Spiral Optimization Algorithm

SK5001 Advance Numerical Analysis

Aria Wahyu Wicaksono- 10219083

In this assignment, all numerical experiments were performed on Intel Core i5 4210U with 8GB RAM running Windows 10. We will use spiral optimization algorithm to find roots of two-dimensional system using this objective or merit function:

$$F(\boldsymbol{x}) = \frac{1}{1 + |f_1(\boldsymbol{x})| + |f_2(\boldsymbol{x})|}$$

which has the maximum value of 1 iff $\boldsymbol{x}$ is zero of f_1 and f_2.

The main code: (**"./soa.py"**)

```
import numpy as np
import time

roots = []
f_root = [] # array to store f1, f2, and f value of roots
iteration = 0
start_time = time.time_ns() // 1_000_000

while len(roots) < num_roots:
    iteration += 1
    # initialize m random points that lies in the domain
    x = np.array([np.array([
            np.random.uniform(low=x1_lim[0], high=x1_lim[1]),
            np.random.uniform(low=x2_lim[0], high=x2_lim[1])])
            for _ in range(m)])
    x_star = x[np.argmax(f(x.T))]
    for _ in range(k_max):
        x = np.dot(S_2,x.T).T - np.dot((S_2 - np.diag([1,1])), x_star)
        # mask is boolean of each element of x that lies in the domain
        mask = np.where((x[:,0] >= x1_lim[0]) & (x[:,0] <= x1_lim[1])
                        & (x[:,1] >= x2_lim[0]) & (x[:, 1] <= x2_lim[1]))
        # select only x that lies in the domain
        x_def = x[mask]
        x_star = x_def[np.argmax(f(x_def.T))]
    # check if roots array is still empty
    if len(roots) == 0:
        roots = np.array([x_star])
        continue
    for i, root in enumerate(roots):
        # check if x_star already in array
        if np.all(np.isclose(x_star, root, atol=d, rtol=0)):
            break
        # x_star is not in roots array,
        # check if i is last index of roots array
        elif i == roots.shape[0] - 1:
            roots = np.append(roots, [x_star], axis=0)

end_time = time.time_ns() // 1_000_000
```

```
print(f"Roots: \n{roots}")
print(f"Number of iteration to find all roots: {iteration}")
for root in roots: f_root.append([f_1(*root), f_2(*root), f(root)])
print("f1, f2, and merit function value:")
for f_val in f_root: print(f_val)
print(f"Execution time for {iteration} iterations: {end_time-
    start_time}ms")
```

Code for plotting: (**"./plot.py"**)

```
import matplotlib.pyplot as plt
import numpy as np
import time
from soa import f_1, f_2, title, roots, x1_lim, x2_lim

x_1 = np.linspace(*x1_lim, 200)
x_2 = np.linspace(*x2_lim, 200)
a, b = np.meshgrid(x_1, x_2)
fig, ax = plt.subplots(1,1, figsize=(8,8))
ax.set_xlabel(r"$x_1$")
ax.set_ylabel(r"$x_2$")
ax.set_title(title)
ax.contour(a, b, f_1(a,b), [0], colors='b') # blue
ax.contour(a, b, f_2(a,b), [0], colors='k') # black
ax.scatter(*roots.T, marker="X", color="red")
plt.grid(True)
plt.show()
```

To test the code, we are using this following system of equations:

Problem 1:

$$f_1(x_1, x_2) = e^{x_1 - x_2} - \sin(x_1 + x_2) = 0$$

$$f_2(x_1, x_2) = x_1^2 x_2^2 - \cos(x_1 + x_2) = 0$$

with

$$D = \{(x_1, x_2) : -10 \leq x_1 \leq 10, -10 \leq x_2 \leq 10\}$$

Using this following parameter values:

$$m = 50, k_{max} = 250, r = 0.95, \theta = \frac{\pi}{4}, d = 10^{-2}, num_{roots} = 6$$

