In [1]:
```
import numpy as np
import math
import matplotlib.pyplot as plt
import tkinter as tk
```

Зададим t исходя из таблицы критериев Стюдента

In [2]:
```
T=[4.3,3.18,2.77,2.57,2.44,2.36,2.3,2.26]
```

Форма ввода количества проектов

In [3]:
```
def win_k(title):
    window=tk.Tk()
    window.title(str(title))

    def Sub():
        window.destroy()

    b=tk.StringVar()
    label=tk.Label(text="Введите количество проектов")
    entry = tk.Entry(textvariable=b)
    entry.insert(tk.END,0)

    button = tk.Button(window,text="Далее",command=Sub)

    label.pack()
    entry.pack()
    button.pack()

    window.mainloop()
    bb=int(b.get())
    return bb
```

Функция построения оконного интрефейса, ввод первых параметров

In [4]:
```
def win1(title):
    window=tk.Tk()
    window.title(str(title))

    def Sub():
        window.destroy()

    a1=tk.StringVar()
    label=tk.Label(text="Введите количество экспертов")
    entry = tk.Entry(textvariable=a1)
    entry.insert(tk.END,0)

    a2=tk.StringVar()
    label2=tk.Label(text="Введите срок реализации")
    entry2=tk.Entry(textvariable=a2)
    entry2.insert(tk.END,0)

    a3=tk.StringVar()
    label3=tk.Label(text="Введите ставку дисконтирования в %")
    entry3=tk.Entry(textvariable=a3)
    entry3.insert(tk.END,0)

    a4=tk.StringVar()
    label4=tk.Label(text="Введите первоначальные вложения")
    entry4=tk.Entry(textvariable=a4)
    entry4.insert(tk.END,0)

    button = tk.Button(window,text="Далее",command=Sub)

    label.pack()
    entry.pack()
    label2.pack()
    entry2.pack()
    label3.pack()
    entry3.pack()
    label4.pack()
    entry4.pack()
    button.pack()

    window.mainloop()
    a1=int(a1.get())
    a2=int(a2.get())
    a3=int(a3.get())
    a4=int(a4.get())
    return a1,a2,a3,a4
```

Функция построения оконного интрефейса, ввод коэфициента риска

In [5]:
```
def win_risk():
    window=tk.Tk()
    window.title('Выберете коэффициент риска')

    def Sub():
        window.destroy()

    risk = tk.DoubleVar()
    risk.set(0.5)
    risk0 = tk.Radiobutton(text="0 -Абсолютное неприятие риска",var
    risk0_1 = tk.Radiobutton(text="0.1 -Очень сильное неприятие рис
    risk0_2 = tk.Radiobutton(text="0.2 -Сильное неприятие риска",va
    risk0_3 = tk.Radiobutton(text="0.3 -Среднее неприятие риска",va
    risk0_4 = tk.Radiobutton(text="0.4 -Слабое неприятие риска",var
    risk0_5 = tk.Radiobutton(text="0.5 -Нейтральное отношение к рис
    risk0_6 = tk.Radiobutton(text="0.6 -Слабое предпочтение риска",
    risk0_7 = tk.Radiobutton(text="0.7 -Среднее предпочтение риска"
    risk0_8 = tk.Radiobutton(text="0.8 -Сильное предпочтение риска"
    risk0_9 = tk.Radiobutton(text="0.9 -Очень сильное предпочтение
    risk1 = tk.Radiobutton(text="1 -Абсолютное предпочтение риска",
    button = tk.Button(text="Выберите",command=Sub)

    risk0.pack()
    risk0_1.pack()
    risk0_2.pack()
    risk0_3.pack()
    risk0_4.pack()
    risk0_5.pack()
    risk0_6.pack()
    risk0_7.pack()
    risk0_8.pack()
    risk0_9.pack()
    risk1.pack()
    button.pack()
    window.mainloop()
    return float(risk.get())
```

Функция оконного интерфейса, для ввода оценок экспертов по годам

In [6]:
```
def win2(k,y,title,title_label):
    window=tk.Tk()
    window.title(str(title))
    f_top=[]
    text=[]
    name=[]
    label=[]
    label2=[]
    top_top_top=tk.Frame(window)
    top_top_top.pack()
    top_top=tk.Frame(window)
```

```
    top_top.pack()
    kk=0
    kkk=0
    while kk<y:
        if title_label=='год':
            t_l=f"{kk+1} год"
        else:
            t_l='Ввод данных'
        label2.append(tk.Label(top_top_top,width = 8*3,text=t_l))
        label2[kk].pack(side=tk.LEFT)

        label.append(tk.Label(top_top,width = 8,text="Al"))
        label[kkk].pack(side=tk.LEFT)
        kkk+=1
        label.append(tk.Label(top_top,width = 8,text="A0"))
        label[kkk].pack(side=tk.LEFT)
        kkk+=1
        label.append(tk.Label(top_top,width = 8,text="Ar"))
        label[kkk].pack(side=tk.LEFT)
        kkk+=1
        kk+=1
    i=0
    sh=0
    while i<k:
        f_top.append(tk.Frame(window))
        f_top[i].pack()
        j=0
        while j<y:
            name.append(tk.StringVar())
            text.append(tk.Entry(f_top[i], width = 8, textvariable =
            text[sh].insert(tk.END,j)
            text[sh].pack(side=tk.LEFT)
            sh+=1

            name.append(tk.StringVar())
            text.append(tk.Entry(f_top[i], width = 8, textvariable =
            text[sh].insert(tk.END,j)
            text[sh].pack(side=tk.LEFT)
            sh+=1

            name.append(tk.StringVar())
            text.append(tk.Entry(f_top[i], width = 8, textvariable =
            text[sh].insert(tk.END,j)
            text[sh].pack(side=tk.LEFT)
            sh+=1
            j+=1
        i+=1
    def Sub():
        window.destroy()
    button = tk.Button(window,text="Завершить",command=Sub)
    button.pack()
    window.mainloop()
    mass=[]
    for n in name:
```

```
        mass.append(float(n.get()))
    i=0
    mass2d=[]
    while i<len(mass):
        prom=[]
        prom.append(mass[i])
        i+=1
        prom.append(mass[i])
        i+=1
        prom.append(mass[i])
        i+=1
        mass2d.append(prom)
    mass3d=[]
    i=0
    while i<y:
        prom=[]
        st=i
        j=0
        while j<k:
            prom.append(mass2d[st])
            st+=y
            j+=1
        i+=1
        mass3d.append(prom)
    return mass3d
```

Функция нахождения среднего

In [7]:
```
def M_x(matr,k):
    M_x=[]
    sl=0
    s0=0
    sr=0
    for m in matr:
        sl+=m[0]/k
        s0+=m[1]/k
        sr+=m[2]/k
    M_x.append(sl)
    M_x.append(s0)
    M_x.append(sr)
    return M_x
```

Функция возвращает ближайшее обычное множество

In [8]:
```
def Om(matr):
    Om=[]
    for m in matr:
        o=[]
        o.append(m[0]+0.5*(m[1]-m[0]))
        o.append(m[2]+0.5*(m[1]-m[2]))
        Om.append(o)
    return Om
```

2 функции поиска минимального и максимального числа из 2 значений

In [9]:
```
def _min(a,b):
    if a<b:
        return a
    elif a>b:
        return b
    else:
        return a
def _max(a,b):
    if a>b:
        return a
    elif a<b:
        return b
    else:
        return a
```

Функция возвращает матрицу парного сходства

In [10]:
```
def MP(Om,k):
    MP=[]
    i=0
    j=0
    while i<k:
        j=0
        mass=[]
        while j<k:
            mass.append(round((_min(Om[i][1],Om[j][1])-_max(Om[i][0
            j+=1
        MP.append(mass)
        i+=1
    return MP
```

Функция возвращает аддитивный показатель общей согласованности

In [11]:
```
def APOS(Om):
    mass_a=[]
    mass_b=[]
    for om in Om:
        mass_a.append(om[0])
        mass_b.append(om[1])
    np_mass_a=np.array(mass_a)
    np_mass_b=np.array(mass_b)
    APOS=round((np.min(np_mass_b)-np.max(np_mass_a))/(np.max(np_mas
    return APOS
```

Функция вовращает сумму столбцов матрицы парного сходства

In [12]:
```
def sumMP(MP):
    np_MP=np.array(MP)
    sumMP=np.sum(np_MP,axis=0)
    return sumMP
```

S1 и S2

In [13]:
```
def S1_S2(matr,M_x):
    prom=[]
    for m in matr:
        prom.append((m[0]-M_x[0])**2)
    sum_prom=0
    for p in prom:
        sum_prom+=p
    S1=math.sqrt(1/3*sum_prom)

    prom=[]
    for m in matr:
        prom.append((m[2]-M_x[2])**2)
    sum_prom=0
    for p in prom:
        sum_prom+=p
    S2=math.sqrt(1/3*sum_prom)
    return S1,S2
```

F_up и F_down

In [14]:
```
def F_up_down(M_x,S1,S2,t):
    F_up=[]
    F_down=[]
    F_up.append(M_x[0]-(S1*t/2))
    F_up.append(M_x[1])
    F_up.append(M_x[2]+(S2*t/2))
    F_down.append(M_x[0]+(S1*t/2))
    F_down.append(M_x[1])
    F_down.append(M_x[2]-(S2*t/2))
    return F_up,F_down
```

