Numerical Linear Algebra Homework Project 4: Unconstrained Optimization

Catalano Giuseppe, Cerrato Nunzia

In this project we want to perform unconstrained optimization using the Newton method both in its standard form and by considering its variants which make use of the backtracking and the trust region approach. We will apply these methods to four test functions and we will compare these approaches to see how convergence changes.

1 Algorithms

```
def Newton(func, grad, hess, tol, maxit, x_0, sol_x, sol_f, alpha=1,
↪  sigma=0.0001, rho=0.5, backtracking=False):
    r''' This function implements the standard Newton method for
    ↪  unconstrained optimization.

    Parameters
    ----------
    func : function
        Function to be minimized. It must be :math:`f : \mathbb{R}^{n}
        ↪  \rightarrow \mathbb{R}`
    grad : function
        Gradient of the function. It returns a 1d-array (vector)
    hess : function
        Hessian of the function. It returns a 2d-array (matrix)
    tol : float
        Tolerance parameter for the stopping criterion
    maxit : int
        Maximum number of iterations
    x_0 : ndarray
        Starting point
    sol_x : ndarray
        Exact solution (x) to the minimization problem
    sol_f : float
        Exact minimum value of the function
    alpha : float
        Step lenght. Default value alpha=1
    sigma : float
        Constant parameter in :math:`(0,1)`. Used if backtracking = True.
        ↪  Default value sigma=0.0001
    rho : float
        Reduction parameter in :math:`(0,1)`. Used if backtracking = True.
        ↪  Default value rho=0.5

    Results
```

```
30        -------
31        results : dict
32            Dictonary of the results given by the function. It contains the
              ↪  following items:
33              - 'convergence' : (bool) True if the algorithm converges, False
                ↪  if it doesn't converge
34              - 'k' : (int) final iteration at which convergence is reached
35              - 'min_point' : (ndarray) computed point at which the minimum of
                ↪  the function is reached
36              - 'min_value' : (float) computed minimum value of the function
37              - 'interm_point' : (list) list of the intermediate points
38              - 'error_x' : (list) list that contains the 2-norm of the
                ↪  difference between each intermediate point and the exact
                ↪  solution (min_point).
39              - 'error_f' : (list) list that contains the difference between
                ↪  the function evaluated at each intermediate point and its
                ↪  value in the exact minimum point
40              - 'scalar_product' : (list) list that contains the scalar product
                ↪  between the descent direction and the gradient
41
42        '''
43    old_point = x_0
44    alpha_0 = alpha
45
46    # Create a list to save intermediate points
47    interm_points = [old_point]
48    # Create a list to save the scalar product between the gradient and the
      ↪  descent direction
49    scalar_prod = []
50
51    new_point = x_0
52    # Cycle on the number of iterations
53    for k in range(maxit):
54    gradient = grad(old_point)
55    hessian = hess(old_point)
56
57    # Compute norms for the stopping criterion
58    norm_grad = np_lin.norm(gradient)
59    norm_diff_x = np_lin.norm(new_point - old_point)
60
61    # Check if the stopping criterion is satisfied
62    if norm_grad <= tol and norm_diff_x <= tol*(1 + np_lin.norm(old_point)):
63    min_value = func(new_point)
64    conv = True
65    print(k)
66    break
67
68    # Compute the descent direction by solving a linear system
69    p = np_lin.solve(hessian, -gradient)
70    scalar_prod.append(gradient@p)
71
```

```
# Implement backtracking if backtracking == True
if backtracking == True:
alpha = alpha_0
iteraz_backtracking = 0
while func(old_point + alpha*p) > func(old_point) + (sigma*alpha)*(p @
    gradient) \
and iteraz_backtracking < 100:
alpha = rho*alpha
iteraz_backtracking += 1

# Compute the new point and add it to the list of intermediate points
new_point = old_point + alpha*p
interm_points.append(new_point)

old_point = new_point

if k == maxit-1:
min_value = func(new_point)
conv = False

gradient = grad(new_point)
hessian = hess(new_point)
p = np_lin.solve(hessian, -gradient)
scalar_prod.append(gradient@p)

# Compute the 2norm of the difference between each intermediate point and
    the exact solution
error_x = [np_lin.norm(interm_x - sol_x) for interm_x in interm_points]

# Compute the difference between the function evaluated in each intermediate
    point and
# its value in the exact minimum point
error_f = [func(interm_x) - sol_f for interm_x in interm_points]

results = {'convergence': conv, 'k' : k, 'min_point' : new_point,
    'min_value' : min_value ,
        'interm_point' : interm_points, 'error_x' : error_x, 'error_f' :
            error_f,
        'scalar_product' : scalar_prod}

return results

