```
begin
    using NPZ ✓
    using LinearAlgebra ✓
end
```

```
data =
Dict("theta" ⇒ [1.5708, 1.5708, 0.523599, 1.0472], "dr" ⇒ 6×2 Matrix{Float64}:      , '
                                                          -0.0124687   0.000193654
                                                           0.0131732  -0.104997
                                                          -0.0193151   0.0708246
                                                          -0.123992   -0.0060532
                                                           0.0700385   0.0109086
                                                           0.0725637   0.0291229
```

```
data = npzread("./data_2.npz")
```

систему уравнений вида

$$(\tilde r_i - \tilde r_j)\cdot(\tilde r_i - \tilde r_k) = |\tilde r_i - \tilde r_j|\cdot|\tilde r_i - \tilde r_k|\cos(\theta_{ijk})$$

$$\left[\vec r_{ij} := r_i - r_j, \quad \vec n_{ij} = r_{ij}/|r_{ij}|, \quad \theta = \theta_{ijk}\right]$$

$$+ dr_i - dr_j)\cdot(r_{ik} + dr_i - dr_k) = |r_{ij} + dr_i - dr_j|\cdot|r_{ij} + dr_k - dr_k|\cos\theta$$

$$(r_{ik} + r_{ij}) - dr_j\cdot r_{ik} - dr_k\cdot r_{ij} \approx \sqrt{r_{ij}^2 + 2r_{ij}\cdot(dr_i - dr_j)}\cdot\sqrt{r_{ik}^2 + 2r_{ik}\cdot(dr_i - dr_k)}\ \cos\theta$$

$$\approx \big(|r_{ij}| + n_{ij}\cdot(dr_i - dr_j)\big)\cdot\big(|r_{ik}| + n_{ik}\cdot(dr_i - dr_k)\big)\cos\theta$$

$$\approx \big(|r_{ij}||r_{ik}| + dr_i\cdot(|r_{ik}|n_{ij} + |r_{ij}|n_{ik}) - |r_{ik}|n_{ij}\cdot dr_j - |r_{ij}|$$

$$\left[\vec\alpha = |r_{ik}|(\vec n_{ik} - \vec n_{ij}\cos\theta) \qquad \vec\beta = |r_{ij}|(\vec n_{ij} - \vec n_{ik}\cos\theta)\right]$$

$$dr_i\cdot(\vec\alpha + \vec\beta) - dr_j\cdot\vec\alpha - dr_k\cdot\vec\beta = |r_{ij}||r_{ik}|\cos\theta - r_{ij}\cdot r_{ik}$$

```
make_system (generic function with 1 method)
```

```
function make_system(r, p, theta)
    A   = zeros(length(theta), length(r))
    rhs = zeros(length(theta))
    for (θ, (i, j, k), n) ∈ zip(theta, eachrow(p), 1 : length(theta))

        r_ij = r[i, :] - r[j, :]
        ρ_ij = norm(r_ij)
        n_ij = r_ij / ρ_ij

        r_ik = r[i, :] - r[k, :]
        ρ_ik = norm(r_ik)
        n_ik = r_ik / ρ_ik

        α = ρ_ik * (n_ik - n_ij * cos(θ))
        β = ρ_ij * (n_ij - n_ik * cos(θ))

        A[n, 2i-1:2i] = α + β
        A[n, 2j-1:2j] = -α
        A[n, 2k-1:2k] = -β

        rhs[n] = ρ_ij * ρ_ik * cos(θ) - r_ij ⋅ r_ik
    end
    return A, rhs
end
```

Если система $Ax = f$ недоопределённая, то мы вместо этой системы будем решать задачу оптимизации

$$\min \|x\|^2 \ \text{ при } \ Ax = f$$

Функция Лагранжа:

$$L = x^T x - \lambda^T(Ax - f)$$

Ищем стационарную точку:

$$\frac{\partial L}{\partial x} = 2x^T - \lambda^T A = 0 \qquad (1)$$

$$\frac{\partial L}{\partial \lambda} = f^T - x^T A^T = 0 \qquad (2)$$

Решаем:

$$\lambda^T A A^T \overset{(1)}{=} 2x^T A^T \overset{(2)}{=} 2f^T$$

$$\lambda^T = 2f^T\left(AA^T\right)^{-1}$$

$$x^T \overset{(1)}{=} \frac{1}{2}\lambda^T A = f^T\left(AA^T\right)^{-1}A$$

$$x = A^T\left(AA^T\right)^{-1}f$$

Выражение $A^T\left(AA^T\right)^{-1}$ – это правая псевдообратная матрица для A.