Построение графика экспертных оценок

In [15]:
```
def draw(matr):
    fig, ax = plt.subplots()
    for m in matr:
        plt.plot([m[0],m[1],m[2]],[0,1,0], marker = 'o')
    fig.set_figwidth(12)
    fig.set_figheight(6)
    plt.xlabel("X")
    plt.ylabel("Y")
    plt.show()
```

Построение графика обобщенного экспертного критерия 2 типа

In [16]:
```
def drawF(F_up,F_down):
    fig, ax = plt.subplots()
    plt.plot([F_up[0],F_up[1],F_up[2]],[0,1,0], marker = 'o')
    plt.plot([F_down[0],F_down[1],F_down[2]],[0,1,0], marker = 'o')
    fig.set_figwidth(12)
    fig.set_figheight(6)
    plt.xlabel("X")
    plt.ylabel("Y")
    plt.show()
```

Построение графика обобщенного экспертного критерия 1 типа

In [17]:
```
def drawF1(F):
    fig, ax = plt.subplots()
    plt.plot([F[0],F[1],F[2]],[0,1,0], marker = 'o')
    fig.set_figwidth(12)
    fig.set_figheight(6)
    plt.xlabel("X")
    plt.ylabel("Y")
    plt.show()
```

Коэфициент дисконтирования

In [18]:
```
def coof_discont(r,year):
    R=[]
    i=1
    r=r+1
    while i<=year:
        R.append(r**i)
        i+=1
    return R
```

Матрица Cf 2 типа

In [19]:
```
def Cf(R,F_up,F_down):
    Cf=[]
    Cf_up=[]
    Cf_down=[]
    for f in F_up:
        Cf_up.append(f/R)
    for f in F_down:
        Cf_down.append(f/R)
    Cf.append(Cf_up)
    Cf.append(Cf_down)
    return Cf
```

Матрица Cf_sum 2 типа

In [20]:
```
def Cf_sum(Cf):
    a_up=0
    b_up=0
    c_up=0
    a_down=0
    b_down=0
    c_down=0
    for cf in Cf:
        a_up+=cf[0][0]
        b_up+=cf[0][1]
        c_up+=cf[0][2]
        a_down+=cf[1][0]
        b_down+=cf[1][1]
        c_down+=cf[1][2]
    Cf_sum=[[a_up,b_up,c_up],[a_down,b_down,c_down]]
    return Cf_sum
```

Расчет NPV 2 типа

In [21]:
```
def NPV(Cf_sum,C):
    f=[]
    for cf in Cf_sum:
        for c in cf:
            f.append(c-C)
    f1=[]
    f2=[]
    i=0
    while i<6:
        if i<3:
            f1.append(f[i])
        else:
            f2.append(f[i])
        i+=1
    ff=[]
    ff.append(f1)
    ff.append(f2)
    return ff
```

Матрица Cf 1 типа

In [22]:
```
def Cf_1(R,M_x):
    Cf=[]
    for mx in M_x:
        Cf.append(mx/R)
    return Cf
```

Матрица Cf_sum 1 типа

In [23]:
```
def Cf_sum_1(Cf):
    a=0
    b=0
    c=0
    for cf in Cf:
        a+=cf[0]
        b+=cf[1]
        c+=cf[2]
    Cf_sum=[a,b,c]
    return Cf_sum
```

Нахождение NPV 1 типа

In [24]:
```
def NPV_1(NPV,risk):
    f=[]
    f.append(risk*NPV[0][0]+(1-risk)*NPV[1][0])
    f.append(NPV[0][1])
    f.append(risk*NPV[0][2]+(1-risk)*NPV[1][2])
    return f
```

Код, который вызывает функции описанные выше

In [25]:
```
def procedura(name_prog):
    k,year,r,C=win1(f'Ввод первичных параметров {name_prog}-го прое
                    #количество лет, ставка дисконтирования и пер
    risk=win_risk()
    r=r/100          #Так как ввод ставки дисконтирования в %, раз
    matr_y=win2(k,year,f'Ввод экспертных оценок {name_prog}-го прое
                     #количество экспертов и количество лет.
                     #Функция возвращает 3-х мерный массив введен
    R_=coof_discont(r,year)#Получаем коэф. дисконтирования, передав
                         #количество лет. Функция возращает масси
    F_up_akkum=[]    #Объявляем аккумулирующие массивы, которые бу
    F_down_akkum=[]
    Cf_akkum=[]
    Cf_1_akkum=[]
    K=k
    matr=[]
    M_x_=[]
    Om_=[]
    MP_=[]
    APOS_=0
    sumMP_=[]
    i=0
    def pod_funk():
        nonlocal matr,M_x_,k,Om_,MP_,APOS_,sumMP_,year,i
        print(f'Количество экспертов={k}')
        print()
        if k<=2:
```

```
            print(f'Повторный ввод экспертных оценок за {i+1} год')
            matr_yy=win2(K,1,f'Повторный ввод экспертных оценок за 
            matr=matr_yy[0]
            k=K
        for m in matr:              #выводим данные экспертных оценок 
            print(m)
        print()

        draw(matr)                  #выводим график экспертных оценок
        print()

        M_x_=M_x(matr,k)            #находим матрицу средних значений 
        print(f'M(x)={M_x_}')       #выводим матрицу средних значений
        print()

        Om_=Om(matr)                #находим ближайшее обычное множест
        print('Ближайшее обычное множество')
        for o in Om_:               #построчно выводим ближайшее обычн
            print(o)
        print()

        MP_=MP(Om_,k)                 #по матрице ближайшего обычного 
        print('Матрица парного сходства')
        for mp in MP_:
            print(mp)
        print()

        APOS_=APOS(Om_)             #по матрице ближайшего обычного мн
                                    #общей согласованности
        print(f'Аддитивный показатель общей согласованности={APOS_}
        print()

        sumMP_=sumMP(MP_)           #находим сумму столбцов матрицы па
        print('Сумма столбцов матрицы парного сходства')
        print(sumMP_)
        print()

                                #объявляем и обнуляем итератор
    while i<year:                   #создаем цикл расчета для каждого 
        matr=matr_y[i]              #берем данные экспертных оценок за
        print(f'Год {i+1}')         #выводим текущий год
        k=K
        pod_funk()
        while APOS_<0.6:
            sumMP_np=np.array(sumMP_)
            sum_MP_min=np.min(sumMP_np)
            min_k=0
            for s in sumMP_np:
                if s==sum_MP_min:
                    break
                min_k+=1
            print(f'Показатель согласованности {APOS_}<0.6')
            print(f'Удаляем данные эксперта №{min_k+1}')
            matr2=matr.pop(min_k)
```

```
            k-=1
            pod_funk()


        S1_,S2_=S1_S2(matr,M_x_) #находим S1,S2 для множества 2 типа
        print(f'S1={S1_}')
        print(f'S2={S2_}')
        print()

        t=T[k-3]
        print(f'Параметр t={t}')
        print()

        F_up_,F_down_=F_up_down(M_x_,S1_,S2_,t) #находим F_up,F_down
        F_up_akkum.append(F_up_)                #Записываем полученны
        F_down_akkum.append(F_down_)
        print(f'F_up={F_up_}')
        print(f'F_down={F_down_}')
        print()

        drawF(F_up_,F_down_)                    #Выводим график обобще
        print()

        print(f'Коэфициент дисконтирования={R_[i]}')
        print()

        Cf_=Cf(R_[i],F_up_,F_down_)             #Выводим матрицу Cf 2
        Cf_akkum.append(Cf_)
        print('Cf--2--')
        for cf in Cf_:
            print(cf)
        print()

        Cf_1_=Cf_1(R_[i],M_x_)                  #Выводим матрицу Cf 1
        Cf_1_akkum.append(Cf_1_)
        print('Cf--1--')
        print(Cf_1_)
        print()

        print('**************************************************
        print()
        i+=1
    Cf_sum_=Cf_sum(Cf_akkum)       #Получаем сумму всех матриц Cf 2
    print('Сумм Cf--2--')
    for cf in Cf_sum_:
        print(cf)
    print()

    NPV_=NPV(Cf_sum_,C)           #Получаем NPV 2 типа
    print('NPV--2--')
    for npv in NPV_:
        print(npv)
    print()
```