The problem initial configuration code: (**"./problems/problem1.py"**)

```
import numpy as np

title = "Problem 1 Contour Plot"

def f_1(x_1, x_2):
    return np.exp(x_1-x_2) - np.sin(x_1+x_2)

def f_2(x_1, x_2):
    return x_1**2*x_2**2-np.cos(x_1+x_2)
def f(x):
```

```
11.     return 1/(1+abs(f_1(*x))+abs(f_2(*x)))
12.
13. d = 1e-2 # the least distance between each variable of roots
14. num_roots = 6 # number of roots that we want to find
15. k_max = 250
16. m = 50
17. r = 0.95
18. theta = np.pi/4
19. x1_lim = [-10, 10]
20. x2_lim = [-10, 10]
21. R = np.array([[np.cos(theta), -np.sin(theta)],[np.sin(theta),
    np.cos(theta)]])
22. S_2 = np.dot(np.diag([r,r]), R)
```

We add the following code in the top of main code: (**"/.soa.py"**)

```
1. from problems.problem1 import x1_lim, x2_lim, f_1, f_2, f, \
2.                               k_max, m, S_2, d, num_roots, \
3.                               title
```

Problem 1 results:

Solution	x_1	x_2	$f_1(x_1, x_2)$	$f_2(x_1, x_2)$	$F(x_1, x_2)$
1	-6.43716	0.155348	1.964e-06	4.267e-06	0.999992
2	-6.11711	-0.16348	-2.915e-07	-1.182e-06	0.999997
3	-0.93212	1.067873	2.186e-06	-7.228e-07	0.999996
4	-0.15528	6.439841	-5.430e-06	-2.323e-06	0.999999
5	0.163334	6.122434	-1.505e-08	2.611e-06	0.999994
6	0.667121	0.690104	-1.543e-06	2.014e-06	0.999997

Number of iteration to find all roots: 19
Execution time for 19 iterations: 305ms (0.305 s)