В Julia мы можем вычислить `x` как `pinv(A) * f` или даже просто как `A \ f`.

```
get_dr (generic function with 1 method)
```

```
function get_dr(data)
    r = data["r"]
    p = data["p"]
    theta = data["theta"]

    A, f = make_system(r, p, theta)
    x = A \ f
    return reshape(x, (2, :))'
end
```

```
6×2 adjoint(::Matrix{Float64}) with eltype Float64:
 -0.0395144   0.000410848
  0.0157527  -0.116101
 -0.0113353   0.0620907
 -0.125167   -0.0723292
  0.0913028   0.0844964
  0.068961    0.0414325
```

```
get_dr(data)
```

Это отличается от предполагаемого ответа:

```
6×2 Matrix{Float64}:
 -0.0124687   0.000193654
  0.0131732  -0.104997
 -0.0193151   0.0708246
 -0.123992   -0.0060532
  0.0700385   0.0109086
  0.0725637   0.0291229
```

```
data["dr"]
```

Вычислим величину $Ax - f$ для моего решения и для предполагаемого:

```
(4×12 Matrix{Float64}:                                                               , [0.01546
   2.32954  -0.0242212  -1.96025   -0.847831  …   0.872052   0.0      0.0
   0.0       0.0        -0.285975  -1.68763       0.0322582  0.0      0.0
   0.0       0.0         0.0        0.0          -0.284476   0.0      0.0
   0.0       0.0        -0.282988  -2.08919       0.0        1.06187  0.637985
```

```
A, f = make_system(data["r"], data["p"], data["theta"])
```

```
[-1.38778e-17, -5.55112e-17, 0.0, -5.55112e-17]
```

```
# мое решение: Ax - f ≈ 0
A * vec(get_dr(data)') - f
```

```
[0.00233143, -0.0255362, 0.0419974, -0.00760672]
```

```
# Предполагаемое решение:
A * vec(data["dr"]') - f
```

Для моего решения $Ax - f \approx 0$, для предполгаемого – нет. Следовательно, проблема не на этапе решения задачи наименьших квадратов, а на этапе составления линейной системы.

Вычислим также, насколько хорошо мой метод решил исходную нелинеаризованную задачу:

```
residual (generic function with 1 method)
```

```
function residual(r, p, theta)
    # Функция, которая вычисляет невязки rᵢⱼ ⋅ rᵢₖ - |rᵢⱼ| * |rᵢₖ| * cos(θ)
    ret = zeros(length(theta))
    for (n, (i, j, k)) ∈ enumerate(eachrow(p))
        r_ij = r[i, :] - r[j, :]
        r_ik = r[i, :] - r[k, :]
        ret[n] = r_ij ⋅ r_ik - norm(r_ij) * norm(r_ik) * cos(theta[n])
    end
    return ret
end
```

```
[-0.015463, -0.427638, 0.101147, -0.436696]
```

```
# исходные точки
residual(data["r"], data["p"], data["theta"])
```

```
[-0.00256702, -0.00121216, -0.0671204, -0.00896368]
```

```
# моё решение
residual(data["r"] + get_dr(data), data["p"], data["theta"])
```

```
[0.00331999, -0.0120986, 0.00435196, -0.012381]
```

```
# правильный ответ
residual(data["r"] + data["dr"], data["p"], data["theta"])
```

Как мы видим, моё решение имеет более маленькую невязку во всех уравнениях, кроме третьего. В третьем уравнении она равна 0.07, что на порядок больше, чем в остальных уравнениях.

Что именно не так с третьим уравнением, понять не удалось.