```

2 Test Functions

dfbfd

(a)

The first function that we want to consider is the following:

$$f(x_1, x_2) = (x_1 - 2)^4 + (x_1 - 2)^2 x_2^2 + (x_2 + 1)^2. \tag{1}$$

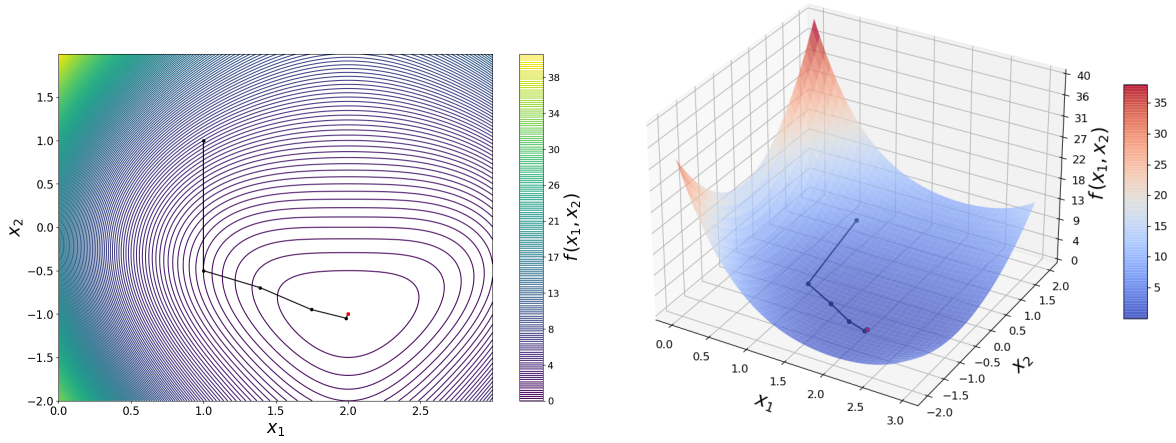
k	$\|\mathbf{x}_k - \mathbf{x}^*\|_2$	$f_1(\mathbf{x}_k) - f_1(\mathbf{x}^*)$	$-\nabla f_1(\mathbf{x}_k)^T[\nabla^2 f_1(\mathbf{x}_k)]^{-1}\nabla f_1(\mathbf{x}_k)$
0	2.236	6.000	-9.000
1	1.118	1.500	-1.761
2	6.805×10^{-1}	4.092×10^{-1}	-5.552×10^{-1}
3	2.592×10^{-1}	6.489×10^{-2}	-1.237×10^{-1}
4	5.012×10^{-2}	2.531×10^{-3}	-5.026×10^{-3}
5	1.277×10^{-3}	1.632×10^{-6}	-3.262×10^{-6}
6	1.659×10^{-6}	2.754×10^{-12}	-5.508×10^{-12}
7	1.404×10^{-12}	1.971×10^{-24}	-3.943×10^{-24}

Table 1: $x_0 = (1, 1)^T$, Backtracking = False

This function assumes a minimum value, equal to 0, in the point $(x_1^*, x_2^*) = (2, -1)$. We would like to obtain this minimum value by using the standard Newton algorithm implemented by the function ****Newton**** in the library ***Project_4.py ****. We report below the gradient and the Hessian of f, which must be passed to the function ****Newton****.

$$\nabla f(x_1, x_2) = \begin{bmatrix} 4(x_1 - 2)^3 + 2x_2^2(x_1 - 2) \\ 2x_2(x_1 - 2)^2 + 2x_2(x_2 - 2) \end{bmatrix}, \tag{2}$$

$$\nabla^2 f(x_1, x_2) = \begin{bmatrix} 12(x_1 - 2)^2 + 2x_2^2 & 4x_2(x_1 - 2) \\ 4x_2(x_1 - 2) & 2(x_1 - 2)^2 + 2 \end{bmatrix}. \tag{3}$$



(a) (b)

Figure 1: Contour plot for the function $f^{(a)}(x_1, x_2)$ where the intermediate points are obtained by using the standard Newton method.

