```
import numpy as np
from src.base_fem import BaseFem
from src.source import area


class Minimization(BaseFem):
    def __init__(self, n, shape):
        super().__init__(n, shape)
        self.set_node_values()

    def get_elems(self, i):
        ans = []
        for _ in self.omega.triangles.triangles:
            if i in _:
                ans.append(_)
        return ans

    def get_coords_xy(self, tr):
        omega = self.omega
        trc = np.zeros((3, 2))
        trc[0][0], trc[0][1] = omega.x_coords[tr[0]], omega.y_coords[tr[0]]
        trc[1][0], trc[1][1] = omega.x_coords[tr[1]], omega.y_coords[tr[1]]
        trc[2][0], trc[2][1] = omega.x_coords[tr[2]], omega.y_coords[tr[2]]
        return trc

    def get_shape(self, tr, i):
        trc = self.get_coords_xy(tr)
        m = np.column_stack((trc, np.ones(3)))
        r = np.zeros(3)

        for j in range(3):
            if tr[j] == i:
                r[j] = 1

        a = np.linalg.solve(m, r)
        return lambda x, y: a[0] * x + a[1] * y + a[2]

    def integrate(self, tr, func, s):
        trc = self.get_coords_xy(tr)
        f12 = func((trc[0, 0] + trc[1, 0]) / 2, (trc[0, 1] + trc[1, 1]) / 2)
        f13 = func((trc[0, 0] + trc[2, 0]) / 2, (trc[0, 1] + trc[2, 1]) / 2)
        f23 = func((trc[1, 0] + trc[2, 0]) / 2, (trc[1, 1] + trc[2, 1]) / 2)
        return s / 3 * (f12 + f13 + f23)

    def set_node_values(self):
        rank = len(self.omega.x_coords)
        A = np.zeros((rank, rank))
        R = np.zeros(rank)

        for i in range(rank):
            elements = self.get_elems(i)
            for e in elements:
                si = self.get_shape(e, i)
                trc = self.get_coords_xy(e)
                a = area(trc)
                for ind in e:
                    sj = self.get_shape(e, ind)
                    if i == ind:
                        A[i, ind] += self.integrate(e, lambda x, y: si(x, y) * sj(x, y),
a)
                    else:
                        A[i, ind] += self.integrate(e, lambda x, y: si(x, y) * sj(x, y),
a) / 2

        self.node_values = np.linalg.solve(A, R)
```