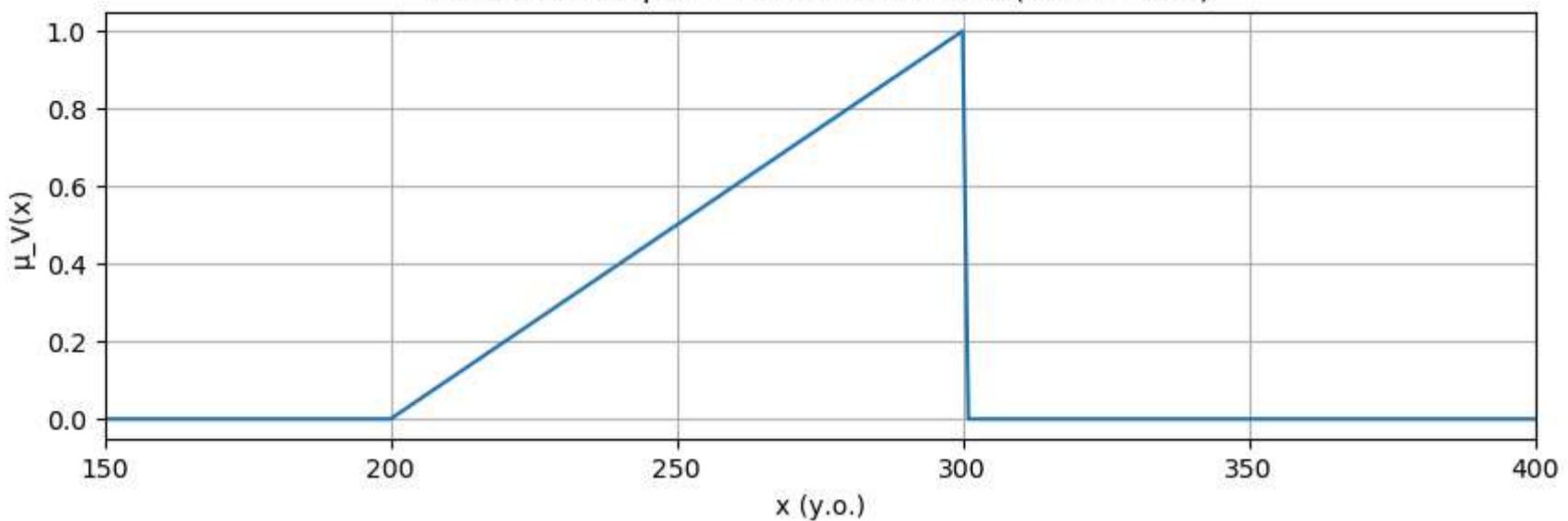
```

In [61]: mu_V = mu_rising_segment(x, 200, 300)

plt.figure(figsize=(10, 3))
plt.plot(x, mu_V)
plt.xlim(150, 400)
plt.ylim(-0.05, 1.05)
plt.grid(True)
plt.title("Установа В: зростаюча впевненість (200 -> 300)")
plt.xlabel("x (у.о.)")
plt.ylabel("μ_V(x)")
plt.show()

```

### Установа В: зростаюча впевненість (200 -> 300)

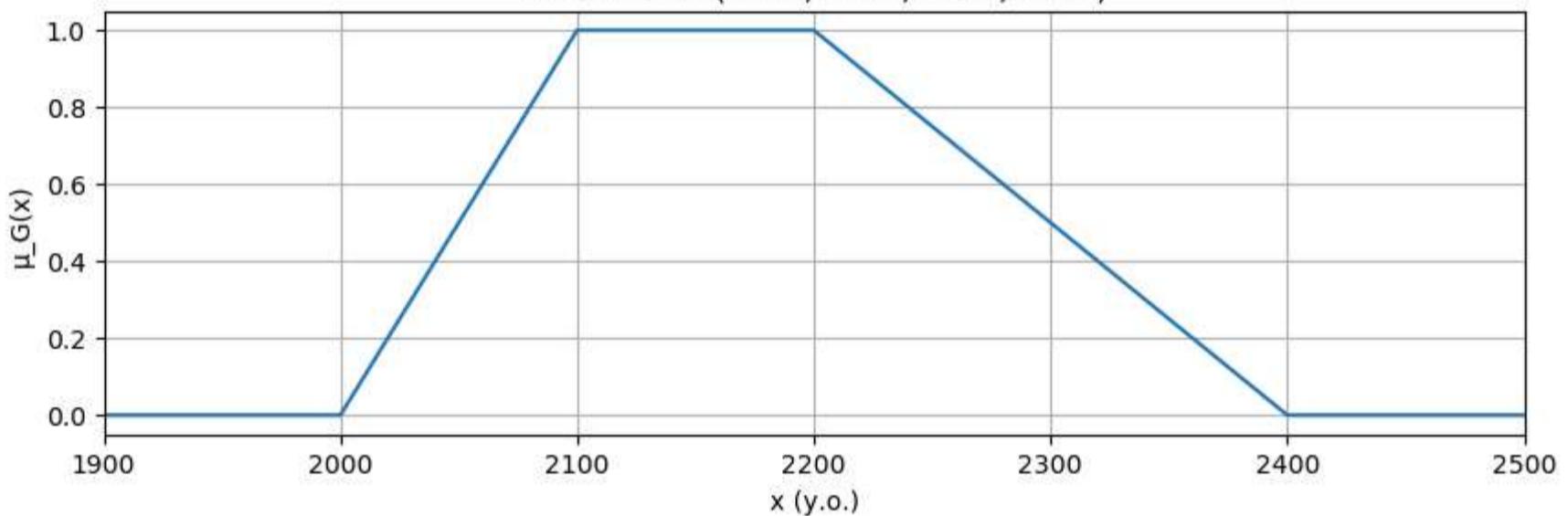


Установа Г

```
In [63]: mu_G = mu_trapezoid(x, 2000, 2100, 2200, 2400)

plt.figure(figsize=(10, 3))
plt.plot(x, mu_G)
plt.xlim(1900, 2500)
plt.ylim(-0.05, 1.05)
plt.grid(True)
plt.title("Установа Г: (2000, 2100, 2200, 2400)")
plt.xlabel("x (y.o.)")
plt.ylabel("μ_G(x)")
plt.show()
```