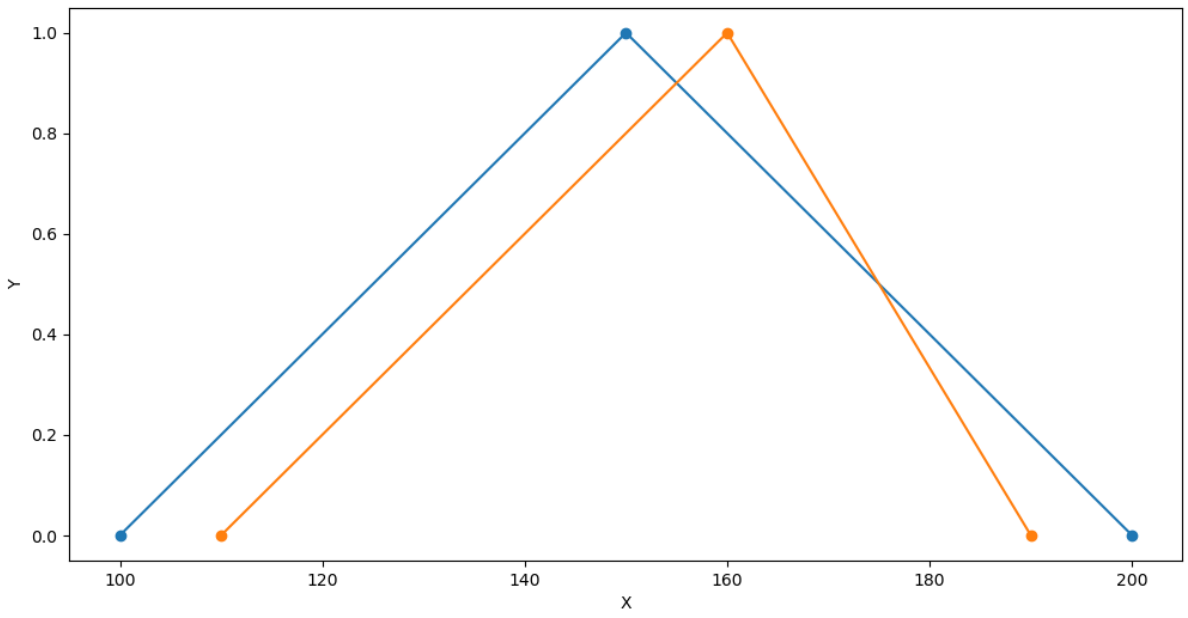
```
    drawF(NPV_[0],NPV_[1])       #Выводим график NPV 2 типа
    print()

    Cf_sum_1_=Cf_sum_1(Cf_1_akkum)   #Получаем сумму всех матриц Cf
    print('Сумм Cf--1--')
    print(Cf_sum_1_)
    print()

    print(f"Коэффициент риска={risk}")
    print()

    NPV_1_=NPV_1(NPV_,risk)        #Получаем NPV 1 типа
    print('NPV--1--')
    print(NPV_1_)
    print()

    drawF1(NPV_1_)                  #Выводим график NPV 1 типа
    print()
    return NPV_1_
```

In [26]:

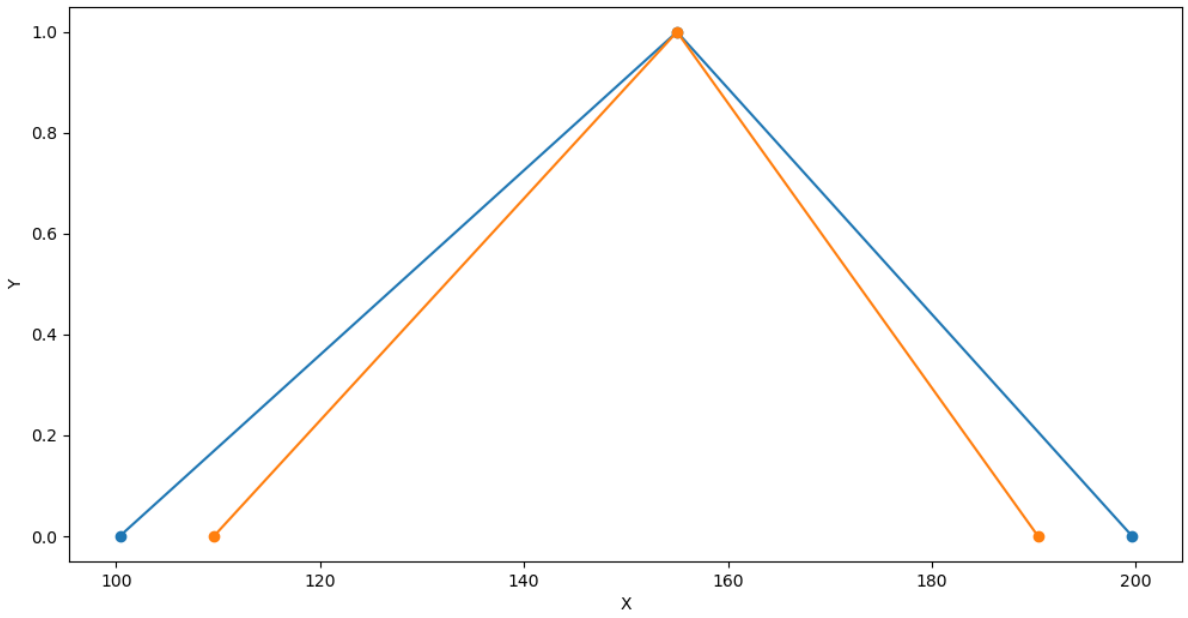
```
k_prj=win_k('Количество проектов')
k_p=1
Npv1=[]
while k_p<=k_prj:
    print("*************************************")
    print(f"*            Проект {k_p}              *")
    print("*************************************")
    Npv1.append(procedura(k_p))
    k_p+=1

if len(Npv1)>1:
    print('Ранжирование проектов по шансу на успех:')
    i=0
    usp_mass=[]
    while i<len(Npv1):
        j=0
        F1=Npv1[i]
        mass_proj=[]
        while j<len(Npv1):
            F2=Npv1[j]
            if i!=j:
                print(f'Сравниваем проект №{i+1} и проект №{j+1}')
                c1=_max(F2[0]-F1[0],0)+_max(F2[1]-F1[1],0)+_max(F2[
                c2=abs(F2[0]-F1[0])+abs(F2[1]-F1[1])+abs(F2[2]-F1[2
                c=c1/c2
                usp=_max(_min(1-c,1),0)
                print(f'Проект {i+1} успешнее, чем проект {j+1} на
                print()
                mass_proj.append(usp)
            else:
                mass_proj.append(1)
            j+=1
        usp_mass.append(mass_proj)
```

```
        i+=1
    usp_sum={}
    i=1
    for um in usp_mass:
        #print(um)
        um_np=np.array(um)
        summ=np.sum(um_np)
        usp_sum[i]=summ
        i+=1

    sorted_values = sorted(usp_sum.values())
    sorted_dict = {}
    for i in sorted_values:
        for k in usp_sum.keys():
            if usp_sum[k] == i:
                sorted_dict[k] = usp_sum[k]
                break

    print('Сортировка проектов по восходящей')
    i=1
    for k,v in sorted_dict.items():
        print(f'Ранг {i} Проект {k}')
        i+=1
```

```
**************************************
*            Проект 1                *
**************************************
Год 1
Количество экспертов=2

Повторный ввод экспертных оценок за 1 год
[100.0, 150.0, 200.0]
[110.0, 160.0, 190.0]
```

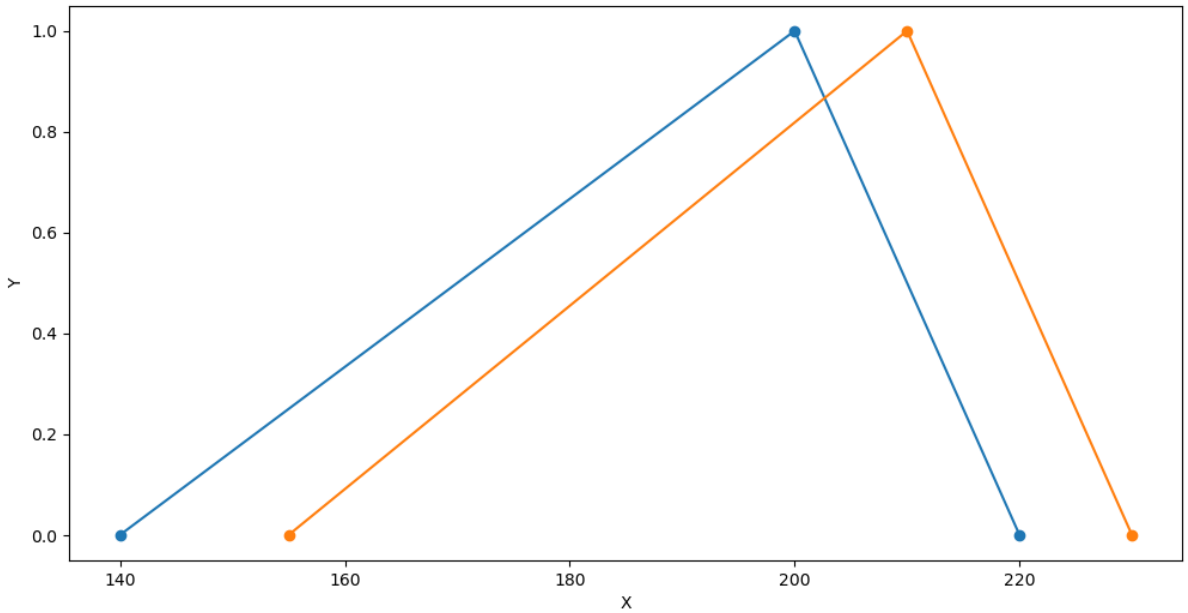


```
M(x)=[105.0, 155.0, 195.0]

Ближайшее обычное множество
[125.0, 175.0]
[135.0, 175.0]

Матрица парного сходства
[1.0, 0.8]
[0.8, 1.0]

Аддитивный показатель общей согласованности=0.8

Сумма столбцов матрицы парного сходства
[1.8 1.8]

S1=4.0824829046386295
S2=4.0824829046386295

Параметр t=2.26

F_up=[100.38679431775834, 155.0, 199.61320568224164]
F_down=[109.61320568224166, 155.0, 190.38679431775836]
```