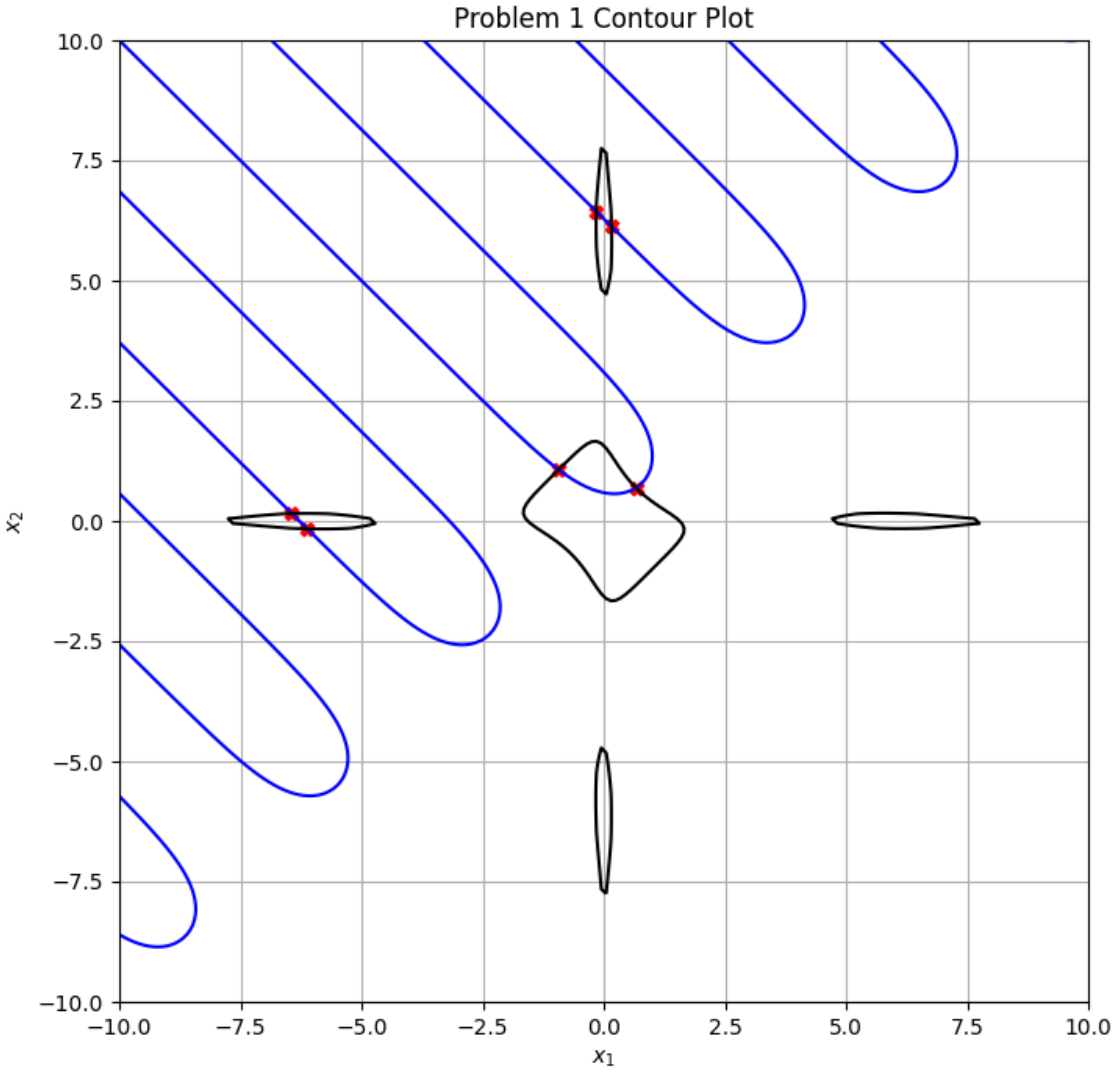

Figure 1. Contour plot for system of equations in problem 1 (f_1 in blue, f_2 in black)

From the above results, we found all 6 roots after 19 iterations (305 ms). The number of iterations and the execution time vary depending on how random numbers generated. The f_1 and f_2 values for each solution are really close to zero (and the merit function also really close to its maximum value, one), so we can conclude that the solutions are indeed the roots of f_1 and f_2.

Problem 2:

$$f_1(x_1, x_2) = 0.5\sin(x_1 x_2) - 0.25\frac{x_2}{\pi} - 0.5x_1 = 0$$

$$f_2(x_1, x_2) = \left(1 - \frac{0.25}{\pi}\right)(e^{2x_1} - e) + e\frac{x_2}{\pi} - 2ex_1 = 0$$

with

$$D = \{(x_1, x_2) : -1 \leq x_1 \leq 3, -17 \leq x_2 \leq 4\}$$

Using this following parameter values:

$$m = 400, k_{max} = 100, r = 0.95, \theta = \frac{\pi}{4}, d = 10^{-1}, num_{roots} = 12$$

The problem initial configuration code: (**"./problems/problem2.py"**)

```
1. import numpy as np
2.
3. title = "Problem 2 Contour Plot"
4.
5. def f_1(x_1, x_2):
6.     return 0.5*np.sin(x_1*x_2) - 0.25*x_2/np.pi - 0.5*x_1
7.
8. def f_2(x_1, x_2):
9.     return (1-0.25/np.pi)*(np.exp(2*x_1)-np.exp(1))+np.exp(1)*x_2/np.pi -
   2*np.exp(1)*x_1
10. def f(x):
11.     return 1/(1+abs(f_1(*x))+abs(f_2(*x)))
12.
13. d = 1e-1 # the least distance between each variable of roots
14. num_roots = 12 # number of roots that we want to find
15. k_max = 100
16. m = 400
17. r = 0.95
18. theta = np.pi/4
19. x1_lim = [-1, 3]
20. x2_lim = [-17, 4]
21. R = np.array([[np.cos(theta), -np.sin(theta)],[np.sin(theta),
    np.cos(theta)]])
22. S_2 = np.dot(np.diag([r,r]), R)
23.
```

We add the following code in the top of main code: (**"/.soa.py"**)

```
1. from problems.problem2 import x1_lim, x2_lim, f_1, f_2, f, \
2.                               k_max, m, S_2, d, num_roots, \
3.                               title
```

Problem 2 results:

Solution	x_1	x_2	$f_1(x_1, x_2)$	$f_2(x_1, x_2)$	$F(x_1, x_2)$
1	-0.26045	0.623712	-2.752e-04	3.584e-04	0.99915
2	0.299107	2.835931	-1.684e-04	-1.505e-04	0.99723
3	0.500692	3.141667	-3.532e-04	-2.326e-04	0.99910
4	1.294197	-3.13744	3.056e-05	-3.299e-03	0.99875
5	1.337394	-4.14168	-4.490e-04	-1.749e-03	0.99960
6	1.433893	-6.82096	-8.706e-06	-1.698e-03	0.99918
7	1.481341	-8.38259	5.657e-04	1.535e-03	0.99778
8	1.530607	-10.2024	5.974e-04	3.326e-03	0.99708
9	1.578202	-12.1795	-1.677e-03	-3.295e-03	0.99845
10	1.604483	-13.3587	-3.628e-03	1.043e-04	0.99658
11	1.654568	-15.8173	7.122e-04	1.010e-03	0.99590
12	1.663559	-16.2911	3.658e-03	-9.660e-04	0.99463