2.1 (b)

$f : \mathbb{R}^4 \to \mathbb{R}$, $f(\mathbf{x}) = \mathbf{b}^T + \frac{1}{2}\mathbf{x}^T H \mathbf{x}$

$$\mathbf{b} = (5.04, -59.4, 146.4, -96.6)^T \tag{4}$$

$$H = \begin{bmatrix} 0.16 & -1.2 & 2.4 & -1.4 \\ -1.2 & 12.0 & -27.0 & 16.8 \\ 2.4 & -27.0 & 64.8 & -42.0 \\ -1.4 & 16.8 & -42.0 & 28.0 \end{bmatrix} \tag{5}$$

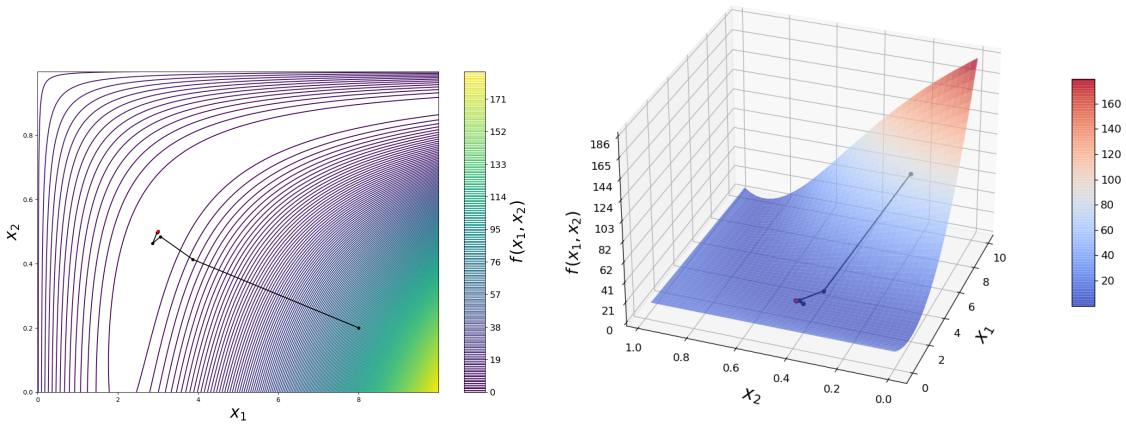
k	$\|\mathbf{x}_k - \mathbf{x}^*\|_2$	$f_1(\mathbf{x}_k) - f_1(\mathbf{x}^*)$	$-\nabla f_1(\mathbf{x}_k)^T[\nabla^2 f_1(\mathbf{x}_k)]^{-1}\nabla f_1(\mathbf{x}_k)$
0	5.009	8.17×10^{1}	-1.455×10^{2}
1	8.66×10^{-1}	2.423	-4.527
2	6.494×10^{-2}	2.407×10^{-2}	-4.643×10^{-2}
3	1.393×10^{-1}	3.45×10^{-3}	-6.32×10^{-3}
4	2.103×10^{-2}	1.383×10^{-4}	-2.704×10^{-4}
5	1.377×10^{-3}	2.863×10^{-7}	-5.717×10^{-7}
6	3.033×10^{-6}	2.186×10^{-12}	-4.372×10^{-12}
7	2.836×10^{-11}	1.233×10^{-22}	-2.466×10^{-22}
8	4.441×10^{-16}	4.437×10^{-31}	-8.79×10^{-31}

Table 2: $x_0 = (8, 0.2)^T$, backtracking = False

k	$\|\mathbf{x}_k - \mathbf{x}^*\|_2$	$f_1(\mathbf{x}_k) - f_1(\mathbf{x}^*)$	$-\nabla f_1(\mathbf{x}_k)^T[\nabla^2 f_1(\mathbf{x}_k)]^{-1}\nabla f_1(\mathbf{x}_k)$
0	5.745	5.223×10^{2}	-1.045×10^{3}
1	9.769×10^{-13}	2.842×10^{-14}	-9.948×10^{-27}
2	1.688×10^{-13}	0.000	-2.005×10^{-26}

2.2 (c)