### Установа Г: (2000, 2100, 2200, 2400)

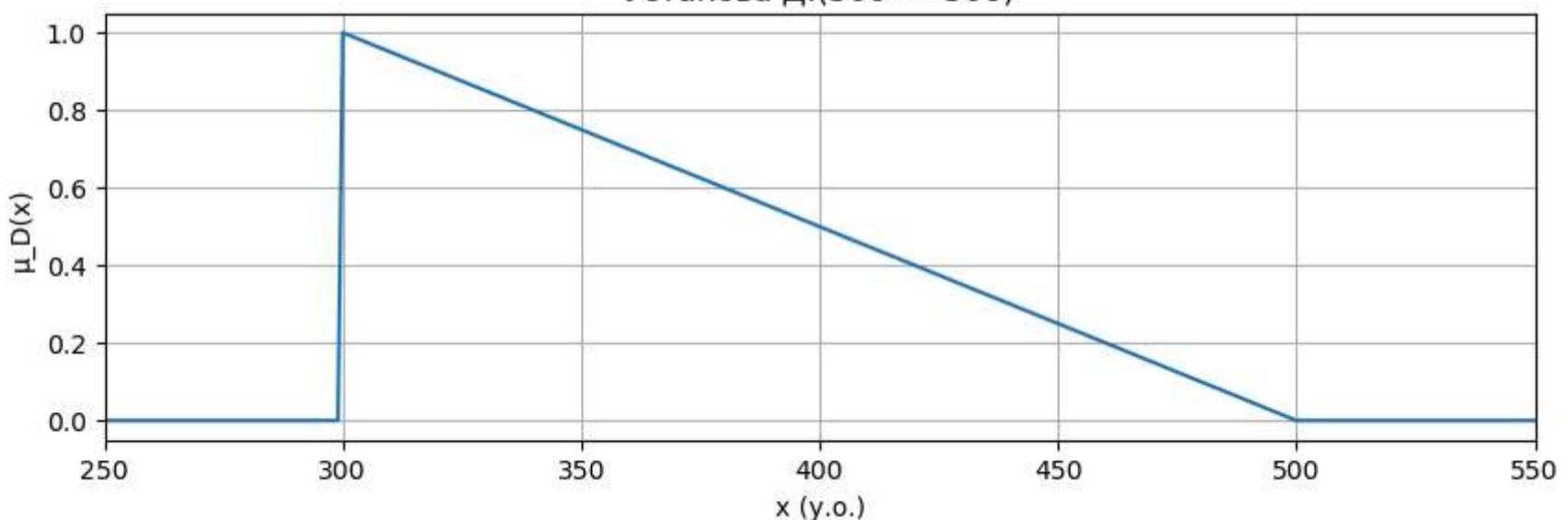


Установа Д

```
In [65]: mu_D = mu_falling_segment(x, 300, 500)

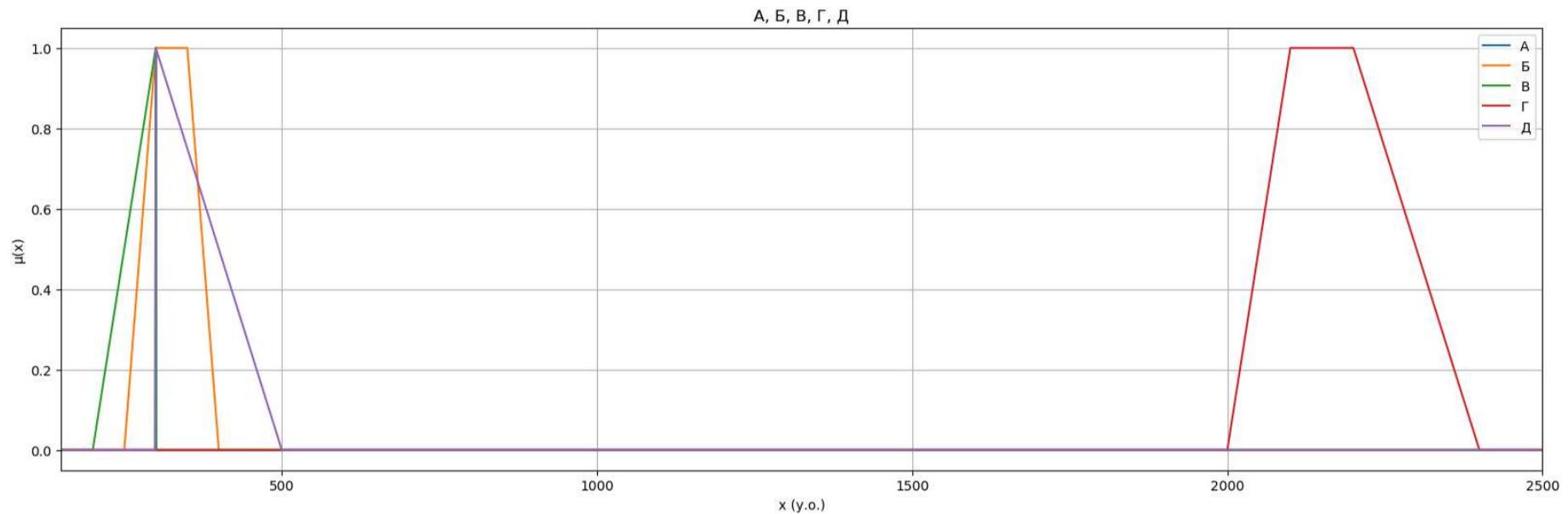
plt.figure(figsize=(10, 3))
plt.plot(x, mu_D)
plt.xlim(250, 550)
plt.ylim(-0.05, 1.05)
plt.grid(True)
plt.title("Установа Д: (300 -> 500)")
plt.xlabel("x (y.o.)")
plt.ylabel("μ_D(x)")
plt.show()
```