Number of iteration to find all roots: 157
Execution time for 157 iterations: 2108 ms (2.108 s)

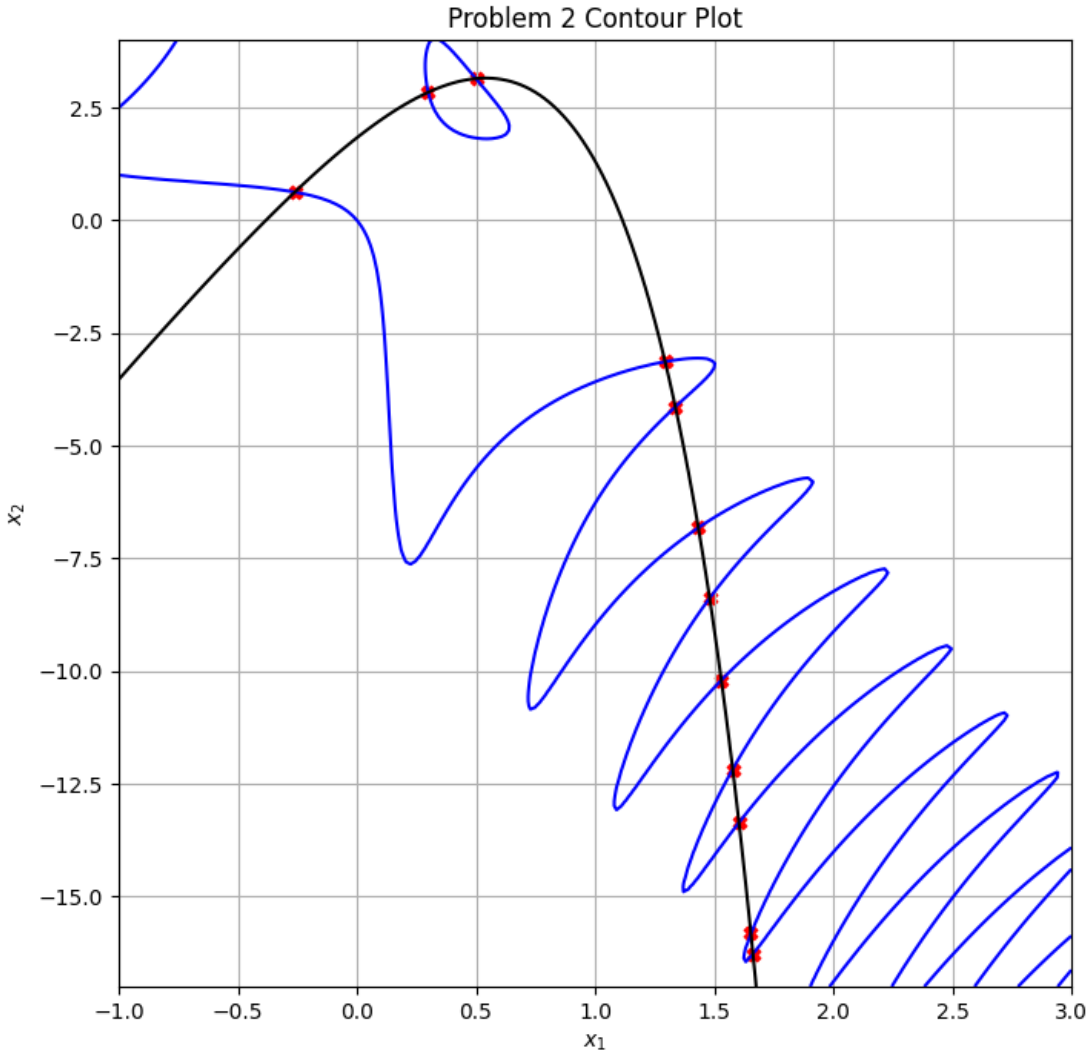

Figure 2. Contour plot for system of equations in problem 2 (f_1 in blue, f_2 in black)

From the above results, we found all 12 roots after 157 iterations (2.108 s). The number of iterations and the execution time vary depending on how random numbers generated. The f_1 and f_2 values for each solution are really close to zero (and the merit function also really close to its maximum value, one), although it is in the order of 10^{-3}. To increase the accuracy, we can increase m and k_{max} parameters, although we will have longer execution time as a result.