$f : \mathbb{R}^2 \to \mathbb{R}$, $f(x_1, x_2) = (1.5 - x_1(1 - x_2))^2 + (2.25 - x_1(1 - x_2^2))^2 + (2.625 - x_1(1 - x_2^3))^2$



(a) (b)

Figure 2: Contour plot and 3D plot for the function $f^{(c)}(x_1, x_2)$ where the intermediate points are obtained by using the standard Newton method $x_0 = (0.75, -1.25)^T$.

k	$\|\mathbf{x}_k - \mathbf{x}^*\|_2$	$f_1(\mathbf{x}_k) - f_1(\mathbf{x}^*)$	$-\nabla f_1(\mathbf{x}_k)^T[\nabla^2 f_1(\mathbf{x}_k)]^{-1}\nabla f_1(\mathbf{x}_k)$
0	5.009	2.043	-3.291
1	3.963	2.553×10^{-1}	-5.885×10^{-2}
2	4.218	2.328×10^{-1}	-6.421×10^{-1}
3	1.661×10^{1}	5.743×10^{2}	-9.291×10^{2}
4	6.137	5.226×10^{1}	-6.327×10^{1}
5	3.426	1.596×10^{1}	-3.709
6	2.974	1.421×10^{1}	-8.445×10^{-3}
7	3.041	1.42×10^{1}	-5.688×10^{-8}
8	3.041	1.42×10^{1}	-1.083×10^{-18}
9	3.041	1.42×10^{1}	0.000

Table 3: $x_0 = (8, 0.8)^T$, backtracking = False *** PROBLEMA? ***

k	$\|\mathbf{x}_k - \mathbf{x}^*\|_2$	$f_1(\mathbf{x}_k) - f_1(\mathbf{x}^*)$	$-\nabla f_1(\mathbf{x}_k)^T[\nabla^2 f_1(\mathbf{x}_k)]^{-1}\nabla f_1(\mathbf{x}_k)$
0	5.009	2.043	-3.291
1	3.963	2.553×10^{-1}	-5.885×10^{-2}
2	4.218	2.328×10^{-1}	-6.421×10^{-1}
3	2.920	2.131×10^{-1}	-7.337×10^{-2}
4	2.471	1.649×10^{-1}	-1.224×10^{-1}
5	1.340	1.502×10^{-1}	-1.22×10^{-1}
6	1.192	7.989×10^{-2}	-1.697×10^{-1}
7	6.453×10^{-1}	5.012×10^{-2}	-5.027×10^{-2}
8	3.335×10^{-1}	1.73×10^{-2}	-2.418×10^{-2}
9	8.357×10^{-2}	3.009×10^{-3}	-5.269×10^{-3}
10	2.128×10^{-2}	1.004×10^{-4}	-1.967×10^{-4}
11	2.767×10^{-4}	1.503×10^{-7}	-3.005×10^{-7}
12	1.103×10^{-6}	2.293×10^{-13}	-4.585×10^{-13}
13	2.404×10^{-13}	8.166×10^{-25}	-1.633×10^{-24}
14	0.000	0.000	0.000

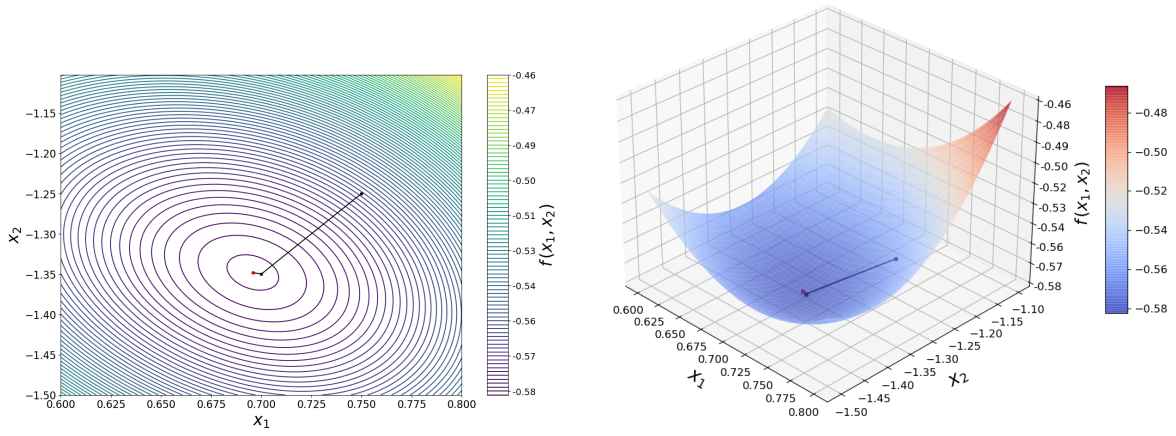
Table 4: $x_0 = (8, 0.8)^T$, backtracking = True