### Установа Д:(300 -> 500)



## Порівняння всіх функцій

```
In [67]: plt.figure(figsize=(20, 6))
plt.plot(x, mu_A, label="A")
plt.plot(x, mu_B, label="Б")
plt.plot(x, mu_V, label="В")
plt.plot(x, mu_G, label="Г")
plt.plot(x, mu_D, label="Д")
plt.xlim(150, 2500)
plt.ylim(-0.05, 1.05)
plt.grid(True)
plt.title("А, Б, В, Г, Д")
plt.xlabel("x (y.o.)")
plt.ylabel("μ(x)")
plt.legend()
plt.show()
```



Виконуємо агрегацію та знаходимо найбільш можливі значення та неможливі інтервали

```
In [69]: mu_total = np.maximum.reduce([mu_A, mu_B, mu_V, mu_G, mu_D])

tol = 1e-9

mu_max = mu_total.max()
mask_max = np.abs(mu_total - mu_max) <= tol
mask_zero = mu_total <= tol

def mask_to_intervals(x, mask):
    idx = np.where(mask)[0]
    if idx.size == 0:
        return []
    intervals = []
    start = idx[0]
    prev = idx[0]
    for i in idx[1:]:
        if i == prev + 1:
            prev = i
        else:
            intervals.append((x[start], x[prev]))
            start = i
            prev = i
    intervals.append((x[start], x[prev]))
    return intervals

most_possible = mask_to_intervals(x, mask_max)
impossible = mask_to_intervals(x, mask_zero)

print("μ_max =", mu_max)
print("Найбільш можливі значення (де μ = μ_max):", most_possible[:10], "... " if len(most_possible)>10 else "")
print("Неможливі інтервали (де μ = 0):", impossible[:10], "... " if len(impossible)>10 else "")
```

μ<sub>max</sub> = 1.0  
Найбільш можливі значення (де μ = μ<sub>max</sub>): [(300.0, 350.0), (2100.0, 2200.0)]  
Неможливі інтервали (де μ = 0): [(0.0, 200.0), (500.0, 2000.0), (2400.0, 2600.0)]

```
In [70]: plt.figure(figsize=(20, 5))

plt.plot(x, mu_A, label="A (300)")
plt.plot(x, mu_B, label="Б (250,300,350,400)")
plt.plot(x, mu_V, label="В (зростає 200→300)")
plt.plot(x, mu_G, label="Г (2000,2100,2200,2400)")
plt.plot(x, mu_D, label="Д (спадає 300→500)")

plt.plot(x, mu_total, linewidth=3, label="TOTAL = max(...)")

plt.xlim(0, 2600)
plt.ylim(-0.05, 1.05)
plt.grid(True)
```

```

plt.title("Усі функції належності та агрегована  $\mu_{\text{total}}(x)$ ")
plt.xlabel("x (сума фінансування, у.о.)")
plt.ylabel("μ(x)")
plt.legend()
plt.show()

```



Обчислюємо для заданої  $\mu(x)$ :

- множину елементів, для яких функція належності дорівнює одиниці:

$$\mu(x) = 1.$$

- множину всіх елементів, для яких функція належності додатна:

$$\mu(x) > 0.$$

- елементи, для яких функція належності дорівнює нулю:

$$\mu(x) = 0.$$

- елементи, для яких функція належності досягає максимального значення:

$$\mu(x) = \max \mu.$$

```

In [72]: tol = 1e-9

mask_core = np.abs(mu_total - 1.0) <= tol
mask_support = mu_total > tol
mask_zero = mu_total <= tol

mu_max = mu_total.max()
mask_max = np.abs(mu_total - mu_max) <= tol

core = mask_to_intervals(x, mask_core)
support = mask_to_intervals(x, mask_support)
impossible = mask_to_intervals(x, mask_zero)
most_possible = mask_to_intervals(x, mask_max)

print("μ_max =", mu_max)
print("Найбільш можливі значення ( $\mu = \mu_{\text{max}}$ ):", most_possible)
print("Ядро ( $\mu = 1$ ):", core)
print("Підтримка ( $\mu > 0$ ):", support)
print("Неможливі ( $\mu = 0$ ):", impossible[:10], "... " if len(impossible)>10 else "")

```

$\mu_{\text{max}} = 1.0$   
Найбільш можливі значення ( $\mu = \mu_{\text{max}}$ ): [(300.0, 350.0), (2100.0, 2200.0)]  
Ядро ( $\mu = 1$ ): [(300.0, 350.0), (2100.0, 2200.0)]  
Підтримка ( $\mu > 0$ ): [(201.0, 499.0), (2001.0, 2399.0)]  
Неможливі ( $\mu = 0$ ): [(0.0, 200.0), (500.0, 2000.0), (2400.0, 2600.0)]

Для детальнішого аналізу використаємо  $\alpha$ -зрізи:

$$X_\alpha = \{x : \mu_{\text{total}}(x) \geq \alpha\}.$$

Та виділимо області:

