1. Проблема

2. Содержательная постановка

Необходимо определить процентный состав шихты для выплавки стали, нормированной по ГОСТу и обладающей минимальной стоимостью материалов с учетом ограничений на количество материалов.

3. Формальная постановка

Пусть

- $X = x_1, x_2, x_3$ - пространство долей компонентов шихты для выплавки стали.
- $C = c_1, c_2, c_3$ - пространство весов (цен) компонентов шихты для выплавки стали.

Задача:

$$(C, X) = \sum_{i=1}^{3} c_i x_i = c_1 x_1 + c_2 x_2 + c_3 x_3 \to \min_{x_1 + x_2 + x_3 = 200}$$

Ограничения нормированности по ГОСТу (условия):

$0.15x_1 + 0.15x_2 + 0.17x_3 \geq 0.16$

$0.15x_1 + 0.15x_2 + 0.17x_3 \leq 0.18$

$0.10x_1 + 0.08x_2 + 0.09x_3 \leq 0.09$

$0.007x_1 + 0.003x_2 + 0.005x_3 \leq 0.005$

$0 \leq x$

Ограничения на количество материалов:

$x_1 \leq 150$
$x_2 \leq 100$
$x_3 \leq 75$

4. Алгоритм и ПО

В качестве ПО будем использовать ЯП **Python** с подключенными модулями:

- **numpy** - для работы с линейной алгеброй
- **cvxpy** - для работы с линейным программированием

В качестве среды разработки **Jupyter Lab**,

5. Решение задачи

Приведем решение задачи с применением выбранного алгоритма.

Возьмем функцию из прошлой задачи:

In [1]:
```
def optimal_receipt(c, A, b=[0.16, 0.18, 0.09, 0.005], cost_accuracy=None, percentage_accuracy=3):
    '''
    Функция, возвращающая оптимальные
    доли материалов шихты

    c - список
    стоимостей материалов

    A - 2D-список
    составов материалов

    b - список
    значений по ГОСТ'у
    (по умолчанию, как в задаче)

    cost_accuracy - целое число
    знаков после запятой стоимости за тонну
    (по умолчанию None - до целого числа)

    percentage_accuracy - целое число
    знаков после запятой значений оптимальных долей
    (по умолчанию 3)
    '''

    import cvxpy
    import numpy as np
    from warnings import warn

    for i in c:
        if i < 0:
            raise Exception('Значение стоимости не может быть отрицательным')
    for i in A:
        for j in i:
            if j < 0:
                raise Exception('Значение состава материала не может быть отрицательным')
    for i in b:
        if i < 0:
            warn('Значение ограничения отрицательное, возможны проблемы!')

    c = np.array(c)
    A = np.array(A)
    b = np.array(b)

    x = cvxpy.Variable(shape=len(c), integer = False)

    constraints = [(A[0,:] @ x >= b[0]),
                   (A[0,:] @ x <= b[1]),
                   (A[1,:] @ x <= b[2]),
                   (A[2,:] @ x <= b[3]),
                   (sum(x) == 1),
                   (x >= 0)]

    total_value = c @ x
    problem = cvxpy.Problem(cvxpy.Minimize(total_value), constraints=constraints)

    try:
        print('Оптимальная стоимость тонны: %s' %round(problem.solve(), cost_accuracy))
    except:
        print('Нет решения по ГОСТ\'у.')
        return
    print('Оптимальные доли материалов: {} : {} : {}.'.format(abs(round(x.value[0],percentage_accuracy)),
                                                              abs(round(x.value[1],percentage_accuracy)),
                                                              abs(round(x.value[2],percentage_accuracy))))
    return [abs(round(i,percentage_accuracy)) for i in x.value]
```

Запишем функцию для этой задачи:

In [3]:
```
def receipt(c, A, in_stock, need, b=[0.16, 0.18, 0.09, 0.005], cost_accuracy=None, percentage_accuracy=3):
    '''
    Функция, возвращающая оптимальные
    доли материалов шихты

    c - список
    стоимостей материалов

    A - 2D-список
    составов материалов

    in_stock - список
    количества материалов в наличии

    need - число,
    которое надо произвести

    b - список
    значений по ГОСТ'у
    (по умолчанию, как в задаче)

    cost_accuracy - целое число
    знаков после запятой стоимости за тонну
    (по умолчанию None - до целого числа)

    percentage_accuracy - целое число
    знаков после запятой значений оптимальных долей
    (по умолчанию 3)
    '''
```

```
    import cvxpy
    import numpy as np
    import warnings

    warnings.filterwarnings("ignore", category=RuntimeWarning)
    for i in c:
        if i < 0:
            raise Exception('Значение стоимости не может быть отрицательным')
    for i in A:
        for j in i:
            if j < 0:
                raise Exception('Значение состава материала не может быть отрицательным')
    for i in in_stock:
        if i < 0:
            raise Exception('Значение количества материалов в наличии не может быть отрицательным')
    for i in b:
        if i < 0:
            warnings.warn('Значение ограничения отрицательное, возможны проблемы!')

    c = np.array(c)
    A = np.array(A)
    b = np.array(b) * need
    in_stock = np.array(in_stock)

    x = cvxpy.Variable(shape=len(c), integer = False)

    constraints = [(A[0,:] @ x >= b[0]),
                   (A[0,:] @ x <= b[1]),
                   (A[1,:] @ x <= b[2]),
                   (A[2,:] @ x <= b[3]),
                   (sum(x) == need),
                   (x >= 0),
                   (x[0] <= in_stock[0]),
                   (x[1] <= in_stock[1]),
                   (x[2] <= in_stock[2])]

    total_value = c @ x
    problem = cvxpy.Problem(cvxpy.Minimize(total_value), constraints=constraints)

    try:
        print('Стоимость тонны, если выполнить условие по количеству: %s' %round(problem.solve()/need, cost_accuracy))
        print('Доли материалов, если выполнить условие по количеству: {} : {} : {}.'.format(abs(round(x.value[0],percentage_
                                                                                              abs(round(x.value[1],percentage_
                                                                                              abs(round(x.value[2],percentage_
    except:
        print('Нельзя произвести необходимое число материала. Посчитаем сколько материала максимум мы можем произвести по оп

    optimal = np.array(optimal_receipt(c, A, b=b / need, cost_accuracy=cost_accuracy, percentage_accuracy=percentage_accurac
    coef = min(in_stock / optimal)
    optimal_values = optimal * coef
    output = sum(optimal_values)

    print('Сколько материалов потребуется по оптимальному рецепту: {} : {} : {}.'.format(abs(round(optimal_values[0],percent
                                                                                          abs(round(optimal_values[1],percenta
                                                                                          abs(round(optimal_values[2],percenta
    print('В сумме произведем: %s тонн' %round(output,percentage_accuracy))
```

6. Анализ

Запустим функцию на наших данных:

In [4]:
```
c = [300,200,150]

A = [[0.15,0.15,0.17],
     [0.1,0.08,0.09],
     [0.007,0.003,0.005]]

need = 200

in_stock = [150,100,75]

receipt(c, A, in_stock, need)
```

```
Нельзя произвести необходимое число материала. Посчитаем сколько материала максимум мы можем произвести по оптимальному р
ецепту:
Оптимальная стоимость тонны: 150
Оптимальные доли материалов: 0.0 : 0.0 : 1.0.
Сколько материалов потребуется по оптимальному рецепту: 0.0 : 0.0 : 75.0.
В сумме произведем: 75.0 тонн
```

В ходе решения этой задачи был получен ожидаемый ответ: нет решения для нужного числа материалов. Изначально было видно, что решения, подходящего нашим условиям нет, т.к. у нас никогда не выполнится условие по процентному содержанию хрома. Необходимо либо подкорректировать состав, либо найти дополнительных поставщиков.

Вместо этого программа выдала сколько материала можно произвести по оптимальному рецепту

Возьмем другие данные:

In [10]:
```
c = [300,200,150]

A = [[0.14,0.2,0.17],
     [0.08,0.05,0.12],
     [0.004,0.003,0.009]]
```

```
need = 200

in_stock = [150,100,75]

receipt(c, A, in_stock, need)
```

```
Стоимость тонны, если забить печь полностью: 205
Доли материалов, если забить печь полностью: 40.0 : 100.0 : 60.0.
Оптимальная стоимость тонны: 203
Оптимальные доли материалов: 0.182 : 0.515 : 0.303.
Сколько материалов потребуется по оптимальному рецепту: 35.34 : 100.0 : 58.835.
В сумме произведем: 194.175 тонн
```

Получаем, что в целом, можно забить печь полностью, но тогда стоимость тонны для нас возрастет на 2$.