2.3 (d)

$f : \mathbb{R}^2 \to \mathbb{R}$, $f(x_1, x_2) = x_1^4 + x_1 x_2 + (1 + x_2)^2$

k	$\|\mathbf{x}_k - \mathbf{x}^*\|_2$	$f_1(\mathbf{x}_k) - f_1(\mathbf{x}^*)$	$-\nabla f_1(\mathbf{x}_k)^T[\nabla^2 f_1(\mathbf{x}_k)]^{-1}\nabla f_1(\mathbf{x}_k)$
0	1.119×10^{-1}	2.385×10^{-2}	-4.688×10^{-2}
1	4.601×10^{-3}	4.517×10^{-5}	-8.996×10^{-5}
2	2.951×10^{-5}	1.406×10^{-9}	-3.7×10^{-9}
3	3.805×10^{-9}	-4.436×10^{-10}	-6.372×10^{-18}

Table 5: $x_0 = (0.75, -1.25)^T$, $\alpha = 1$, backtracking=False

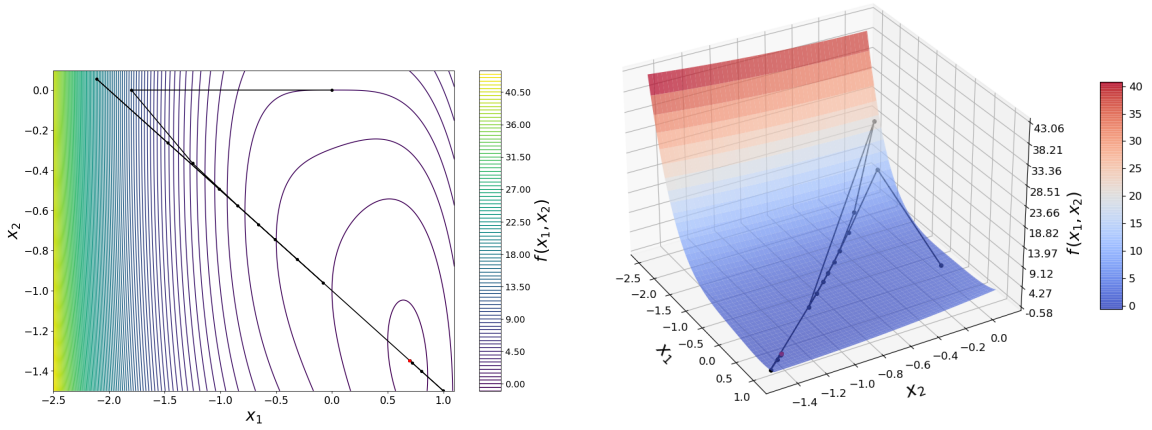


(a) (b)

Figure 3: Contour plot and 3D plot for the function $f^{(d)}(x_1, x_2)$ where the intermediate points and the minimum point are obtained by using the standard Newton method with $alpha = 1$. The starting point considered in this case is $x_0 = (0.75, -1.25)^T$.