- «можливо» ( $\alpha = 0.3$ ),
- «ймовірно» ( $\alpha = 0.5$ ),
- «дуже ймовірно» ( $\alpha = 0.9$ ).

```

In [74]: def alpha_cut_intervals(x, mu, alpha):
    mask = mu >= alpha
    return mask_to_intervals(x, mask)

```

```

In [75]: alpha_levels = [0.3, 0.5, 0.9]
res = []

plt.figure(figsize=(20, 5))

```

```

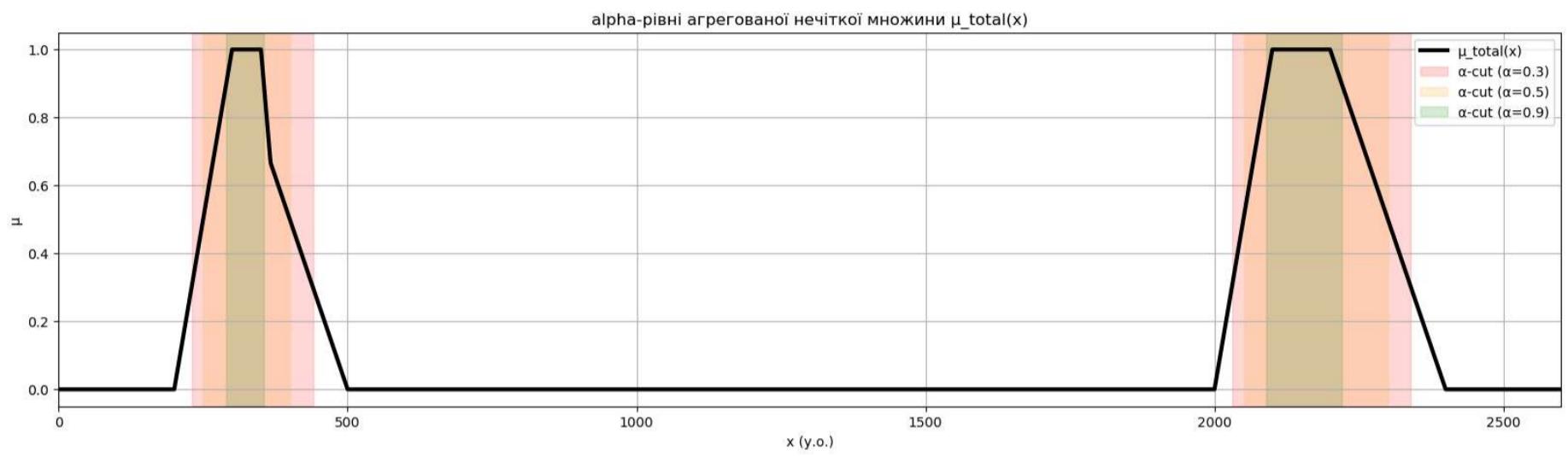
plt.plot(x, mu_total, linewidth=3, color="black", label="μ_total(x)")

colors = {0.3: "red", 0.5: "orange", 0.9: "green"}

for a in alpha_levels:
    intervals = alpha_cut_intervals(x, mu_total, a)
    first = True
    for l, r in intervals:
        res.append((a, l, r))
        plt.axvspan(l, r, alpha=0.15, color=colors[a],
                    label=f"α-cut (α={a})" if first else None)
        first = False

plt.xlim(0, 2600)
plt.ylim(-0.05, 1.05)
plt.grid(True)
plt.title("alpha-рівні агрегованої нечіткої множини μ_total(x)")
plt.xlabel("x (y.o.)")
plt.ylabel("μ")
plt.legend()
plt.show()

```



In [76]: `res`

Out[76]: `[(0.3, 230.0, 440.0),  
 (0.3, 2030.0, 2340.0),  
 (0.5, 250.0, 400.0),  
 (0.5, 2050.0, 2300.0),  
 (0.9, 290.0, 355.0),  
 (0.9, 2090.0, 2220.0)]`

## Висновки

За агрегованою нечіткою множиною  $\mu_{\text{total}}(x)$ , отриманою в результаті операції об'єднання (OR, оператор  $\max$ ), встановлено, що область можливих обсягів фінансування має дві розділені зони:

- «малу» зону — приблизно в інтервалі  $[200; 500]$  у.о.;
- «велику» зону — приблизно в інтервалі  $[2000; 2400]$  у.о.

Найбільш можливі обсяги фінансування, що відповідають  $\mu = 1$ , задаються інтервалами

$$[300; 350] \text{ та } [2100; 2200].$$

На рівні високої впевненості  $\alpha = 0.9$  отримано  $\alpha$ -зрізи

$$[290; 355] \text{ та } [2090; 2220],$$

які характеризують дуже ймовірні значення фінансування.

Неможливими є всі обсяги фінансування  $x$ , для яких

$$\mu_{\text{total}}(x) = 0.$$

Зокрема, у межах розглянутого діапазону до них належать інтервали

$$[0; 200), \quad (500; 2000), \quad (2400; 2600].$$

## 2. Сумарна можливість отримання фінансування.

У попередньому випадку аналізувалась можливість отримання фінансування від будь-якої окремої установи.

Доцільно розглянути також визначення сумарної суми фінансування, якщо кілька (або всі) фінансові установи можуть брати участь у фінансуванні проекту одночасно.

Нехай внесок кожної фінансової установи описується нечітким числом

$$\tilde{X}_i, \quad i \in \{A, B, V, G, D\},$$

з відповідною функцією належності  $\mu_i(x)$ .

