Код

Симплекс метод

```python
import numpy as np


def simplex(a, f, basic):
    n, m = a.shape
    for i, j in enumerate(basic):
        f -= a[i, :-1] * f[j]
    while True:
        j = np.argmin(f)
        if f[j] >= 0:
            break
        col = np.ma.masked_less_equal(a[:, j], 0, copy=False)
        i = np.argmin(a[:, -1] / col)
        if a[i, j] <= 0:
            return None
        basic[i] = j
        a[i] /= a[i, j]
        for k in range(n):
            if k != i:
                a[k] -= a[i] * a[k, j]
        f -= a[i, :-1] * f[j]
    return basic


def find_basic(a):
    n, m = a.shape
    basic_a = np.hstack([a[:, :-1], np.eye(n), a[:, -1:]])
    basic_f = np.zeros(m + n - 1)
    basic_f[-n:] = 1
    basic_basic = np.arange(m - 1, m + n - 1)
    basic = simplex(basic_a, basic_f, basic_basic)
    new_a = basic_a[:, np.r_[: m - 1, -1]]
    return new_a, basic


def full_simplex(a, f):
    a, basic = find_basic(a)
    basic = simplex(a, f, basic)
    ans = np.zeros_like(f)
    ans[basic] = a[:, -1]
    return ans
```

Интерфейс для тестов

```python
import numpy as np
from scipy.optimize import linprog
from sympy import Expr, Le, symbols

from .simplex import full_simplex
```

```python
x1, x2, x3, x4, x5 = _x = symbols([f"x_{i}" for i in range(1, 6)])


def to_list(expr: Expr, n):
    return [float(expr.coeff(_x[i])) for i in range(n)]


def convert(n, objective, constraints, direction):
    fog = np.array(to_list(objective, n))
    if direction != "min":
        fog *= -1
    ntot = n + sum(not isinstance(constraint, Expr) for constraint in constraints)
    f = np.hstack([fog, np.zeros(ntot - n)])

    q = 0
    a = []
    a_ub, b_ub, a_eq, b_eq = [], [], [], []
    for constraint in constraints:
        if isinstance(constraint, Expr):
            row = to_list(constraint, n)
            a_eq.append(row.copy())
            row.extend([0] * (ntot - n))
            b = -float(constraint.as_coeff_Add()[0])
            row.append(b)
            b_eq.append(b)
        else:
            row = to_list(constraint.lhs, n)
            if isinstance(constraint, Le):
                a_ub.append(row.copy())
            else:
                a_ub.append([-x for x in row])
            row.extend([0] * q)
            q += 1
            row.append(1 if isinstance(constraint, Le) else -1)
            row.extend([0] * (ntot - n - q))
            b = float(constraint.rhs)
            row.append(b)
            if isinstance(constraint, Le):
                b_ub.append(b)
            else:
                b_ub.append(-b)
        a.append(row)
    a = np.array(a)
    return fog, f, a, a_eq, b_eq, a_ub, b_ub


def solve1(n, objective, constraints, direction):
    _fog, f, a, *_rest = convert(n, objective, constraints, direction)
    return full_simplex(a, f)[:n]


def solve(ntest, n, objective, constraints, direction="min"):
    fog, f, a, a_eq, b_eq, a_ub, b_ub = convert(n, objective, constraints, direction)
    ans = full_simplex(a.copy(), f.copy())[:n]
    real_ans = linprog(
        fog,
        A_eq=np.array(a_eq) if a_eq else None,
        b_eq=np.array(b_eq) if b_eq else None,
        A_ub=np.array(a_ub) if a_ub else None,
        b_ub=np.array(b_ub) if b_ub else None,
    ).x
    print(ntest, round(np.sum(fog * ans), 5), round(np.sum(fog * real_ans), 5))
```

Тесты

```python
from .expr import solve, x1, x2, x3, x4, x5

solve(
    ntest=1,
    n=4,
    objective=-6 * x1 - x2 - 4 * x3 + 5 * x4,
    constraints=[
        3 * x1 + x2 - x3 + x4 - 4,
        5 * x1 + x2 + x3 - x4 - 4,
    ],
)

solve(
    ntest=2,
    n=4,
    objective=-x1 - 2 * x2 - 3 * x3 + x4,
    constraints=[
        x1 - 4 * x2 - x3 - 2 * x4 + 4,
        x1 - x2 + x3,
    ],
)

solve(
    ntest=3,
    n=5,
    objective=-x1 - 2 * x2 - x3 + 3 * x4 - x5,
    constraints=[
        x1 + x2 + 2 * x4 + x5 - 5,
        x1 + x2 + x3 + 3 * x4 + 2 * x5 - 9,
        x2 + x3 + 2 * x4 + x5 - 6,
    ],
)

solve(
    ntest=4,
    n=5,
    objective=-x1 - x2 - x3 + x4 - x5,
    constraints=[
        x1 + x2 + 2 * x3 - 4,
        -2 * x2 - 2 * x3 + x4 - x5 + 6,
        x1 - x2 + 6 * x3 + x4 + x5 - 12,
    ],
)

solve(
    ntest=5,
    n=4,
    objective=-x1 + 4 * x2 - 3 * x3 + 10 * x4,
    constraints=[
        x1 + x2 - x3 - 10 * x4,
        x1 + 14 * x2 + 10 * x3 - 10 * x4 - 11,
    ],
)

solve(
    ntest=6,
    n=4,
    objective=-x1 + 5 * x2 + x3 - x4,
    constraints=[
        x1 + 3 * x2 + 3 * x3 + x4 <= 3,
        2 * x1 + 3 * x3 - x4 <= 4,
    ],
)

solve(
    ntest=7,
    n=5,
    objective=-x1 - x2 + x3 - x4 + 2 * x5,
    constraints=[
        3 * x1 + x2 + x3 + x4 - 2 * x5 - 10,
        6 * x1 + x2 + 2 * x3 + 3 * x4 - 4 * x5 - 20,
        10 * x1 + x2 + 3 * x3 + 6 * x4 - 7 * x5 - 30,
    ],
)
```

Результаты

```
1 -4.0 -4.0
2 -4.0 -4.0
3 -11.0 -11.0
4 -10.0 -10.0
5 -4.0 -4.0
6 -3.0 -3.0
7 10.0 10.0
```