Problem 3:

$$f_1(x_1, x_2) = \sin(4\pi x_1 x_2) - (x_1 + x_2) = 0$$

$$f_2(x_1, x_2) = \left(4 - \frac{1}{4\pi}\right)(e^{2x_1} - e) + 4ex_2 - 2ex_1 = 0$$

with

$$D = \{(x_1, x_2) : -2 \leq x_1 \leq 1, -1 \leq x_2 \leq 1\}$$

Using this following parameter values:

$$m = 100, k_{max} = 150, r = 0.95, \theta = \frac{\pi}{4}, d = 10^{-2}, num_{roots} = 10$$

The problem initial configuration code: (**"./problems/problem3.py"**)

```
1. import numpy as np
2.
3. title = "Problem 3 Contour Plot"
4.
5. pi = np.pi
6. e = np.exp(1)
7.
8. def f_1(x_1, x_2):
9.     return np.sin(4*pi*x_1*x_2)-(x_1 + x_2)
10.
11. def f_2(x_1, x_2):
12.     return (4-1/(4*pi))*(np.exp(2*x_1)-e)+4*e*x_2**2-2*e*x_1
13.
14. def f(x):
15.     return 1/(1+abs(f_1(*x))+abs(f_2(*x)))
16.
17. d = 1e-2 # the least distance between each variable of roots
18. num_roots = 10 # number of roots that we want to find
19. k_max = 150
20. m = 100
21. r = 0.95
22. theta = np.pi/4
23. x1_lim = [-2, 1]
24. x2_lim = [-1, 1]
25. R = np.array([[np.cos(theta), -np.sin(theta)],[np.sin(theta),
    np.cos(theta)]])
26. S_2 = np.dot(np.diag([r,r]), R)
```

We add the following code in the top of main code: (**"/.soa.py"**)

```
1. from problems.problem3 import x1_lim, x2_lim, f_1, f_2, f, \
2.                               k_max, m, S_2, d, num_roots, \
3.                               title
```

Problem 3 results:

Solution	x_1	x_2	$f_1(x_1, x_2)$	$f_2(x_1, x_2)$	$F(x_1, x_2)$
1	-1.16806	0.601012	-6.992e-05	4.798e-05	0.999882
2	-0.66421	0.743291	-9.430e-06	-8.662e-05	0.999904
3	-0.36104	0.790205	-8.040e-06	-2.374e-04	0.999755
4	0.078973	-0.77286	-8.011e-05	-2.782e-04	0.999642
5	0.111054	0.765112	-4.921e-05	-2.594e-05	0.999925
6	0.207852	0.733229	2.124e-04	-5.110e-06	0.999783
7	0.335711	-0.66508	-1.802e-04	-2.480e-05	0.999795
8	0.499972	-0.50003	1.013e-04	-7.087e-05	0.999828
9	0.604529	0.272703	-6.014e-05	-7.672e-05	0.999863
10	0.636379	0.104303	1.474e-05	3.651e-04	0.99962

Number of iteration to find all roots: 22
Execution time for 22 iterations: 302ms (0.302 s)