Тоді сумарне фінансування інвестиційного проєкту визначається як нечітка сума:

$$\tilde{S} = \tilde{A} \oplus \tilde{B} \oplus \tilde{V} \oplus \tilde{G} \oplus \tilde{D},$$

де  $\oplus$  — операція додавання нечітких чисел.

Нечітке число у вигляді:

$$\tilde{M} = (m_{\text{low}}, m_{\text{high}}, \alpha, \beta, h),$$

де:

- $[m_{\text{low}}, m_{\text{high}}]$  — інтервал (ядро);
- $\alpha$  — ширина лівого крила;
- $\beta$  — ширина правого крила;
- $h \in (0, 1]$  — рівень довіри.

Установа А:

Сума точно 300 ю.о.:

$$\tilde{A} = (300, 300, 0, 0, 1).$$

Установа Б:

«250–400, найімовірніше 300–350»:

$$\tilde{B} = (300, 350, 50, 50, 1).$$

Установа В:

«200–300, впевненість зростає з ростом суми» — правобічний трикутник:

$$\tilde{V} = (300, 300, 100, 0, 1).$$

Установа Г:

«2000–2400, найнадійніше 2100–2200»:

$$\tilde{G} = (2100, 2200, 100, 200, 1).$$

Установа Д:

Оскільки установа Д є ненадійною та нестабільною, її участь у фінансуванні розглядається у вигляді двох альтернативних сценаріїв.

Сценарій  $D_0$ : фінансування не надається:

$$\tilde{D}_0 = (0, 0, 0, 0, 0.8).$$

Сценарій  $D_+$ : фінансування 300–500 зі спадною впевненістю:

$$\tilde{D}_+ = (300, 300, 0, 200, 0.2).$$

Спочатку обчислюється сума стабільних джерел:

$$\tilde{S}_{\text{base}} = \tilde{A} \oplus \tilde{B} \oplus \tilde{V} \oplus \tilde{G}.$$

Далі формуються два сценарії сумарного фінансування:

$$\tilde{S}_1 = \tilde{S}_{\text{base}} \oplus \tilde{D}_0, \quad \tilde{S}_2 = \tilde{S}_{\text{base}} \oplus \tilde{D}_+.$$

Оскільки можливий лише один із сценаріїв участі установи Д, загальну оцінку сумарного фінансування буде визначатися як:

$$\mu_S(x) = \max \{\mu_{S_1}(x), \mu_{S_2}(x)\}.$$

```
In [80]: def mu_trapezoid_pointwise(x: float, M: tuple[float, float, float, float, float]) -> float:
    m_low, m_high, alpha, beta, h = M

    # Ліве крило
    if alpha > 0 and (m_low - alpha) <= x < m_low:
        return h * (x - (m_low - alpha)) / alpha

    # Плато
    if m_low <= x <= m_high:
        return h

    # Праве крило
    if beta > 0 and m_high < x <= (m_high + beta):
        return h * ((m_high + beta) - x) / beta
```

```

    return 0.0

def mu_trapezoid(x: np.ndarray, M: tuple[float, float, float, float, float]) -> np.ndarray:
    x = np.asarray(x, dtype=float)
    y = np.zeros_like(x, dtype=float)

    m_low, m_high, alpha, beta, h = M

    # Ліве крило
    if alpha > 0:
        mask = ((m_low - alpha) <= x) & (x < m_low)
        y[mask] = h * (x[mask] - (m_low - alpha)) / alpha

    # Праве крило
    if beta > 0:
        mask = (m_high < x) & (x <= (m_high + beta))
        y[mask] = h * ((m_high + beta) - x[mask]) / beta

    return np.clip(y, 0.0, h)

def core_interval(M: tuple[float, float, float, float, float]) -> tuple[float, float, float]:
    m_low, m_high, alpha, beta, h = M
    return m_low, m_high, h

def support_interval(M: tuple[float, float, float, float, float]) -> tuple[float, float]:
    m_low, m_high, alpha, beta, h = M
    return (m_low - alpha), (m_high + beta)

def fuzzy_add(M1: tuple[float, float, float, float, float],
              M2: tuple[float, float, float, float, float]) -> tuple[float, float, float, float, float]:
    m1_l, m1_r, a1, b1, h1 = M1
    m2_l, m2_r, a2, b2, h2 = M2

    h = min(h1, h2)

    eps = 1e-12
    alpha = h * (a1 / max(h1, eps) + a2 / max(h2, eps))
    beta = h * (b1 / max(h1, eps) + b2 / max(h2, eps))

    m_low = m1_l + m2_l - a1 - a2 + alpha
    m_high = m1_r + m2_r + b1 + b2 - beta

    return (float(m_low), float(m_high), float(alpha), float(beta), float(h))

def find_support_intervals(x: np.ndarray, mu_vals: np.ndarray, tol: float = 1e-12) -> list[tuple[float, float]]:
    mask = mu_vals > tol
    idx = np.where(mask)[0]
    if idx.size == 0:
        return []

    intervals = []
    start = idx[0]
    prev = idx[0]
    for i in idx[1:]:
        if i == prev + 1:
            prev = i
        else:
            intervals.append((float(x[start]), float(x[prev])))
            start = i
            prev = i
    intervals.append((float(x[start]), float(x[prev])))
    return intervals

def complement_intervals(domain: tuple[float, float], intervals: list[tuple[float, float]]) -> list[tuple[float, float]]:
    left, right = domain
    if not intervals:
        return [(left, right)]

    intervals = sorted(intervals, key=lambda t: t[0])
    out = []

    cur = left
    for a, b in intervals:
        if a > cur:
            out.append((cur, a))
        cur = max(cur, b)
    out.append((cur, right))

    return out