k	$\|\mathbf{x}_k - \mathbf{x}^*\|_2$	$f_1(\mathbf{x}_k) - f_1(\mathbf{x}^*)$	$-\nabla f_1(\mathbf{x}_k)^T[\nabla^2 f_1(\mathbf{x}_k)]^{-1}\nabla f_1(\mathbf{x}_k)$
0	1.517	1.582	0.000
1	2.837	1.208×10^{1}	-1.432×10^{1}
2	2.181	3.888	-3.680
3	1.725	1.764	-1.114
4	1.352	1.099	-6.198×10^{-1}
5	8.656×10^{-1}	6.593×10^{-1}	2.174
6	3.138	2.143×10^{1}	-2.663×10^{1}
7	2.424	6.213	-6.657
8	1.910	2.391	-1.830
9	1.514	1.320	-6.987×10^{-1}
10	1.126	8.791×10^{-1}	-1.402
11	3.352×10^{-1}	3.219×10^{-1}	-5.271×10^{-1}
12	1.188×10^{-1}	3.348×10^{-2}	-6.1×10^{-2}
13	2.637×10^{-2}	1.514×10^{-3}	-2.957×10^{-3}
14	3.474×10^{-3}	2.572×10^{-5}	-5.128×10^{-5}
15	3.626×10^{-4}	2.789×10^{-7}	-5.586×10^{-7}
16	3.643×10^{-5}	2.375×10^{-9}	-5.637×10^{-9}
17	3.646×10^{-6}	-4.154×10^{-10}	-5.642×10^{-11}
18	3.659×10^{-7}	-4.434×10^{-10}	-5.643×10^{-13}

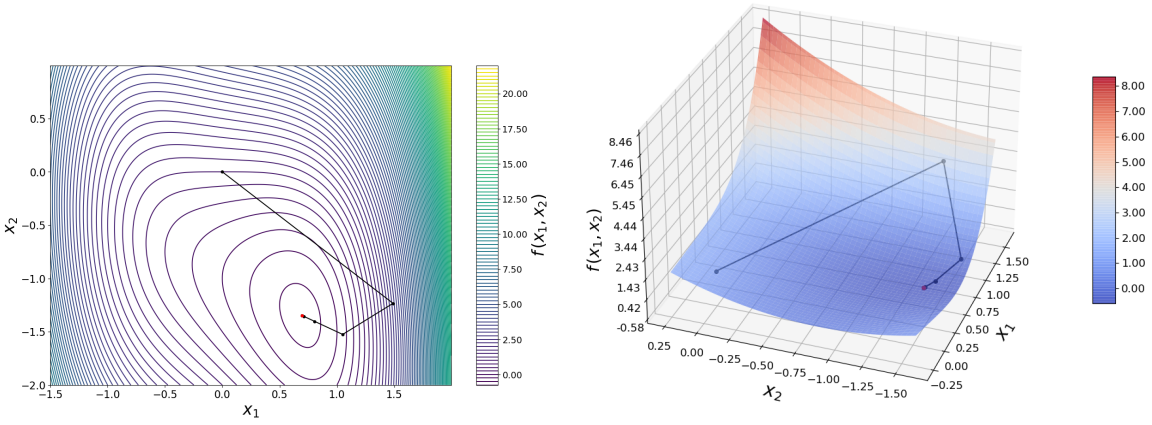
Table 6: Table of data obtained using the standard Newton algorithm to minimize the function $f^{(d)}(x_1, x_2)$ by using $\alpha = 0.9$ and by considering as starting point $x_0 = (0, 0)^T$.



(a) (b)

Figure 4: Contour plot and 3D plot for the function $f^{(d)}(x_1, x_2)$ where the intermediate points and the minimum point (in red) are obtained by using the standard Newton method with $\alpha = 0.9$. The staring point considered in this case is $x_0 = (0, 0)^T$.

k	$\|\mathbf{x}_k - \mathbf{x}^*\|_2$	$f_1(\mathbf{x}_k) - f_1(\mathbf{x}^*)$	$-\nabla f_1(\mathbf{x}_k)^T [\nabla^2 f_1(\mathbf{x}_k)]^{-1} \nabla f_1(\mathbf{x}_k)$
0	1.517	1.582	0.000
1	8.014×10^{-1}	3.716	-5.554
2	3.959×10^{-1}	4.724×10^{-1}	-7.576×10^{-1}
3	1.232×10^{-1}	3.61×10^{-2}	-6.559×10^{-2}
4	1.717×10^{-2}	6.364×10^{-4}	-1.253×10^{-3}
5	4.011×10^{-4}	3.414×10^{-7}	-6.834×10^{-7}
6	2.275×10^{-7}	-4.435×10^{-10}	-2.171×10^{-13}
7	3.061×10^{-9}	-4.436×10^{-10}	-2.197×10^{-26}



(a) (b)

Figure 5: Contour plot and 3D plot of the function $f^{(d)}(x_1, x_2)$ where the intermediate points and the minimum point (in red) are obtained by using the Newton method with the trust region approach. The starting point considered in this case is $x_0 = (0, 0)^T$.