1. Проблема

2. Содержательная постановка

Необходимо определить процентный состав шихты для выплавки стали, нормированной по ГОСТ'у и обладающей минимальной стоимостью 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:

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

Ограничения нормированности по ГОСТу (условия):

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

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

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

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

$0 \le x \le 1$

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.

In [ ]:
```
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('Стоимость тонны: %s' %round(problem.solve(), cost_accuracy))
    except:
        print('Нет решения по ГОСТ\'у.')
        return
    print('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 x.value
```

6. Анализ

Запустим функцию на реальных данных:

In [2]:
```
c = [300,200,150]

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

optimal_receipt(c, A);
```

```
Стоимость тонны: 150
Доли материалов: 0.0 : 0.0 : 1.0.
```

Получаем ожидаемый ответ: использовать только последний вариант. Изначально было видно, что этот вариант самый дешевый, и при этом полностью удовлетворяет ограничениям по ГОСТу.

In [5]:
```
c = [300,200,150]

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

optimal_receipt(c, A);
```

```
Стоимость тонны: 203
Доли материалов: 0.182 : 0.515 : 0.303.
```