```

```

    if cur < right:
        out.append((cur, right))

    return out

```

```

In [81]: def build_institutions():
    A = (300, 300, 0, 0, 1.0)
    B = (300, 350, 50, 50, 1.0)
    V = (300, 300, 100, 0, 1.0)
    G = (2100, 2200, 100, 200, 1.0)
    D0 = (0, 0, 0, 0, 0.8)
    Dp = (300, 300, 0, 200, 0.2)

    return A, B, V, G, D0, Dp

def run_case_sum(domain_in=(0, 2600), domain_sum=(2500, 4200), n_points=2500):
    A, B, V, G, D0, Dp = build_institutions()
    S_base = fuzzy_add(fuzzy_add(fuzzy_add(A, B), V), G)
    S1 = fuzzy_add(S_base, D0) # Д не фінансує (висока впевненість)
    S2 = fuzzy_add(S_base, Dp) # Д фінансує (низька впевненість)

    x_in = np.linspace(domain_in[0], domain_in[1], n_points)

    inputs = [
        ("A", A),
        ("Б", B),
        ("В", V),
        ("Г", G),
        ("Д0 (не фінансує)", D0),
        ("Д+ (фінансує)", Dp),
    ]

    fig, (ax1, ax2) = plt.subplots(2, 1, figsize=(15, 10))

    for name, M in inputs:
        y = mu_trapezoid(x_in, M)
        ax1.plot(x_in, y, linewidth=2, label=name)

    ax1.set_title("Вхідні функції належності (внески установ)")
    ax1.set_xlabel("Сума (у.о.)")
    ax1.set_ylabel("μ(x)")
    ax1.grid(True, alpha=0.3)
    ax1.legend()

    x_sum = np.linspace(domain_sum[0], domain_sum[1], n_points)
    y_s1 = mu_trapezoid(x_sum, S1)
    y_s2 = mu_trapezoid(x_sum, S2)
    y_total = np.maximum(y_s1, y_s2)

    ax2.plot(x_sum, y_s1, linewidth=2, label="S1 = S_base + D0")
    ax2.plot(x_sum, y_s2, linewidth=2, label="S2 = S_base + D+")
    ax2.plot(x_sum, y_total, linewidth=3, label="S = max(S1, S2)")

    ax2.fill_between(x_sum, y_total, alpha=0.12)

    ax2.set_title("Сумарне фінансування: два сценарії та об'єднання")
    ax2.set_xlabel("Сумарна сума (у.о.)")
    ax2.set_ylabel("μ(S)")
    ax2.grid(True, alpha=0.3)
    ax2.legend()

    plt.tight_layout()
    plt.show()

    s1_core_l, s1_core_r, s1_h = core_interval(S1)
    s2_core_l, s2_core_r, s2_h = core_interval(S2)

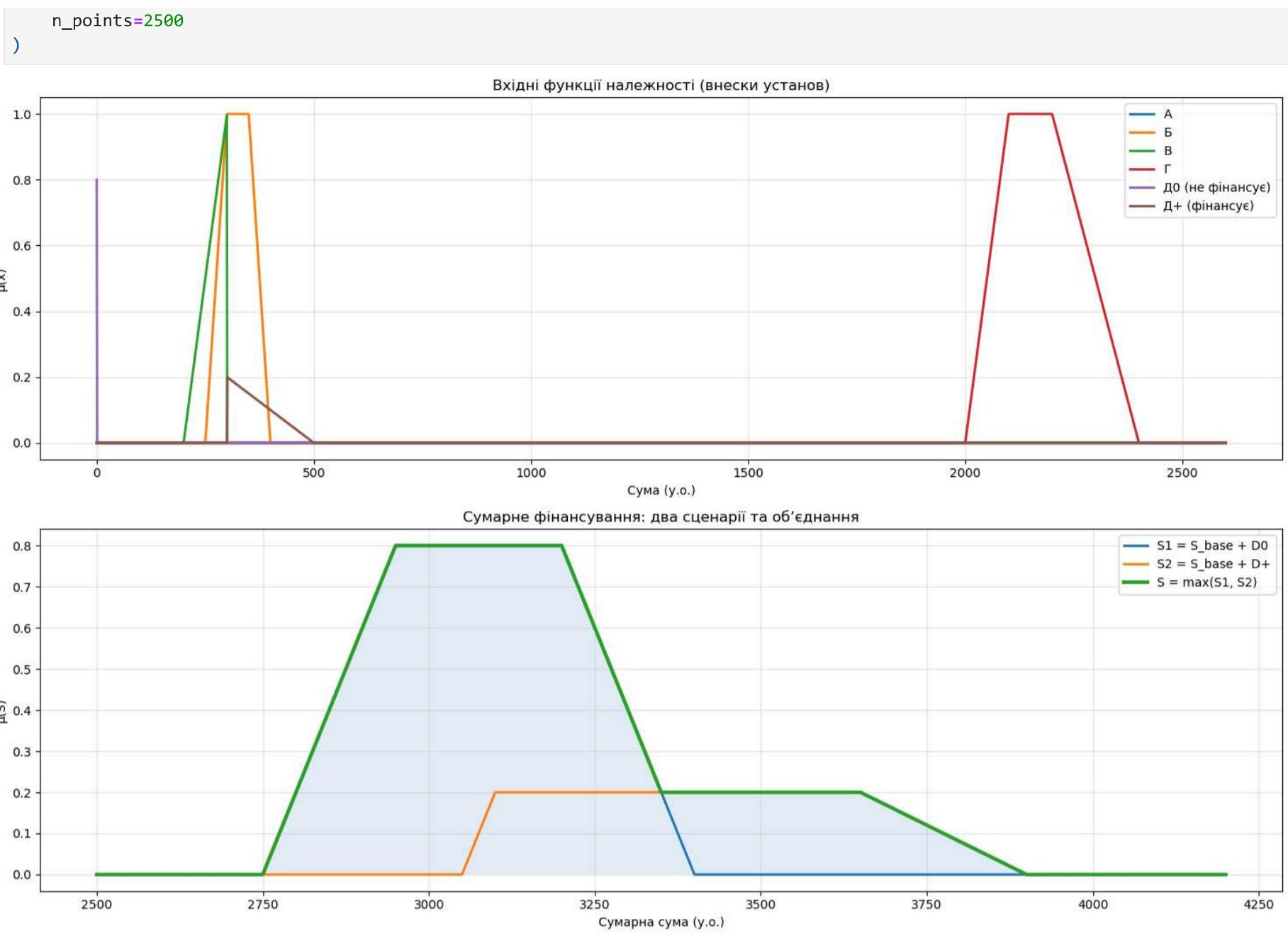
    s1_supp = support_interval(S1)
    s2_supp = support_interval(S2)

    print("==== Параметри сумарних нечітких чисел ===")
    print(f"S_base: core μ={S_base[4]:.2f} -> [{S_base[0]:.0f}; {S_base[1]:.0f}], support -> [{support_interval(S_base)[0]:.0f}; {support_interval(S_base)[1]:.0f}]")
    print(f"S1 = S_base + D0: core μ={s1_h:.2f} -> [{s1_core_l:.0f}; {s1_core_r:.0f}], support -> [{s1_supp[0]:.0f}; {s1_supp[1]:.0f}]")
    print(f"S2 = S_base + D+: core μ={s2_h:.2f} -> [{s2_core_l:.0f}; {s2_core_r:.0f}], support -> [{s2_supp[0]:.0f}; {s2_supp[1]:.0f}]")

    return {
        "A": A, "B": B, "V": V, "G": G, "D0": D0, "D+": Dp,
        "S_base": S_base, "S1": S1, "S2": S2
    }

run_case_sum(
    domain_in=(0, 2600),
    domain_sum=(2500, 4200),