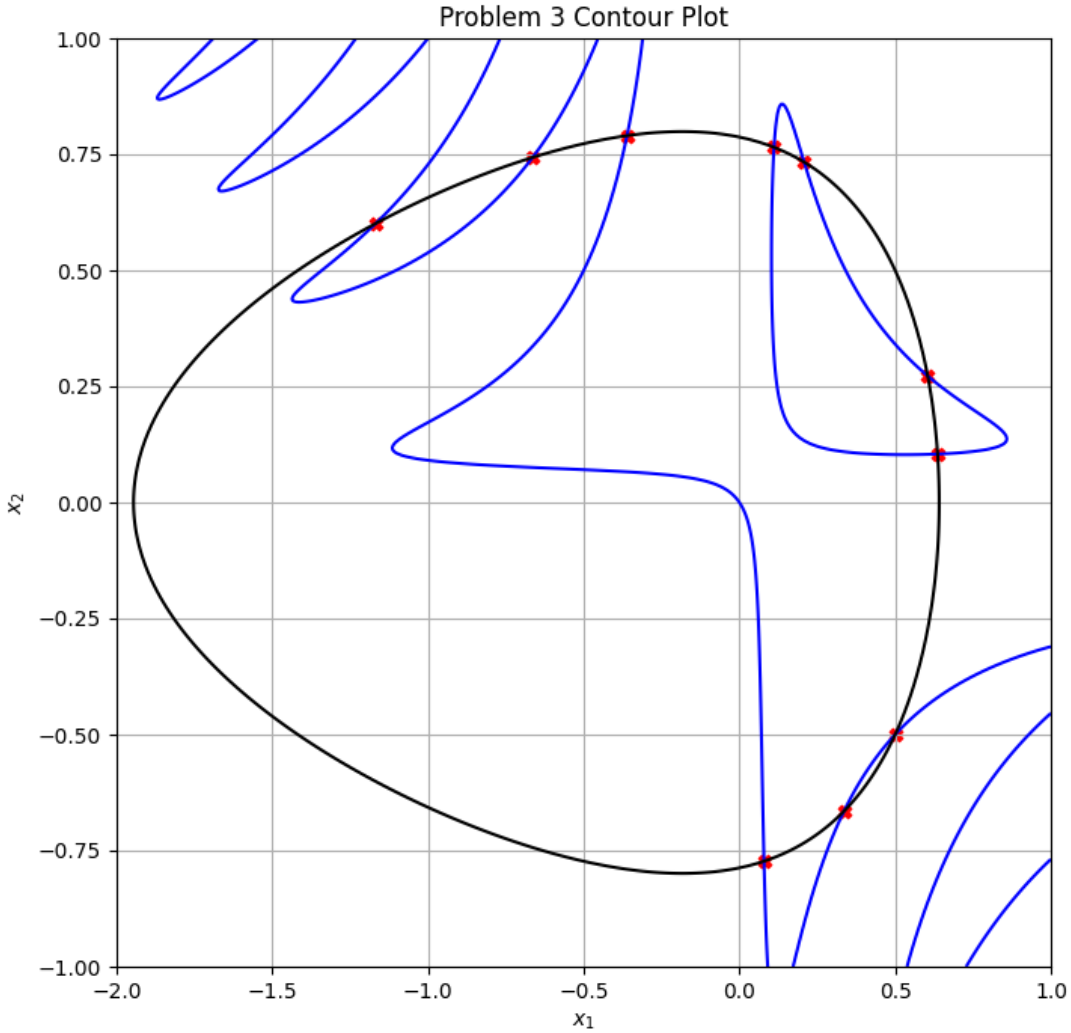

Figure 3. Contour plot for system of equations in problem 3 (f_1 in blue, f_2 in black)

From the above results, we found all 10 roots after 22 iterations ($0.302\ s$). The number of iterations and the execution time vary depending on how random numbers generated. The f_1 and f_2 values for each solution are really close to zero (and the merit function also really close to its maximum value, one), so we can conclude that the solutions are indeed the roots of f_1 and f_2.

References:

K. Sidarto and A. Kania, "Finding All Solutions of Systems of Nonlinear Equations Using Spiral Dynamics Inspired Optimization with Clustering," J. Adv. Comput. Intell. Intell. Inform., Vol.19, No.5, pp. 697-707, 2015.

K. Tamura and K. Yasuda, "Spiral Dynamics Inspired Optimization," J. Adv. Comput. Intell. Intell. Inform., Vol.15, No.8, pp. 1116-1122, 2011.

R. L. Burden and J. D. Faires, "Numerical Analysis," 9th ed., Brooks/Cole, 2010.

References from Website:

Matplotlib Development Team, "matplotlib.pyplot.contour". Matplotlib 3.6.2 Documentation, Retrieved November 17, 2022, from https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.contour.html

Numpy Developer, "Linear Algebra (numpy.linalg)". Linear algebra (numpy.linalg) - NumPy v1.23 Manual, Retrieved November 17, 2022, from https://numpy.org/doc/stable/reference/routines.linalg.html