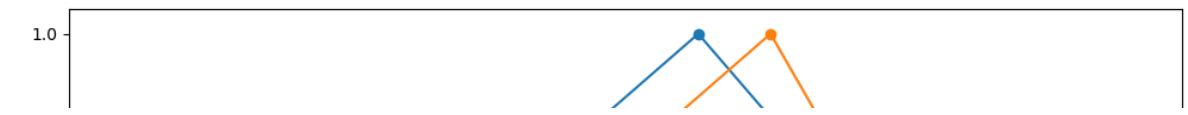


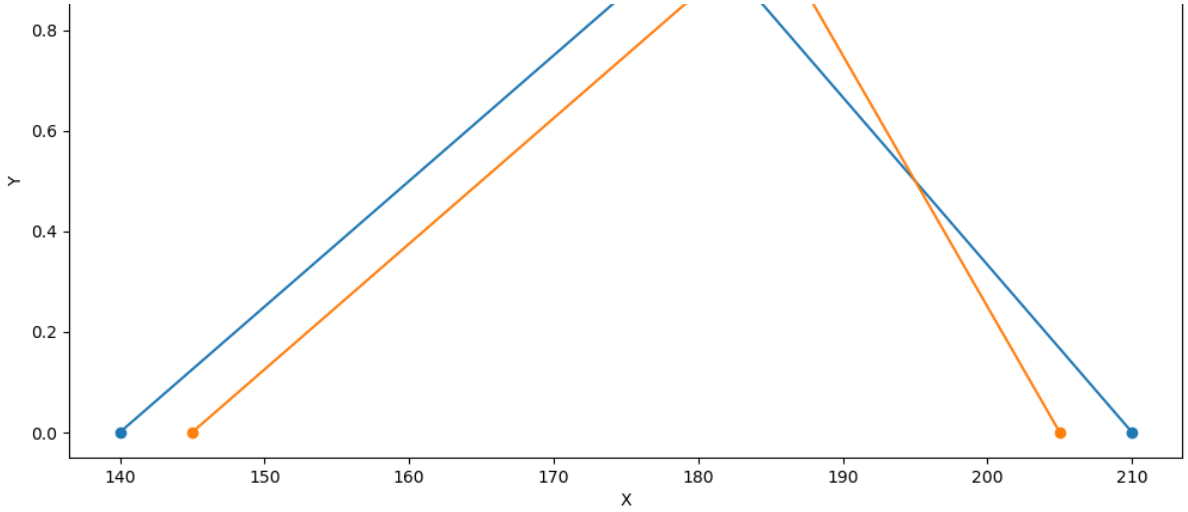

```
Коэфициент дисконтирования=1.05

Cf--2--
[95.60647077881747, 147.61904761904762, 190.1078149354682]
[104.39352922118252, 147.61904761904762, 181.3207564931032]

Cf--1--
[100.0, 147.61904761904762, 185.7142857142857]

**********
```

```
Год 2
Количество экспертов=2

Повторный ввод экспертных оценок за 2 год
[140.0, 200.0, 220.0]
[155.0, 210.0, 230.0]
```

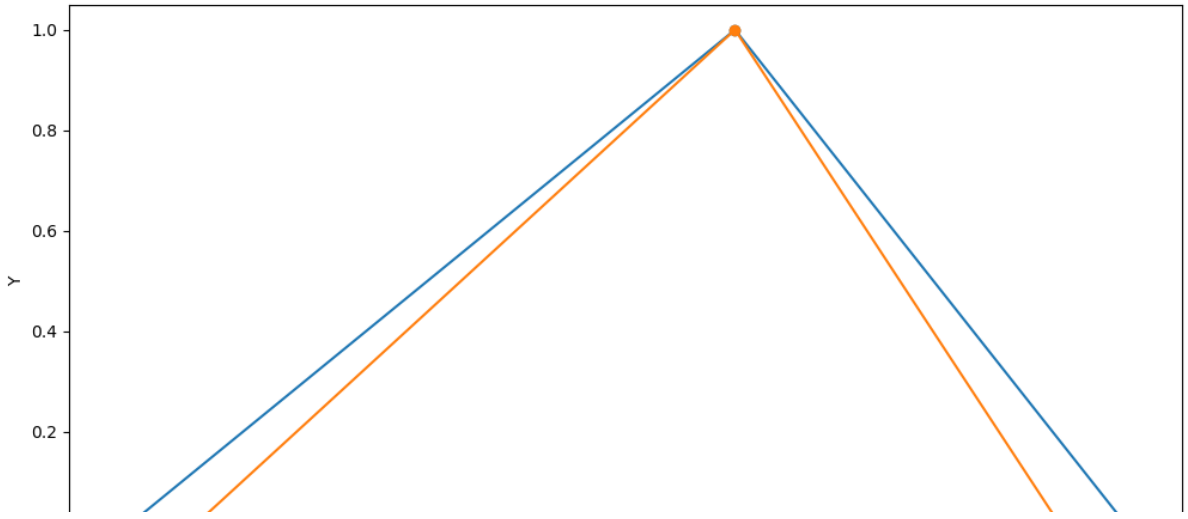


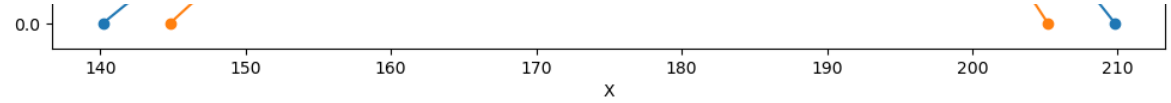

```
M(x)=[147.5, 205.0, 225.0]

Ближайшее обычное множество
[170.0, 210.0]
[182.5, 220.0]

Матрица парного сходства
[1.0, 0.55]
[0.55, 1.0]

Аддитивный показатель общей согласованности=0.55

Сумма столбцов матрицы парного сходства
[1.55 1.55]

Показатель согласованности 0.55<0.6
Удаляем данные эксперта №1
Количество экспертов=1

Повторный ввод экспертных оценок за 2 год
[140.0, 180.0, 210.0]
[145.0, 185.0, 205.0]
```



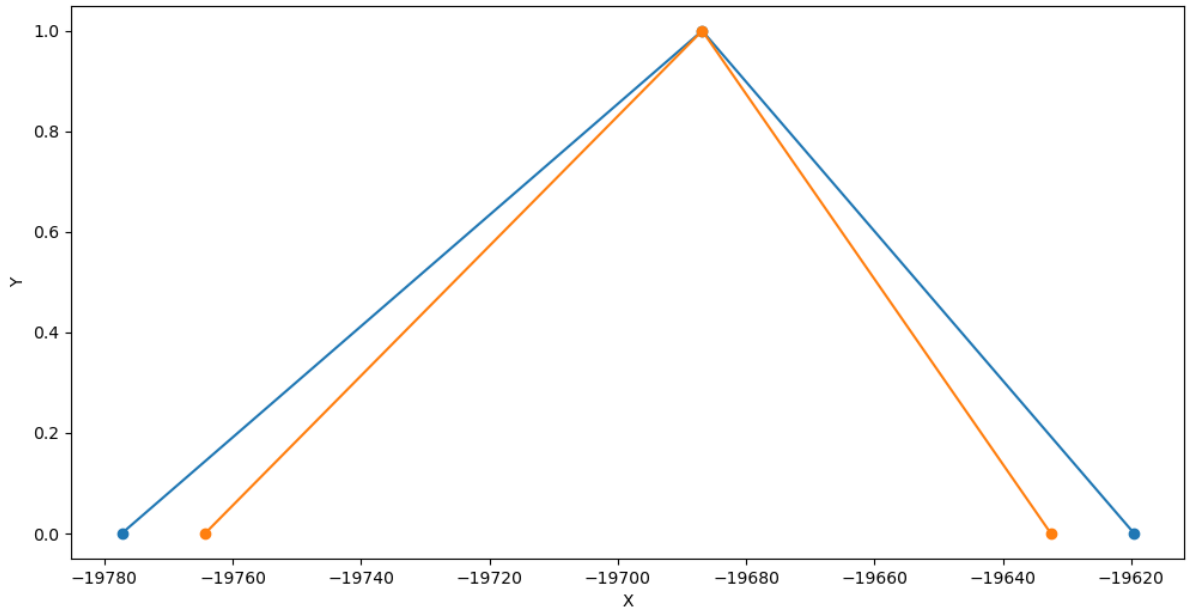



```
M(x)=[142.5, 182.5, 207.5]

Ближайшее обычное множество
[160.0, 195.0]
[165.0, 195.0]

Матрица парного сходства
[1.0, 0.86]
[0.86, 1.0]

Аддитивный показатель общей согласованности=0.86

Сумма столбцов матрицы парного сходства
[1.86 1.86]

S1=2.0412414523193148
S2=2.0412414523193148

Параметр t=2.26

F_up=[140.19339715887918, 182.5, 209.80660284112082]
F_down=[144.80660284112082, 182.5, 205.19339715887918]
```