```



==== Параметри сумарних нечітких чисел ===

S\_base: core  $\mu=1.00 \rightarrow [3000; 3150]$ , support  $\rightarrow [2750; 3400]$   
 S1 = S\_base + D0: core  $\mu=0.80 \rightarrow [2950; 3200]$ , support  $\rightarrow [2750; 3400]$   
 S2 = S\_base + D+: core  $\mu=0.20 \rightarrow [3100; 3650]$ , support  $\rightarrow [3050; 3900]$

```
Out[81]: {'A': (300, 300, 0, 0, 1.0),
           'B': (300, 350, 50, 50, 1.0),
           'V': (300, 300, 100, 0, 1.0),
           'G': (2100, 2200, 100, 200, 1.0),
           'D0': (0, 0, 0, 0, 0.8),
           'D+': (300, 300, 0, 200, 0.2),
           'S_base': (3000.0, 3150.0, 250.0, 250.0, 1.0),
           'S1': (2950.0, 3200.0, 200.0, 200.0, 0.8),
           'S2': (3100.0, 3650.0, 50.0, 250.0, 0.2)}
```

#### Висновки

Найбільш можливі значення для сценарію  $S_1$  відповідають інтервалу  $[2950; 3200]$  при рівні належності  $\mu = 0.80$ .

Найбільш можливі значення для сценарію  $S_2$  відповідають модальному інтервалу  $[3100; 3650]$  при рівні належності  $\mu = 0.20$ .

Оскільки  $\mu(S_2) \leq 0.2$  сценарій  $S_2$  формує лише слабо можливі значення та не впливає на області з високим рівнем упевненості.

Для агрегованої нечіткої множини  $S = \max(S_1, S_2)$  підтримка на заданому діапазоні визначається умовою  $\mu(S) > 0$  що відповідає інтервалу  $[2750; 3899]$ .

Неможливими на заданому діапазоні є всі значення, для яких  $\mu(S) = 0$ . Зокрема, це інтервали  $[2500; 2750]$  та  $[3899; 4200]$ .

Таким чином, у другій постановці задачі отримано нечітку оцінку сумарного фінансування інвестиційного проєкту, яка дозволяє враховувати одночасну участі кількох фінансових установ та визначити найбільш можливі та неможливі обсяги бюджету з урахуванням нечіткості вихідних даних.

## 4.2. Логічне виведення.

Нехай базу знань складають два правила:

$P_1$ : якщо  $x \in A_1$  і  $y \in B_1$ , то  $Z \in C_1$ ,

$P_2$ : якщо  $x \in A_2$  і  $y \in B_2$ , то  $Z \in C_2$ ,

де  $A_1, A_2, B_1, B_2, C_1, C_2$  — нечіткі множини з трапецієподібними функціями належності.

Функції належності задані у параметричному вигляді:

$$\mu_{A_1} = \langle 100, 200, 30, 40, 1 \rangle,$$

$$\begin{aligned}\mu_{A_2} &= \langle 200, 300, 20, 60, 1 \rangle, \\ \mu_{B_1} &= \langle 140, 240, 30, 40, 1 \rangle, \\ \mu_{B_2} &= \langle 240, 320, 50, 40, 1 \rangle, \\ \mu_{C_1} &= \langle 50, 100, 10, 30, 1 \rangle, \\ \mu_{C_2} &= \langle 100, 150, 20, 50, 1 \rangle.\end{aligned}$$

Необхідно знайти  $Z_0$ , якщо

$$x_0 = 220, \quad y_0 = 200.$$

Функція належності

```
In [86]: def mu_trap(x, m_low, m_high, alpha, beta, h=1.0):
    if alpha > 0 and (m_low - alpha) <= x < m_low:
        return h * (x - (m_low - alpha)) / alpha
    if m_low <= x <= m_high:
        return h
    if beta > 0 and m_high < x <= (m_high + beta):
        return h * ((m_high + beta) - x) / beta
    return 0.0

def mu_trap_vec(xs, M):
    return np.array([mu_trap(x, *M) for x in xs], dtype=float)
```

```
In [87]: A1 = (100, 200, 30, 40, 1)
A2 = (200, 300, 20, 60, 1)

B1 = (140, 240, 30, 40, 1)
B2 = (240, 320, 50, 40, 1)

C1 = (50, 100, 10, 30, 1)
C2 = (100, 150, 20, 50, 1)

x0 = 220
y0 = 200

muA1 = mu_trap(x0, *A1)
muA2 = mu_trap(x0, *A2)
muB1 = mu_trap(y0, *B1)
muB2 = mu_trap(y0, *B2)

w1 = min(muA1, muB1)
w2 = min(muA2, muB2)

print("Сили правил:")
print(f"w1 = min(μA1, μB1) = {w1:.3f}")
print(f"w2 = min(μA2, μB2) = {w2:.3f}")
```

Сили правил:

$$\begin{aligned}w1 &= \min(\mu_{A1}, \mu_{B1}) = 0.500 \\ w2 &= \min(\mu_{A2}, \mu_{B2}) = 0.200\end{aligned}$$

Операції Мамдані

```
In [89]: def rule_strength(x0, y0, A, B):
    return min(mu_trap(x0, *A), mu_trap(y0, *B))

def clip(mu_vals, w):
    return np.minimum(mu_vals, w)

def aggregate_max(*arrays):
    res = arrays[0].copy()
    for a in arrays[1:]:
        res = np.maximum(res, a)
    return res

def centroid_defuzzify(zs, mu_vals):
    area = np.trapz(mu_vals, zs)
    if area < 1e-12:
        return np.nan
    return np.trapz(mu_vals * zs, zs) / area
```

```
In [90]: zs = np.linspace(0, 250, 4000)

muC1 = mu_trap_vec(zs, C1)
muC2 = mu_trap_vec(zs, C2)

muZ1 = clip(muC1, w1)
```

```

muZ2 = clip(muC2, w2)

muZ0 = aggregate_max(muZ1, muZ2)

z_star = centroid_defuzzify(zs, muZ0)

```

```

In [91]: plt.figure(figsize=(20, 5))

plt.plot(zs, muC1, label="C1")
plt.plot(zs, muC2, label="C2")
plt.plot(zs, muZ1, linewidth=2, label=f"Z1 = clip(C1, w1={w1:.2f})")
plt.plot(zs, muZ2, linewidth=2, label=f"Z2 = clip(C2, w2={w2:.2f})")

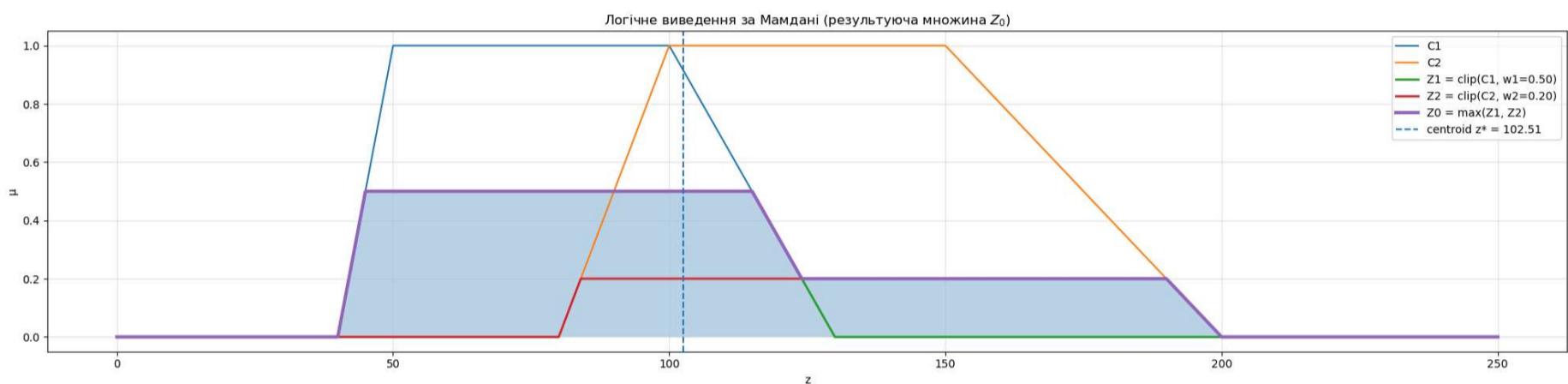
plt.plot(zs, muZ0, linewidth=3, label="Z0 = max(Z1, Z2)")
plt.fill_between(zs, muZ0, 0, alpha=0.3)

plt.axvline(z_star, linestyle="--", label=f"centroid z* = {z_star:.2f}")

plt.title("Логічне виведення за Мамдані (результатуюча множина Z0)")
plt.xlabel("z")
plt.ylabel("μ")
plt.ylim(-0.05, 1.05)
plt.grid(True, alpha=0.3)
plt.legend()

plt.tight_layout()
plt.show()

```



Висновки:

У результаті логічного виведення за Мамдані для заданих значень

$$x_0 = 220, \quad y_0 = 200$$

отримано результуючу нечітку множину

$$\tilde{Z}_0,$$

сформовану шляхом відрізання та агрегації вихідних множин відповідних правил (зафарбована область).