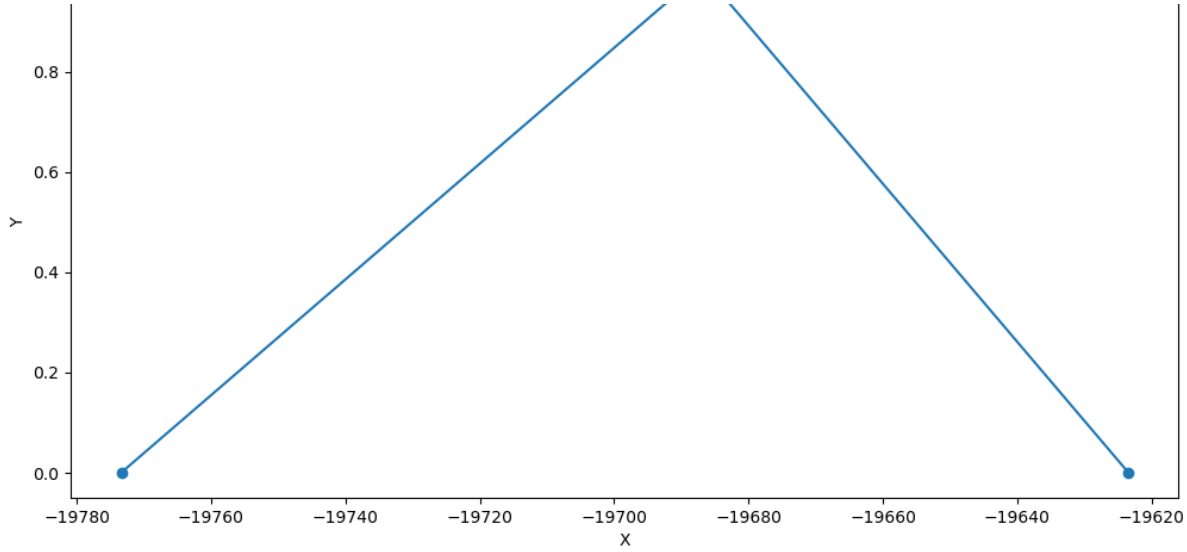




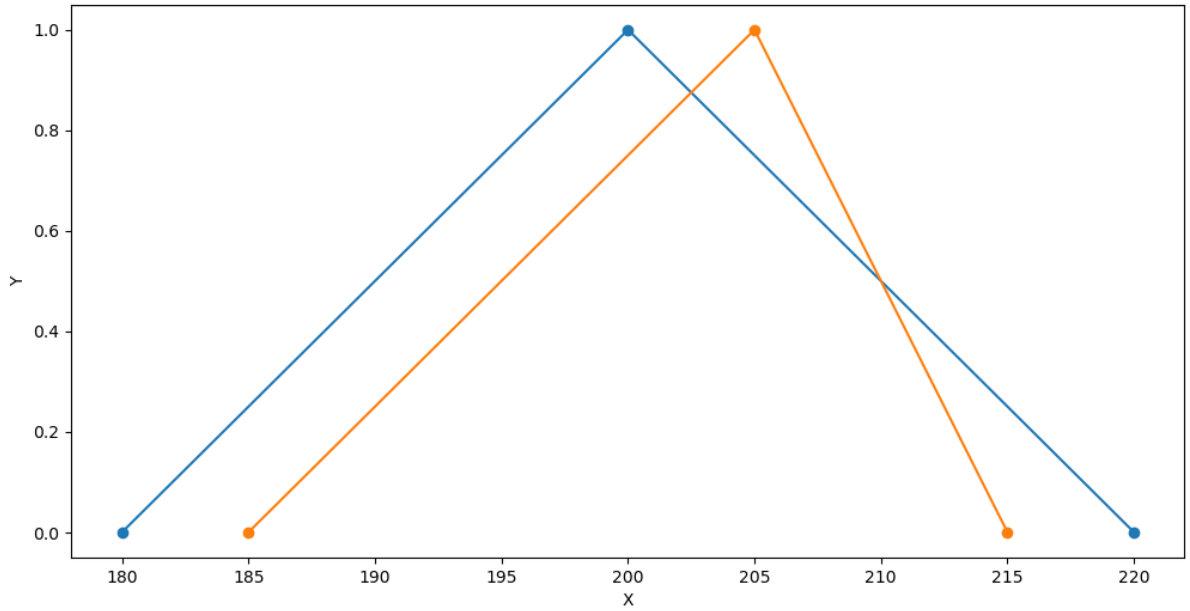

```
Коэфициент дисконтирования=1.1025

Cf--2--
[127.15954390828043, 165.5328798185941, 190.30077355203701]
[131.3438574522638, 165.5328798185941, 186.11646000805368]

Cf--1--
[129.2517006802721, 165.5328798185941, 188.20861678004533]

**********************************************************************

Сумм Cf--2--
[222.76601468709788, 313.1519274376417, 380.4085884875052]
[235.7373866734463, 313.1519274376417, 367.43721650115685]

NPV--2--
[-19777.2339853129, -19686.84807256236, -19619.591411512494]
[-19764.262613326555, -19686.84807256236, -19632.562783498845]
```




```
Сумм Cf--1--
[229.2517006802721, 313.1519274376417, 373.922902494331]

Коэффициент риска=0.7

NPV--1--
[-19773.342573716996, -19686.84807256236, -19623.4828231084]
```

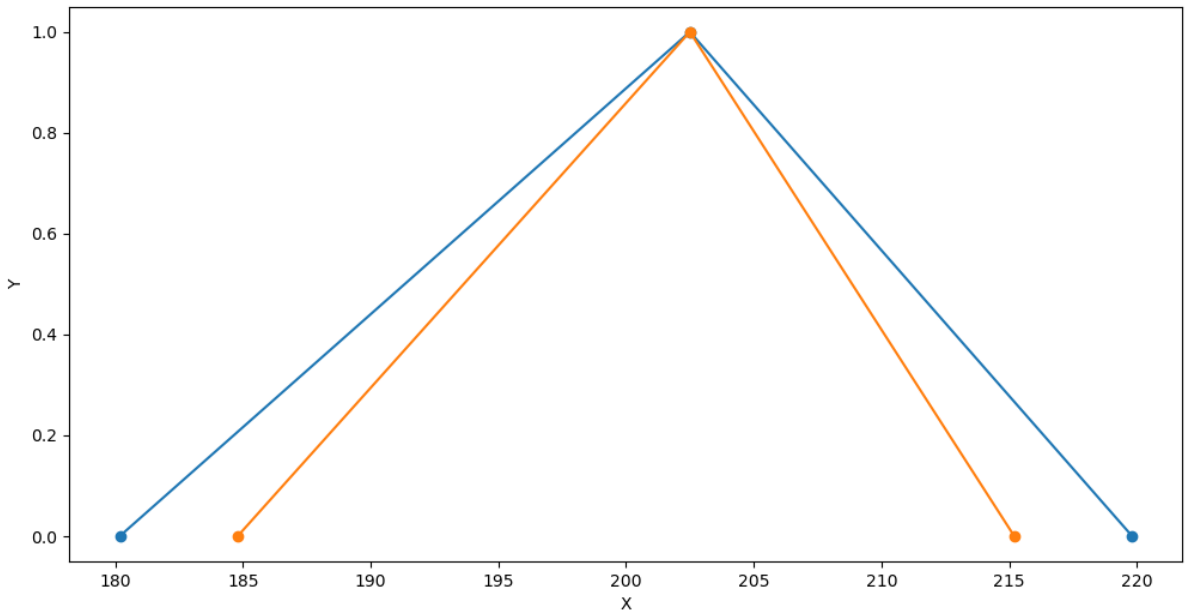



```
**************************************
*            Проект 2                *
**************************************
Год 1
Количество экспертов=2

Повторный ввод экспертных оценок за 1 год
[180.0, 200.0, 220.0]
[185.0, 205.0, 215.0]
```

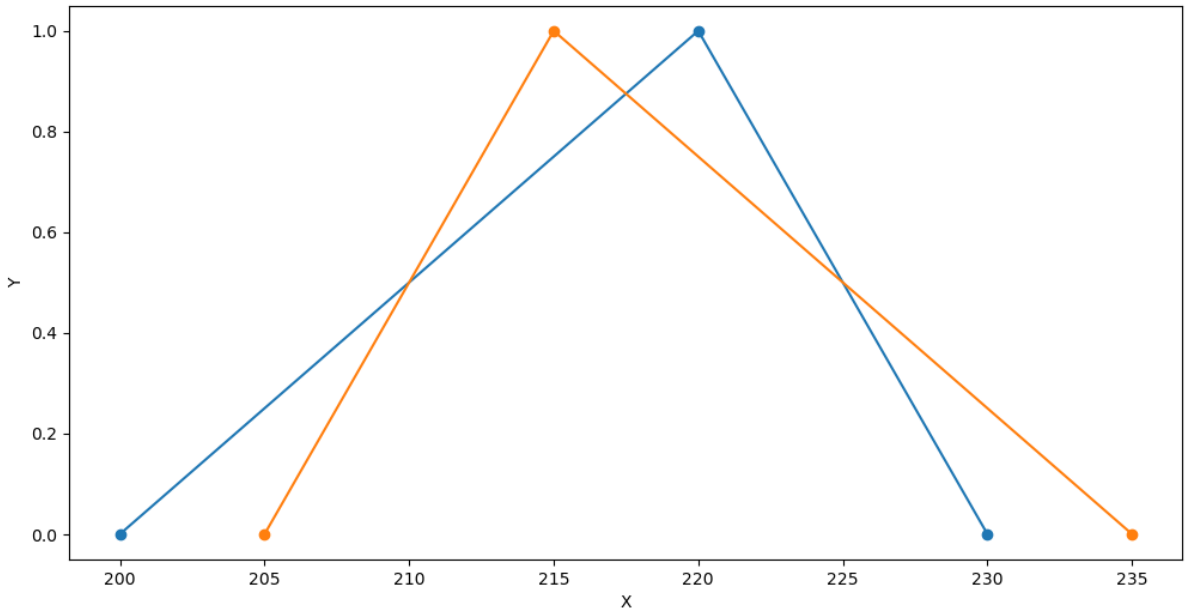



```
M(x)=[182.5, 202.5, 217.5]

Ближайшее обычное множество
[190.0, 210.0]
[195.0, 210.0]

Матрица парного сходства
```

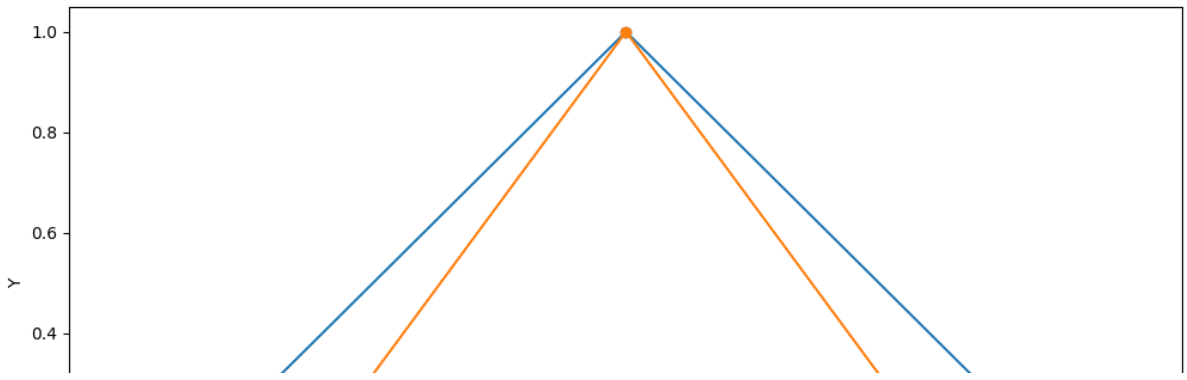
```
[1.0, 0.75]
[0.75, 1.0]

Аддитивный показатель общей согласованности=0.75

Сумма столбцов матрицы парного сходства
[1.75 1.75]

S1=2.0412414523193148
S2=2.0412414523193148

Параметр t=2.26

F_up=[180.19339715887918, 202.5, 219.80660284112082]
F_down=[184.80660284112082, 202.5, 215.19339715887918]
```

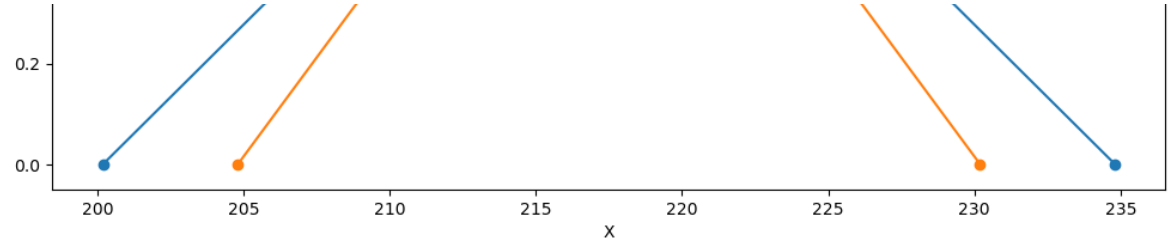



```
Коэфициент дисконтирования=1.05

Cf--2--
[171.61275919893254, 192.85714285714286, 209.3396217534484]
[176.00628842011506, 192.85714285714286, 204.94609253226588]

Cf--1--
[173.8095238095238, 192.85714285714286, 207.14285714285714]

**********

Год 2
Количество экспертов=2

Повторный ввод экспертных оценок за 2 год
[200.0, 220.0, 230.0]
[205.0, 215.0, 235.0]
```




```
M(x)=[202.5, 217.5, 232.5]

Ближайшее обычное множество
[210.0, 225.0]
[210.0, 225.0]

Матрица парного сходства
[1.0, 1.0]
[1.0, 1.0]

Аддитивный показатель общей согласованности=1.0

Сумма столбцов матрицы парного сходства
[2. 2.]

S1=2.0412414523193148
S2=2.0412414523193148

Параметр t=2.26

F_up=[200.19339715887918, 217.5, 234.80660284112082]
F_down=[204.80660284112082, 217.5, 230.19339715887918]
```





Коэфициент дисконтирования=1.1025

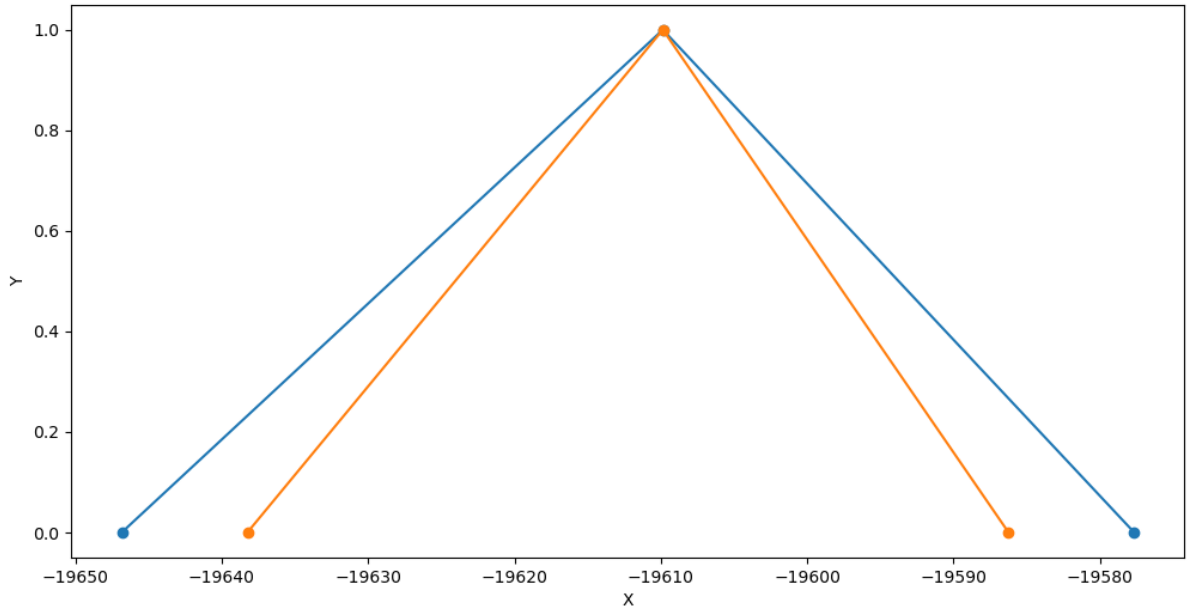
Cf--2--
[181.5813126157634, 197.27891156462584, 212.97651051348828]
[185.76562615974677, 197.27891156462584, 208.79219696950491]

Cf--1--
[183.6734693877551, 197.27891156462584, 210.8843537414966]

****

Сумм Cf--2--
[353.1940718146959, 390.1360544217687, 422.3161322669367]
[361.7719145798618, 390.1360544217687, 413.7382895017708]

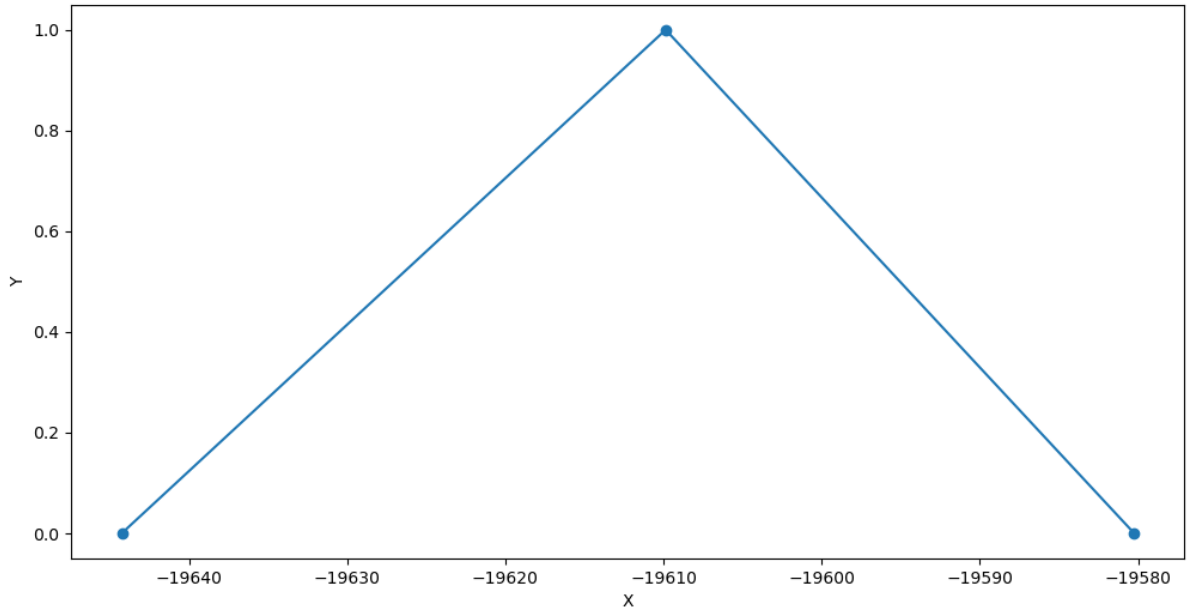
NPV--2--
[-19646.805928185306, -19609.863945578232, -19577.683867733063]
[-19638.228085420138, -19609.863945578232, -19586.26171049823]



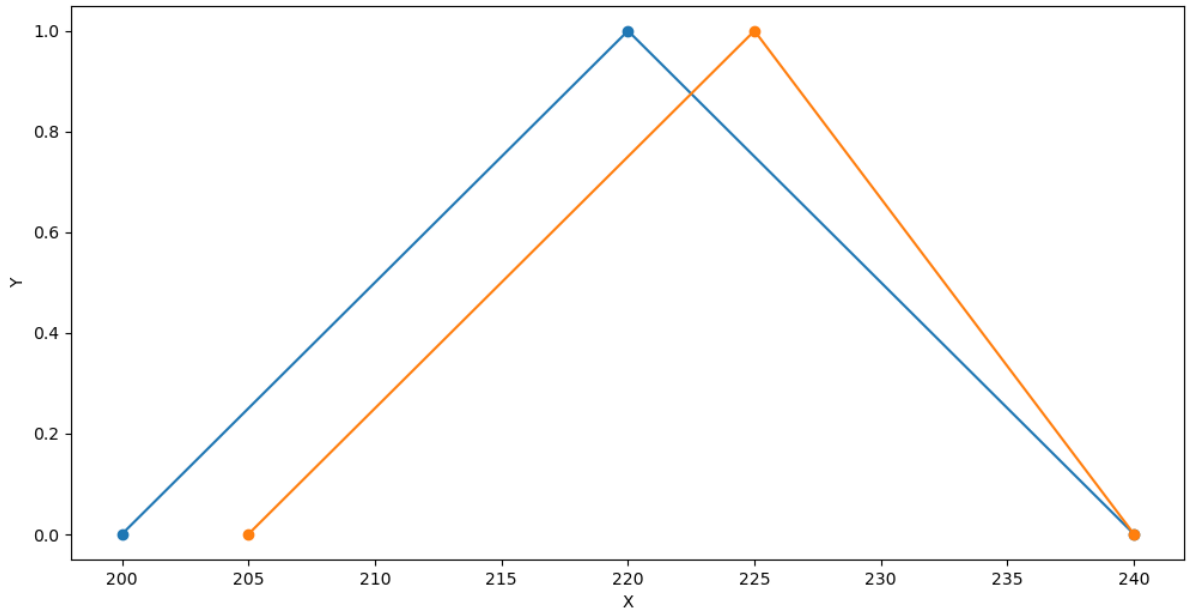
Сумм Cf--1--
[357.4829931972789, 390.1360544217687, 418.02721088435374]

Коэффициент риска=0.7

NPV--1--
[-19644.232575355756, -19609.863945578232, -19580.257220562613]

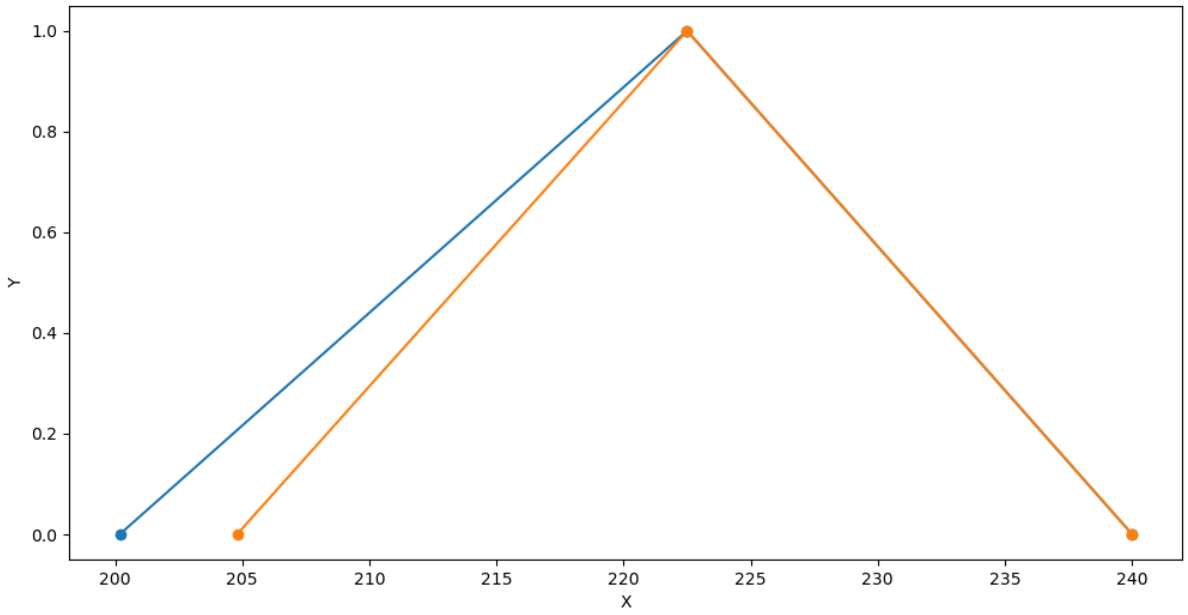



```
************************************
*            Проект 3              *
************************************
Год 1
Количество экспертов=2

Повторный ввод экспертных оценок за 1 год
[200.0, 220.0, 240.0]
[205.0, 225.0, 240.0]
```



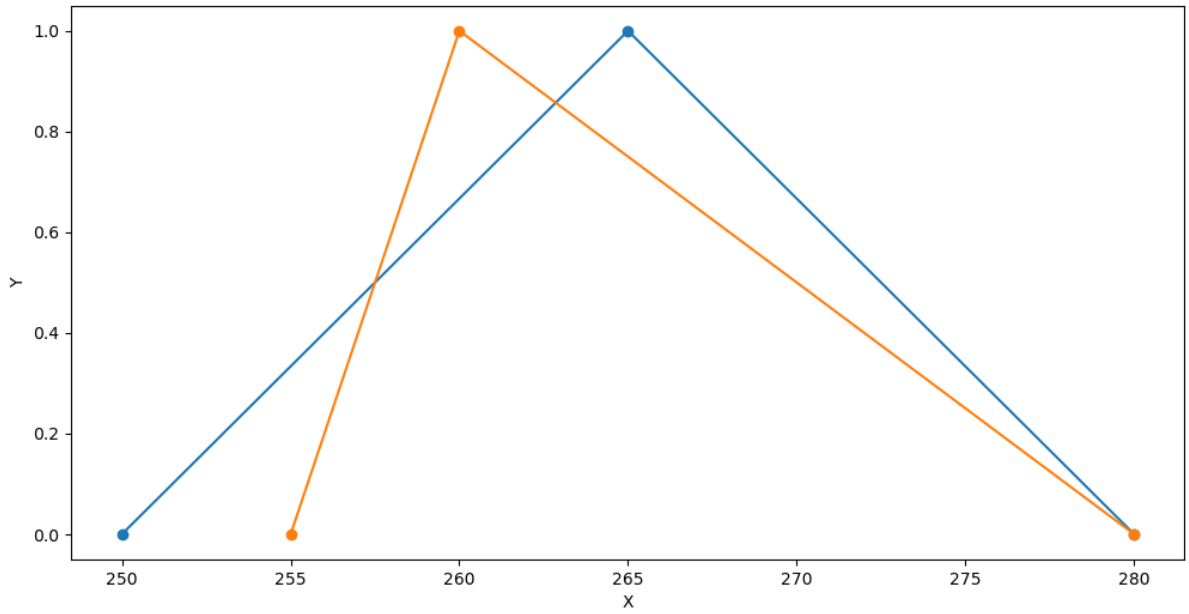

```
M(x)=[202.5, 222.5, 240.0]

Ближайшее обычное множество
```

```
[210.0, 230.0]
[215.0, 232.5]

Матрица парного сходства
[1.0, 0.67]
[0.67, 1.0]

Аддитивный показатель общей согласованности=0.67

Сумма столбцов матрицы парного сходства
[1.67 1.67]

S1=2.0412414523193148
S2=0.0

Параметр t=2.26

F_up=[200.19339715887918, 222.5, 240.0]
F_down=[204.80660284112082, 222.5, 240.0]
```

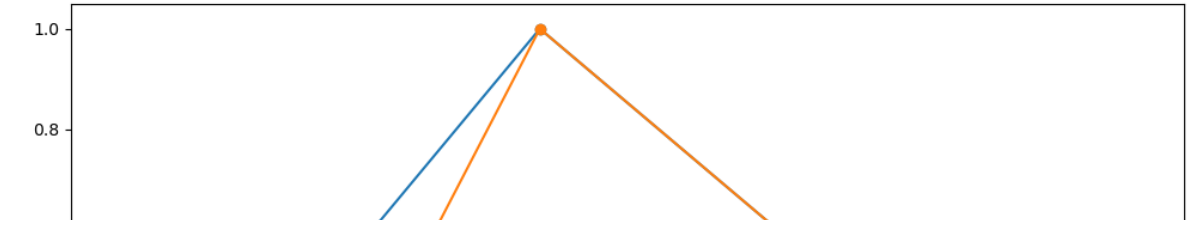


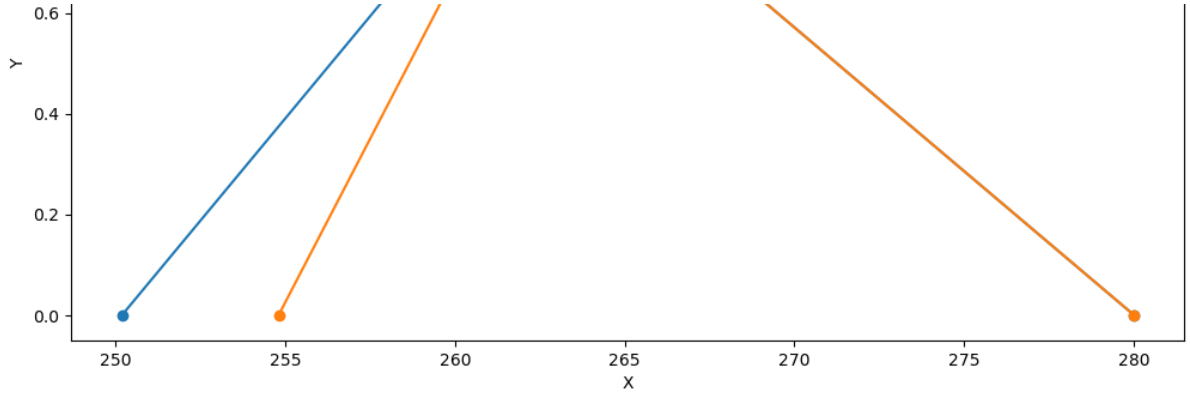

```
Коэфициент дисконтирования=1.05

Cf--2--
[190.6603782465516, 211.9047619047619, 228.57142857142856]
[195.05390746773412, 211.9047619047619, 228.57142857142856]

Cf--1--
[192.85714285714286, 211.9047619047619, 228.57142857142856]

**********

Год 2
Количество экспертов=2
```

```
Повторный ввод экспертных оценок за 2 год
[250.0, 265.0, 280.0]
[255.0, 260.0, 280.0]
```

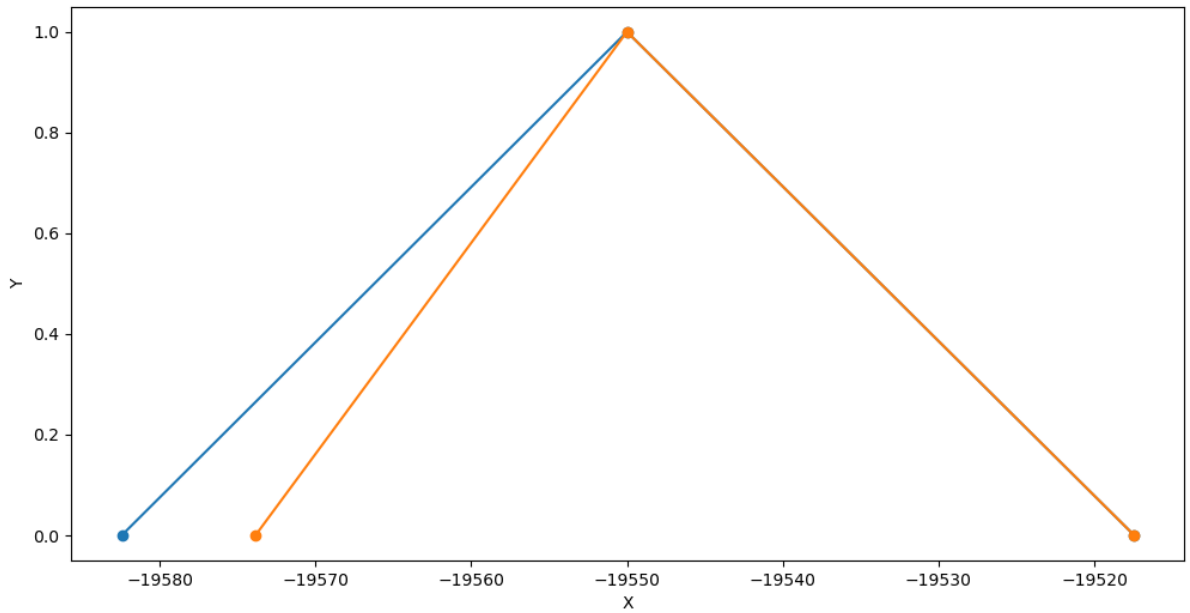



```
M(x)=[252.5, 262.5, 280.0]

Ближайшее обычное множество
[257.5, 272.5]
[257.5, 270.0]

Матрица парного сходства
[1.0, 0.83]
[0.83, 1.0]

Аддитивный показатель общей согласованности=0.83

Сумма столбцов матрицы парного сходства
[1.83 1.83]

S1=2.0412414523193148
S2=0.0

Параметр t=2.26

F_up=[250.19339715887918, 262.5, 280.0]
F_down=[254.80660284112082, 262.5, 280.0]
```



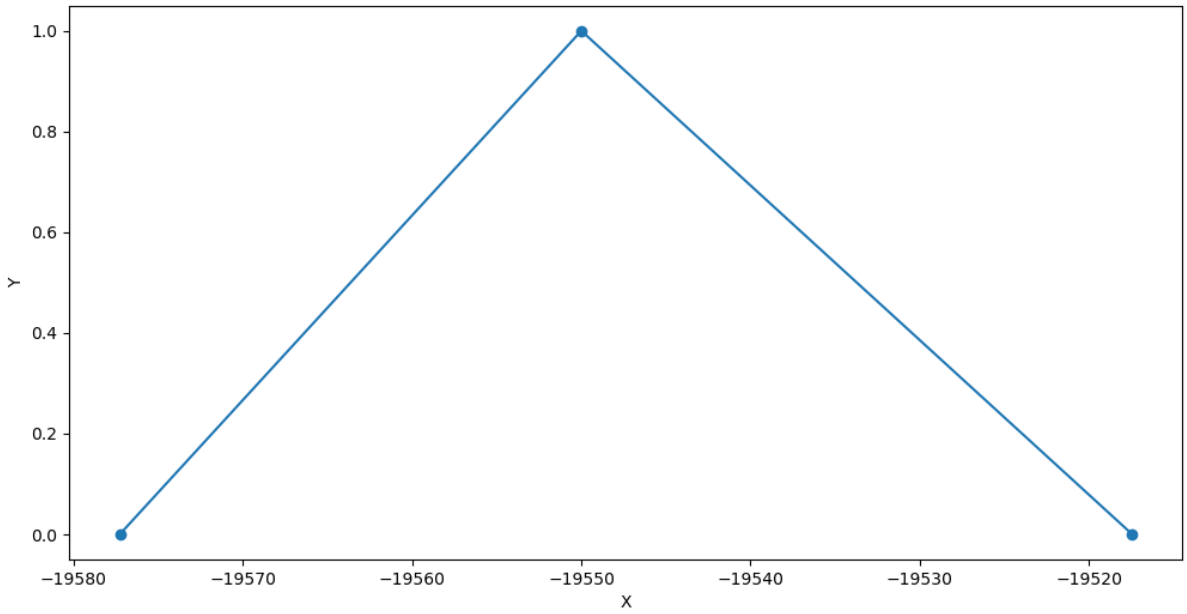



```
Коэфициент дисконтирования=1.1025

Cf--2--
[226.9327865386659, 238.09523809523807, 253.96825396825395]
[231.11710008264927, 238.09523809523807, 253.96825396825395]

Cf--1--
[229.0249433106576, 238.09523809523807, 253.96825396825395]

**********************************************************************

Сумм Cf--2--
[417.5931647852175, 450.0, 482.53968253968253]
[426.17100755038337, 450.0, 482.53968253968253]

NPV--2--
[-19582.406835214784, -19550.0, -19517.460317460318]
[-19573.828992449617, -19550.0, -19517.460317460318]
```




```
Сумм Cf--1--
[421.8820861678005, 450.0, 482.53968253968253]
```

```
Коэффициент риска=0.4

NPV--1--
[-19577.260129555685, -19550.0, -19517.460317460318]
```




```
Ранжирование проектов по шансу на успех:
Сравниваем проект №1 и проект №2
Проект 1 успешнее, чем проект 2 на 4%

Сравниваем проект №1 и проект №3
Проект 1 успешнее, чем проект 3 на 0%

Сравниваем проект №2 и проект №1
Проект 2 успешнее, чем проект 1 на 96%

Сравниваем проект №2 и проект №3
Проект 2 успешнее, чем проект 3 на 0%

Сравниваем проект №3 и проект №1
Проект 3 успешнее, чем проект 1 на 100%

Сравниваем проект №3 и проект №2
Проект 3 успешнее, чем проект 2 на 100%

Сортировка проектов по восходящей
Ранг 1 Проект 1
Ранг 2 Проект 2
Ранг 3 Проект 3
```

In [ ]:

In [ ]:

In [ ]:

In [ ]: