Лабораторная работа №2

Университет ИТМО

Факультет систем управления и робототехники

Выполнили:

- **Соколов Никита**
- **Минеев Кирилл**
- **Соловьев Дмитрий**
- **Семёнов Евгений**

Проверил:

- **Догадин Егор Витальевич**

---

7 октября 2024 г.

Подготовка.

Установка необходимых библиотек:

In [1]:
```
!pip install numpy matplotlib sympy
```

```
Requirement already satisfied: numpy in /Users/nvoron/Documents/GitHub/Num
erical-methods/venv/lib/python3.8/site-packages (1.24.4)
Requirement already satisfied: matplotlib in /Users/nvoron/Documents/GitHu
b/Numerical-methods/venv/lib/python3.8/site-packages (3.7.5)
Requirement already satisfied: sympy in /Users/nvoron/Documents/GitHub/Num
erical-methods/venv/lib/python3.8/site-packages (1.13.3)
Requirement already satisfied: importlib-resources>=3.2.0; python_version
< "3.10" in /Users/nvoron/Documents/GitHub/Numerical-methods/venv/lib/pyth
on3.8/site-packages (from matplotlib) (6.4.5)
Requirement already satisfied: fonttools>=4.22.0 in /Users/nvoron/Document
s/GitHub/Numerical-methods/venv/lib/python3.8/site-packages (from matplotl
ib) (4.54.1)
Requirement already satisfied: pillow>=6.2.0 in /Users/nvoron/Documents/Gi
tHub/Numerical-methods/venv/lib/python3.8/site-packages (from matplotlib)
(10.4.0)
Requirement already satisfied: contourpy>=1.0.1 in /Users/nvoron/Documents
/GitHub/Numerical-methods/venv/lib/python3.8/site-packages (from matplotli
b) (1.1.1)
Requirement already satisfied: kiwisolver>=1.0.1 in /Users/nvoron/Document
s/GitHub/Numerical-methods/venv/lib/python3.8/site-packages (from matplotl
ib) (1.4.7)
Requirement already satisfied: python-dateutil>=2.7 in /Users/nvoron/Docum
ents/GitHub/Numerical-methods/venv/lib/python3.8/site-packages (from matpl
otlib) (2.9.0.post0)
Requirement already satisfied: pyparsing>=2.3.1 in /Users/nvoron/Documents
/GitHub/Numerical-methods/venv/lib/python3.8/site-packages (from matplotli
b) (3.1.4)
Requirement already satisfied: cycler>=0.10 in /Users/nvoron/Documents/Git
Hub/Numerical-methods/venv/lib/python3.8/site-packages (from matplotlib) (
0.12.1)
Requirement already satisfied: packaging>=20.0 in /Users/nvoron/Documents/
GitHub/Numerical-methods/venv/lib/python3.8/site-packages (from matplotlib
) (24.1)
Requirement already satisfied: mpmath<1.4,>=1.1.0 in /Users/nvoron/Documen
ts/GitHub/Numerical-methods/venv/lib/python3.8/site-packages (from sympy)
(1.3.0)
Requirement already satisfied: zipp>=3.1.0; python_version < "3.10" in /Us
ers/nvoron/Documents/GitHub/Numerical-methods/venv/lib/python3.8/site-pack
ages (from importlib-resources>=3.2.0; python_version < "3.10"->matplotlib
) (3.20.2)
Requirement already satisfied: six>=1.5 in /Users/nvoron/Documents/GitHub/
Numerical-methods/venv/lib/python3.8/site-packages (from python-dateutil>=
2.7->matplotlib) (1.16.0)
WARNING: You are using pip version 19.2.3, however version 24.2 is availab
le.
You should consider upgrading via the 'pip install --upgrade pip' command.
```

In [2]:
```
import numpy as np
import matplotlib.pyplot as plt
from random import randint
import math
```

Придумаем четыре целых числа a, b, c, d таким образом,

чтобы все они были различными и ни одно из них не равнялось 0 или ±1.

In [3]:
```
l = []
while True:
    k = randint(2, 20)
    if (k not in [-1, 0, 1]) and (k not in l):
        l.append(k)
    if len(l) == 4:
        break
a, b, c, d = l
print(a, b, c, d, sep = "  ")
```

```
13  20  19  6
```

Исходные значения x и y:

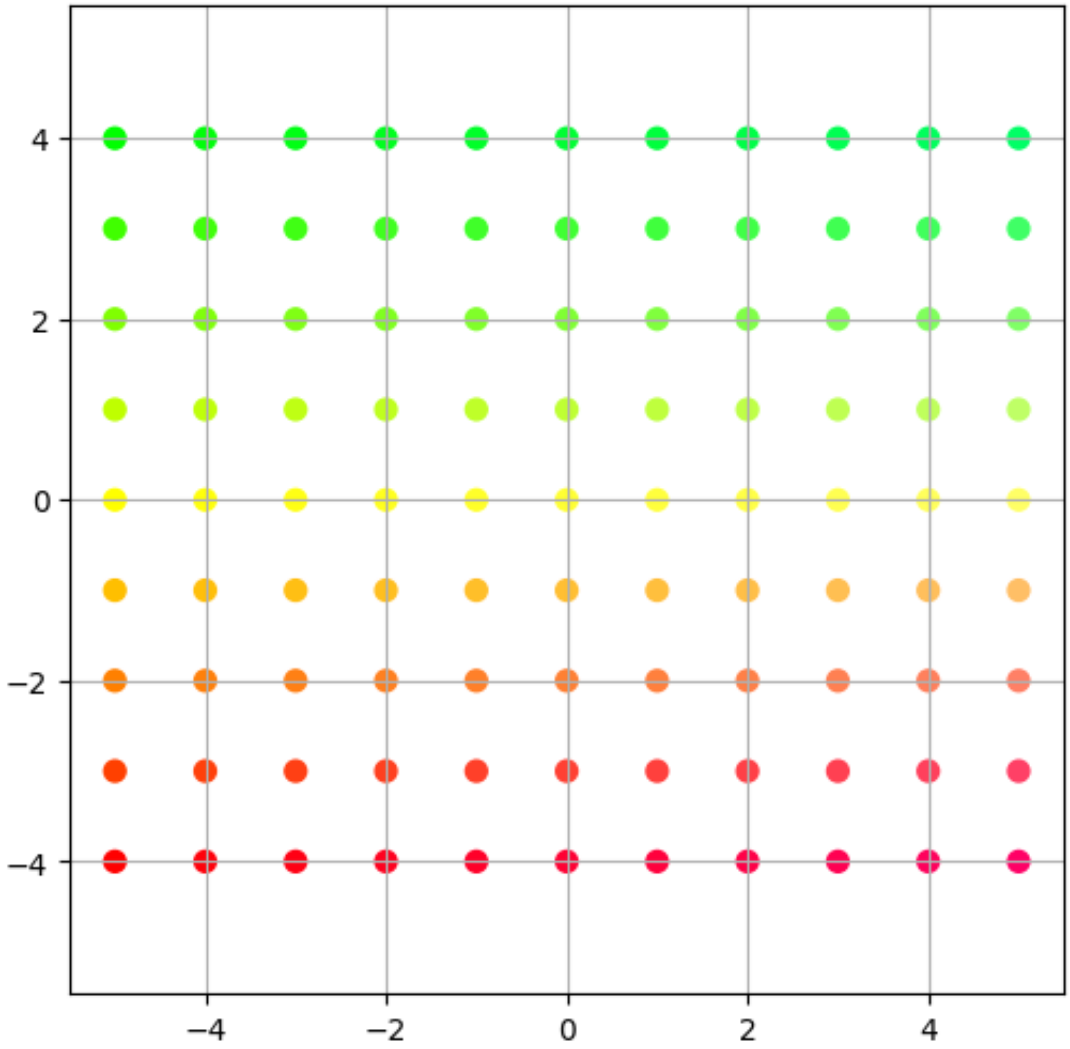
In [4]:
```
x_min, x_max = -5, 5
y_min, y_max = -4, 4

x_old = np.linspace(x_min, x_max, 11)
y_old = np.linspace(y_min, y_max, 9)
xy_old = np.column_stack([[x, y] for x in x_old for y in y_old])
```

Для визуализации линейных отображений:

In [5]:
```
colors = list(map(lambda x, y: (min(1, 1-y/4), min(1, 1+y/4), 1/5 + x/25)
```

Исходная сетка значений:

In [6]:
```
plt.figure(figsize=(6, 6), facecolor="w")
plt.scatter(xy_old[0], xy_old[1], s = 50, c = colors)
plt.axis('equal')
plt.grid(True)
```

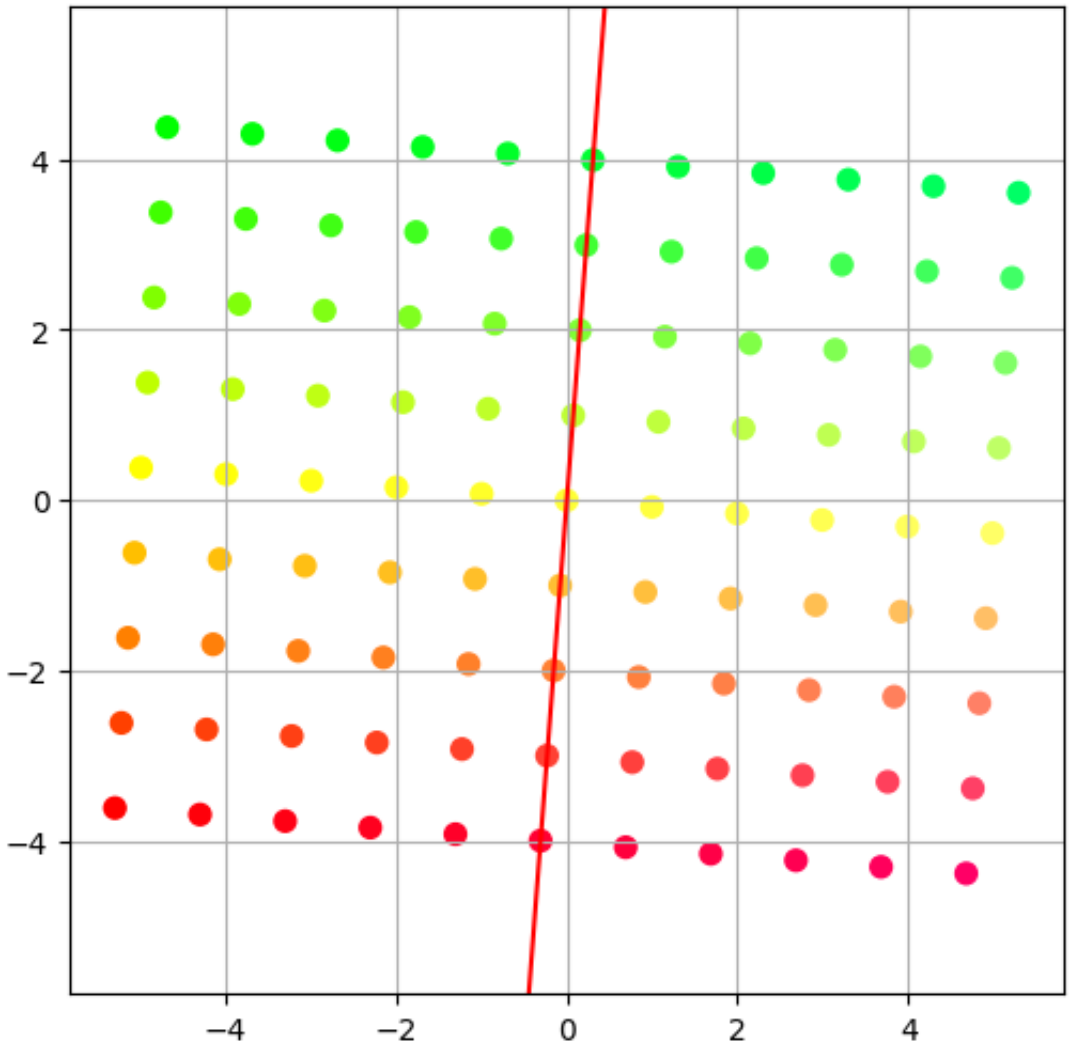


Задание 1.

1. Отражение (симметрия) плоскости относительно прямой y = ax:

In [7]:
```
theta_angle = np.arctan(a) - math.pi / 2

first_matrix = np.array([
    [np.cos(theta_angle), -np.sin(theta_angle)],
    [np.sin(theta_angle), np.cos(theta_angle)]
])

xy_new = np.dot(first_matrix, xy_old)
plt.figure(figsize=(6, 6), facecolor="w")
plt.scatter(xy_new[0], xy_new[1], s = 50, c = colors)
plt.axis('equal')
plt.grid(True)

plt.plot(x_old, a * x_old, color="red")
```

```
plt.ylim(y_min, y_max)
plt.show()
print(first_matrix)
```

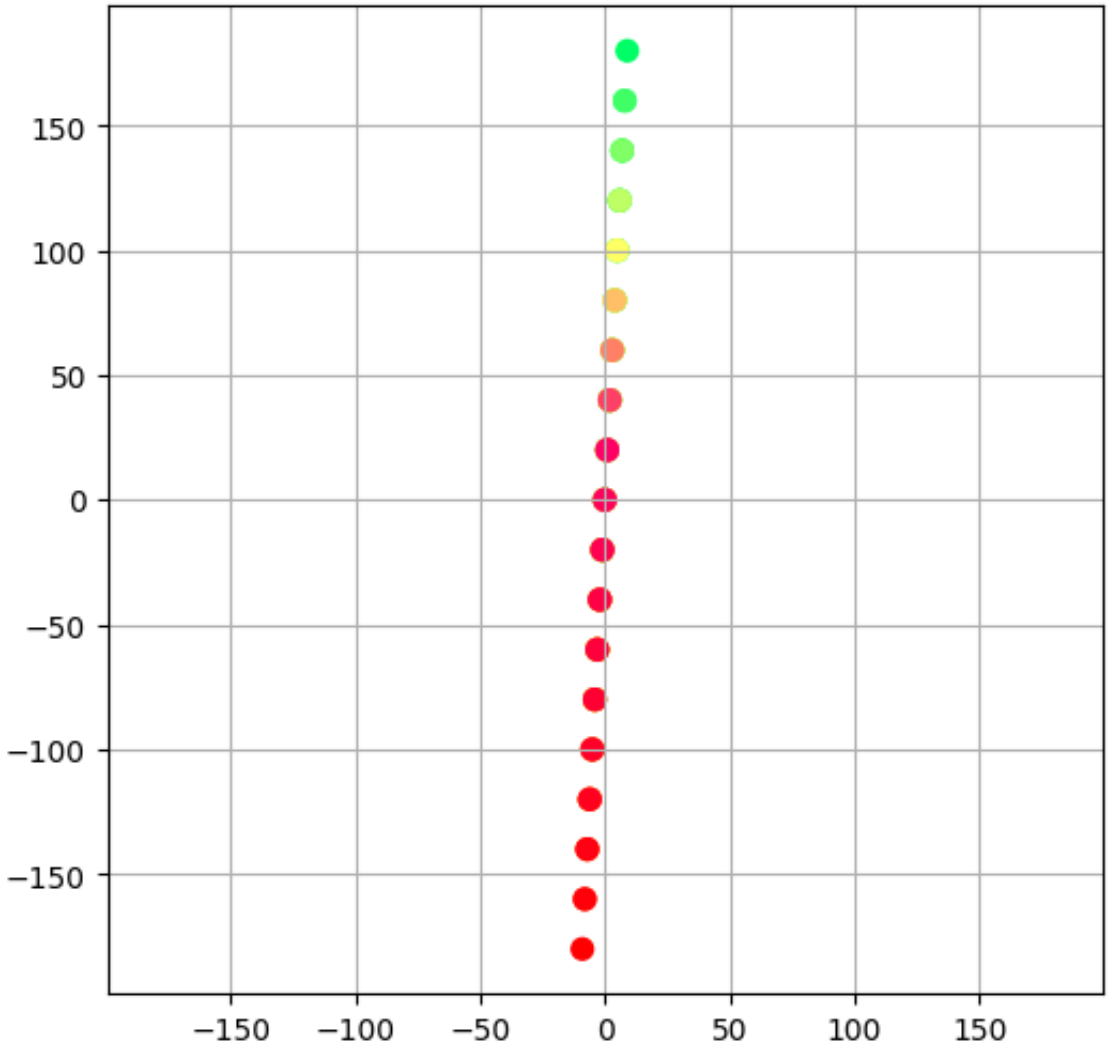



```
[[ 0.99705449  0.0766965 ]
 [-0.0766965   0.99705449]]
```

2. Отображение всей плоскости в прямую y = bx:

In [8]:
```
second_matrix = np.array([
    [1, 1],
    [b, b]
])
xy_new = np.dot(second_matrix, xy_old)

plt.figure(figsize=(6, 6), facecolor="w")
plt.scatter(xy_new[0], xy_new[1], s = 50, c = colors)
plt.axis('equal')
plt.grid(True)
print(second_matrix)
```

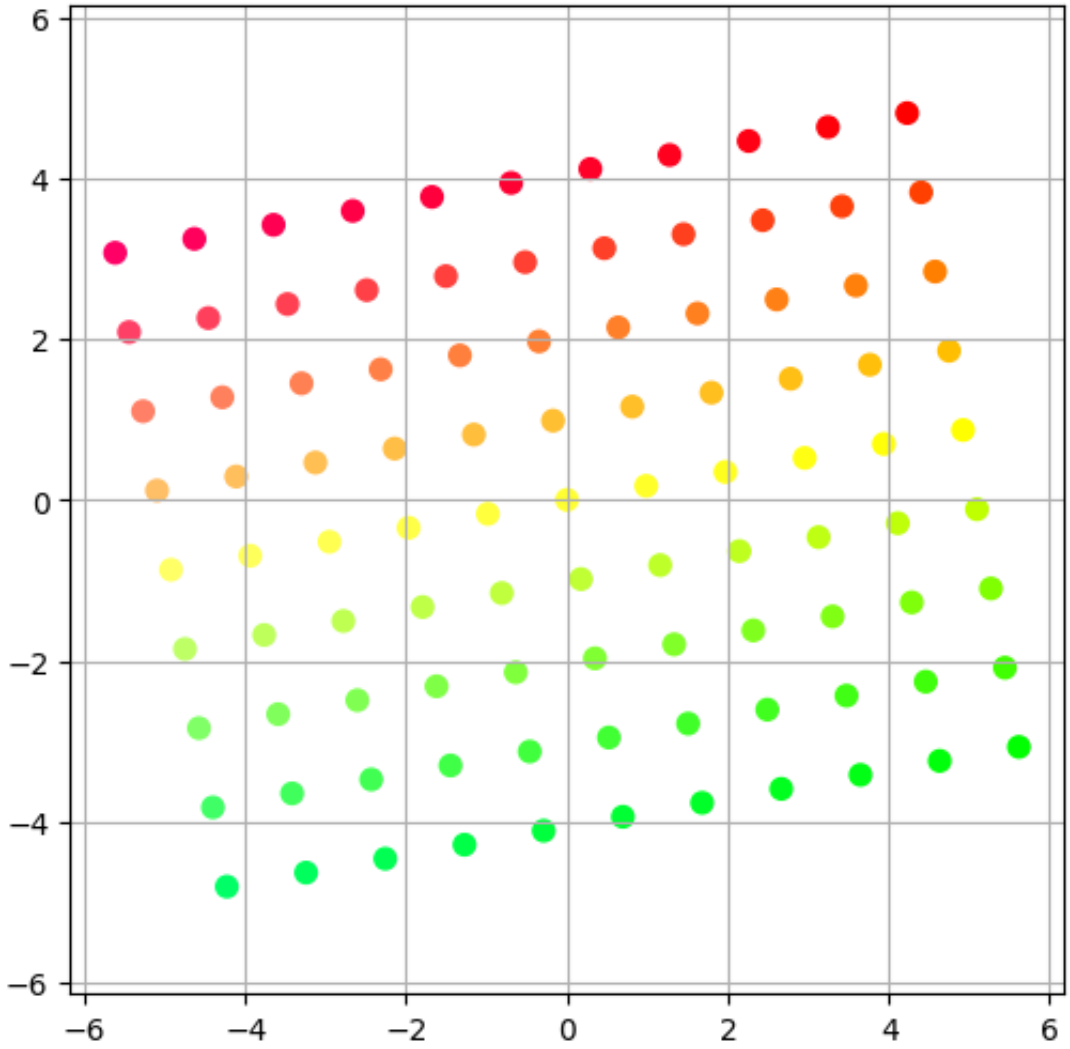
```
[[ 1  1]
 [20 20]]
```



3. Поворот плоскости на 10c градусов против часовой стрелки:

In [9]:
```
alpha_angle = np.radians(10 * c)

third_matrix = np.array([
    [np.cos(alpha_angle), -np.sin(alpha_angle)],
    [np.sin(alpha_angle), np.cos(alpha_angle)]
])

xy_new = np.dot(third_matrix, xy_old)

plt.figure(figsize=(6, 6), facecolor="w")
plt.scatter(xy_new[0], xy_new[1], s = 50, c = colors)
plt.axis('equal')
plt.grid(True)
print(third_matrix)
```

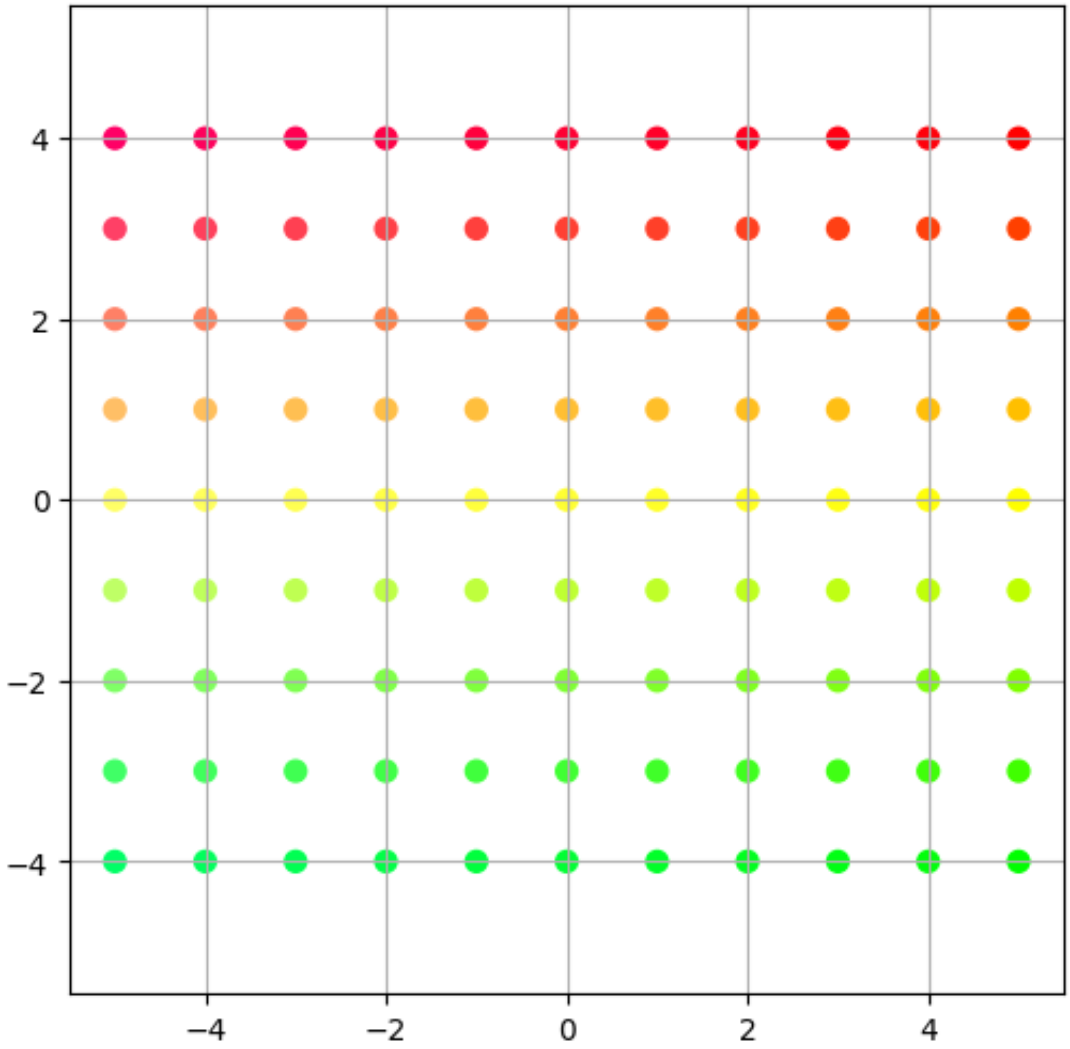
```
[[-0.98480775  0.17364818]
 [-0.17364818 -0.98480775]]
```



4. Центральная симметрия плоскости относительно начала координат:

In [10]:
```
beta_angle = math.pi

fourth_matrix = np.array([
    [np.cos(beta_angle), -np.sin(beta_angle)],
    [np.sin(beta_angle), np.cos(beta_angle)]
])

xy_new = np.dot(fourth_matrix, xy_old)

plt.figure(figsize=(6, 6), facecolor="w")
plt.scatter(xy_new[0], xy_new[1], s = 50, c = colors)
plt.axis('equal')
plt.grid(True)
print(fourth_matrix)
```

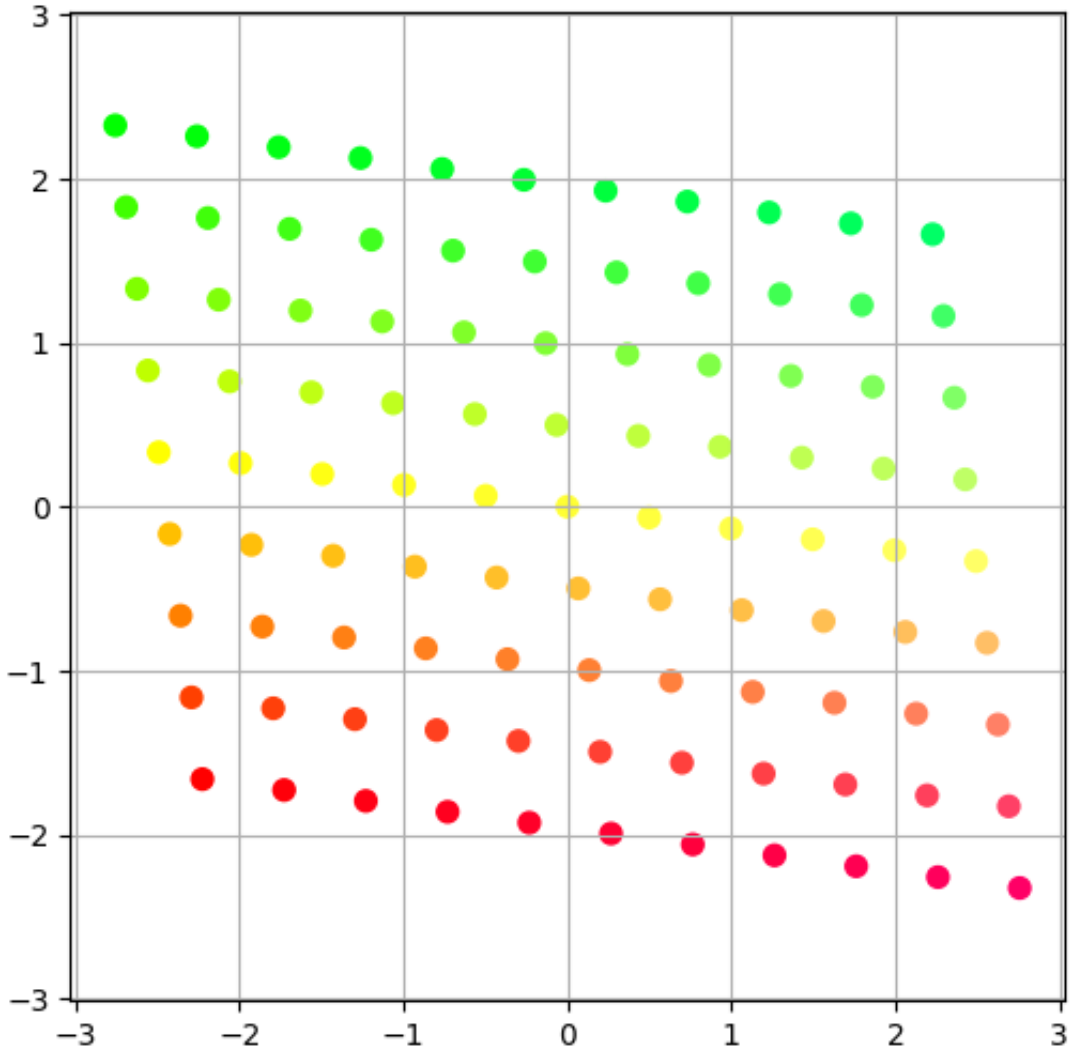
```
[[-1.0000000e+00 -1.2246468e-16]
 [ 1.2246468e-16 -1.0000000e+00]]
```



5. Отображение, которое можно описать так: сначала отражение относительно прямой y = ax, потом поворот на 10d градусов по часовой стрелке:

In [11]:
```
gamma_angle = np.radians(10 * d)

fifth_matrix = np.array([
    [np.cos(gamma_angle), -np.sin(gamma_angle)],
    [np.sin(gamma_angle), np.cos(gamma_angle)]
]) * first_matrix

xy_new = np.dot(fifth_matrix, xy_old)

plt.figure(figsize=(6, 6), facecolor="w")
plt.scatter(xy_new[0], xy_new[1], s = 50, c = colors)
plt.axis('equal')
plt.grid(True)
print(fifth_matrix)
```

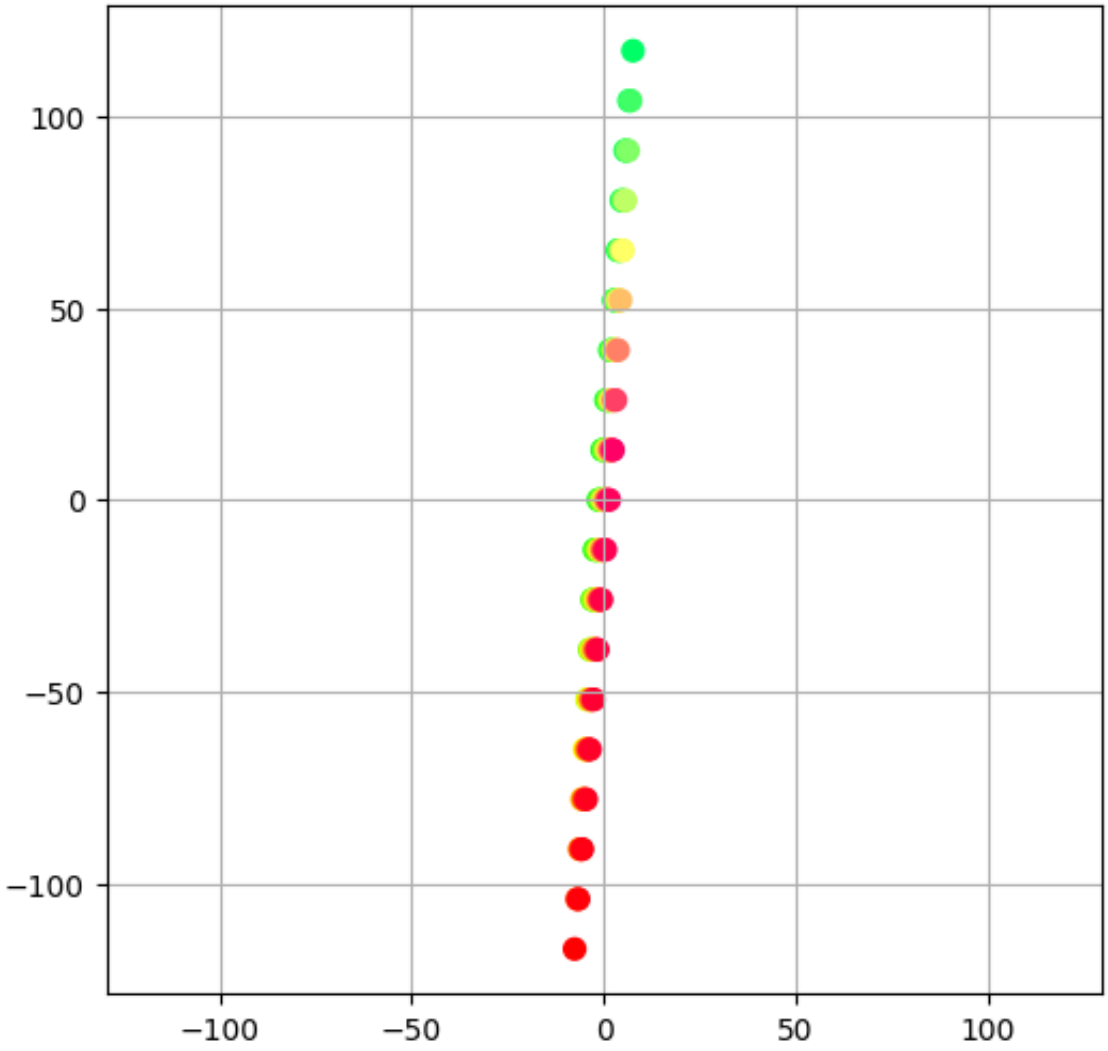
```
[[ 0.49852724 -0.06642112]
 [-0.06642112  0.49852724]]
```



6. Отображение, которое переводит прямую y = 0 в y = ax и прямую x = 0 в y = bx:

In [12]:
```
sixth_matrix = np.array([
    [1, a / b],
    [a, a]
])

xy_new = np.dot(sixth_matrix, xy_old)

plt.figure(figsize=(6, 6), facecolor="w")
plt.scatter(xy_new[0], xy_new[1], s = 50, c = colors)
plt.axis('equal')
plt.grid(True)
print(sixth_matrix)
```

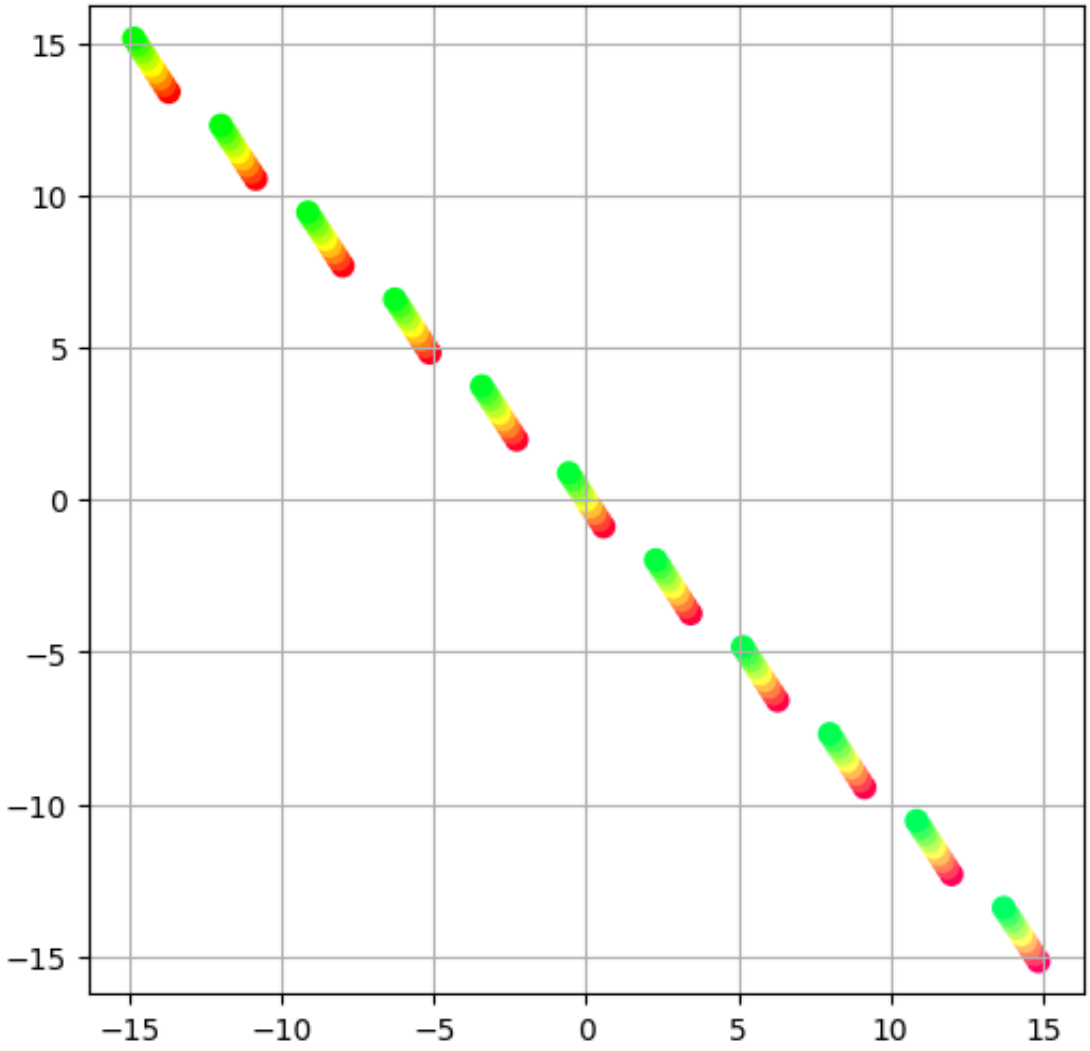
```
[[ 1.    0.65]
 [13.   13.  ]]
```



7. Отображение, которое переводит прямую y = ax в y = 0 и прямую y = bx в x = 0:

In [13]:
```
seventh_matrix = np.round(np.linalg.inv(sixth_matrix), 10)

xy_new = np.dot(seventh_matrix, xy_old)

plt.figure(figsize=(6, 6), facecolor="w")
plt.scatter(xy_new[0], xy_new[1], s = 50, c = colors)
plt.axis('equal')
plt.grid(True)
print(seventh_matrix)
```

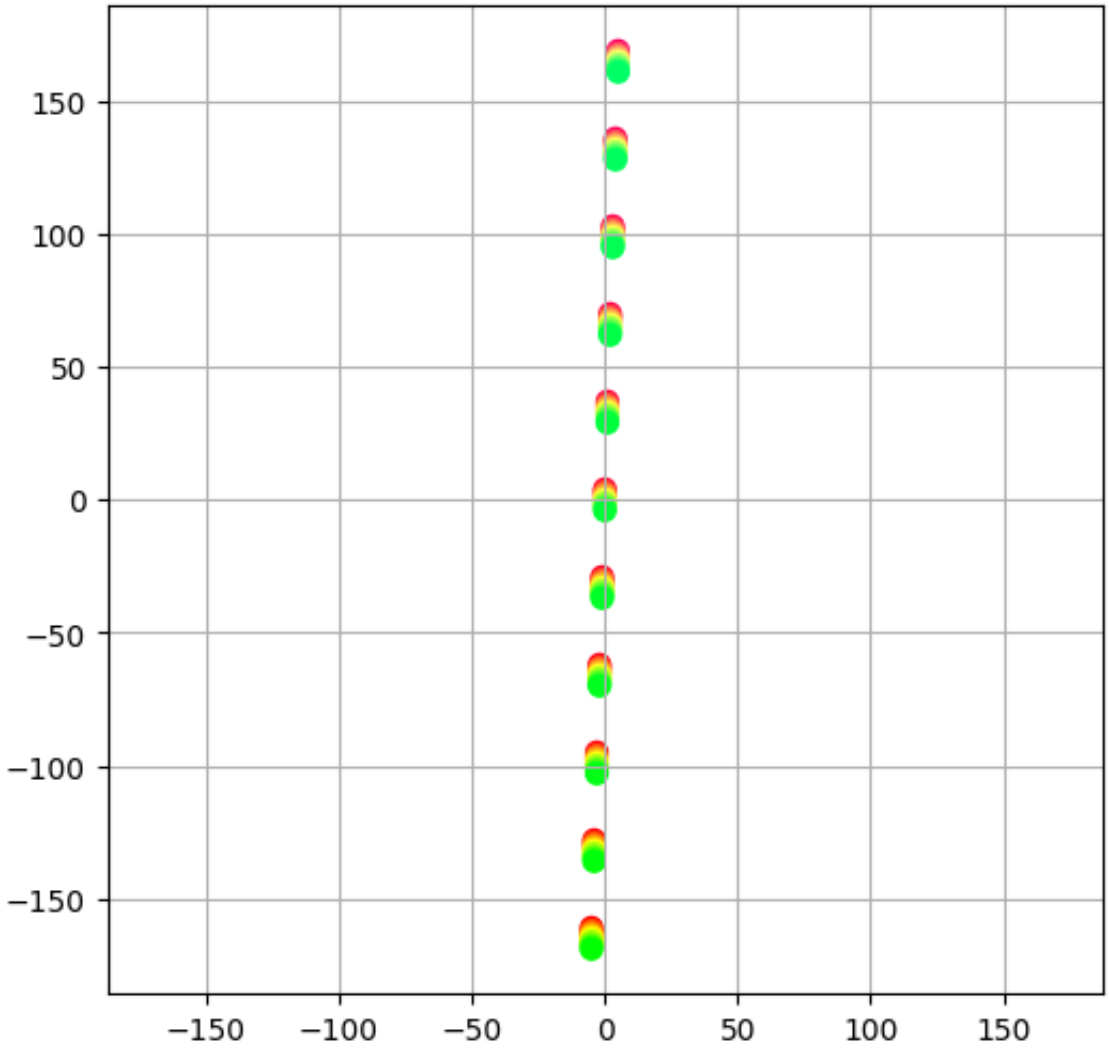
```
[[ 2.85714286 -0.14285714]
 [-2.85714286  0.21978022]]
```



8. Отображение, которое меняет местами прямые y = ax и y = bx:

In [14]:
```
eight_matrix = np.array([
    [1, 0],
    [b+a, -1]
])

xy_new = np.dot(eight_matrix, xy_old)

plt.figure(figsize=(6, 6), facecolor="w")
plt.scatter(xy_new[0], xy_new[1], s = 50, c = colors)
plt.axis('equal')
plt.grid(True)
print(eight_matrix)
```

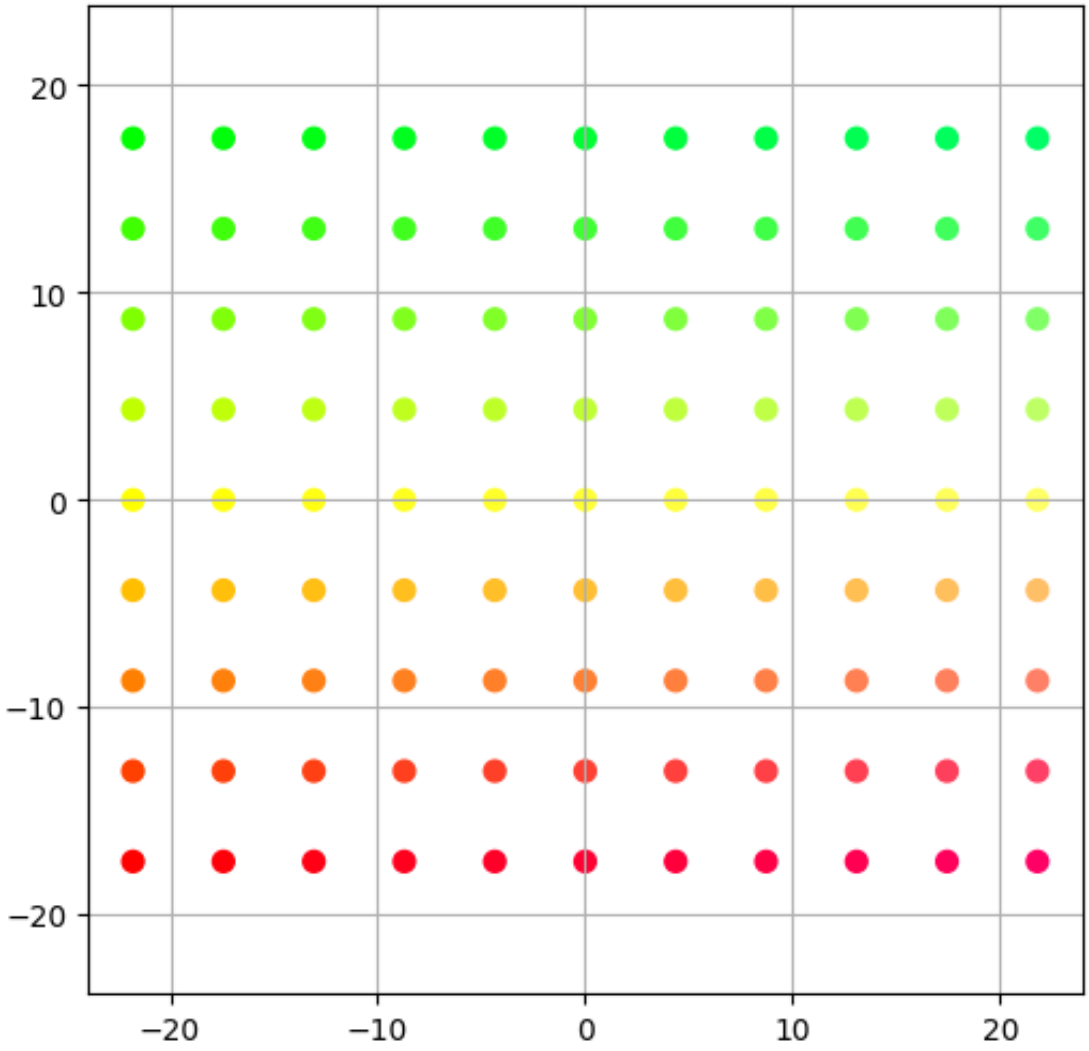
```
[[ 1  0]
 [33 -1]]
```



9. Отображение, которое переводит круг единичной площади с центром в начале координат в круг площади c:

In [15]:
```
ninth_matrix = np.array([
    [np.sqrt(c), 0],
    [0, np.sqrt(c)]
])

xy_new = np.dot(ninth_matrix, xy_old)

plt.figure(figsize=(6, 6), facecolor="w")
plt.scatter(xy_new[0], xy_new[1], s = 50, c = colors)
plt.axis('equal')
plt.grid(True)
print(ninth_matrix)
```

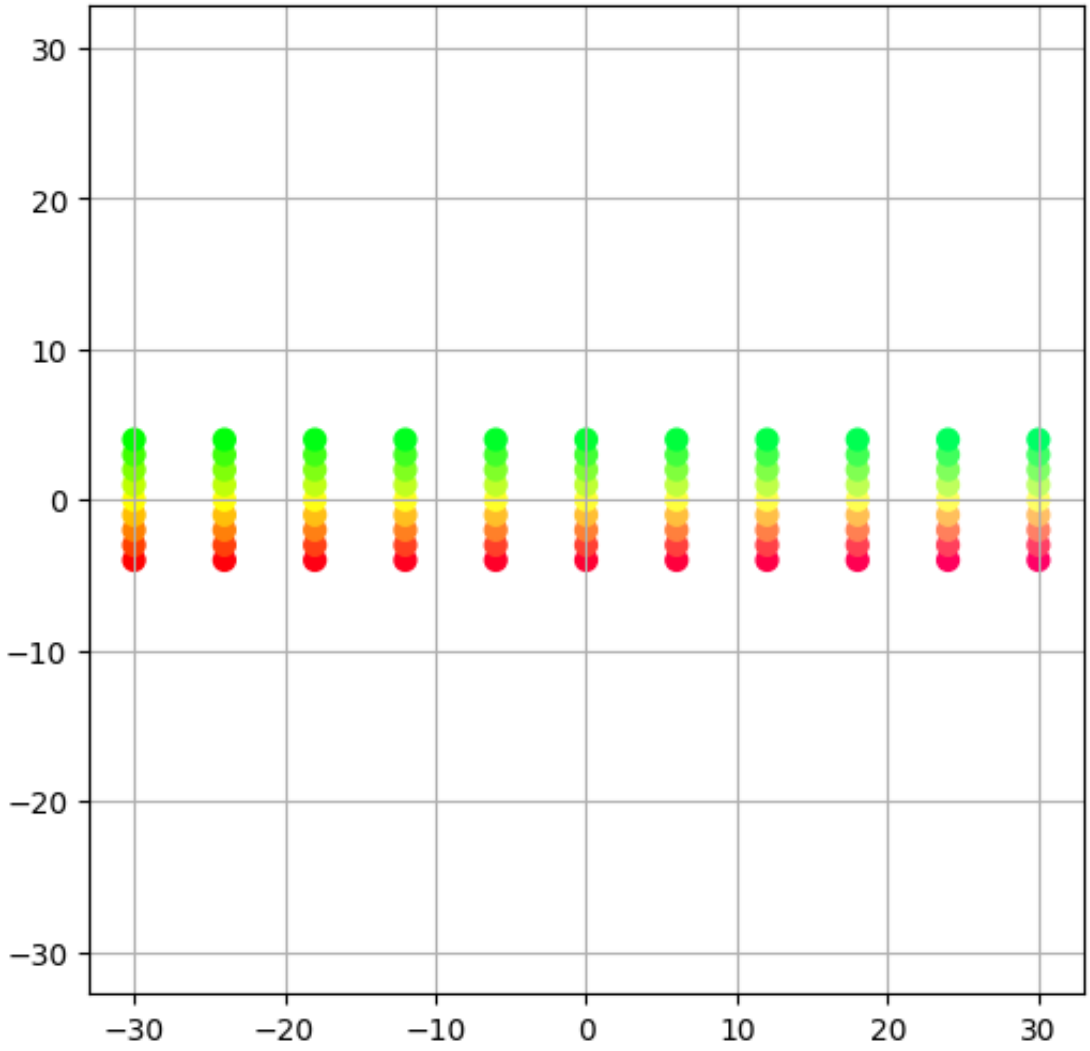
```
[[4.35889894 0.        ]
 [0.         4.35889894]]
```



10. Отображение, которое переводит круг единичной площади с центром в начале координат в некруг площади d:

In [16]:
```
tenth_matrix = np.array([
    [d, 0],
    [0, 1]
])

xy_new = np.dot(tenth_matrix, xy_old)

plt.figure(figsize=(6, 6), facecolor="w")
plt.scatter(xy_new[0], xy_new[1], s = 50, c = colors)
plt.axis('equal')
plt.grid(True)
print(tenth_matrix)
```

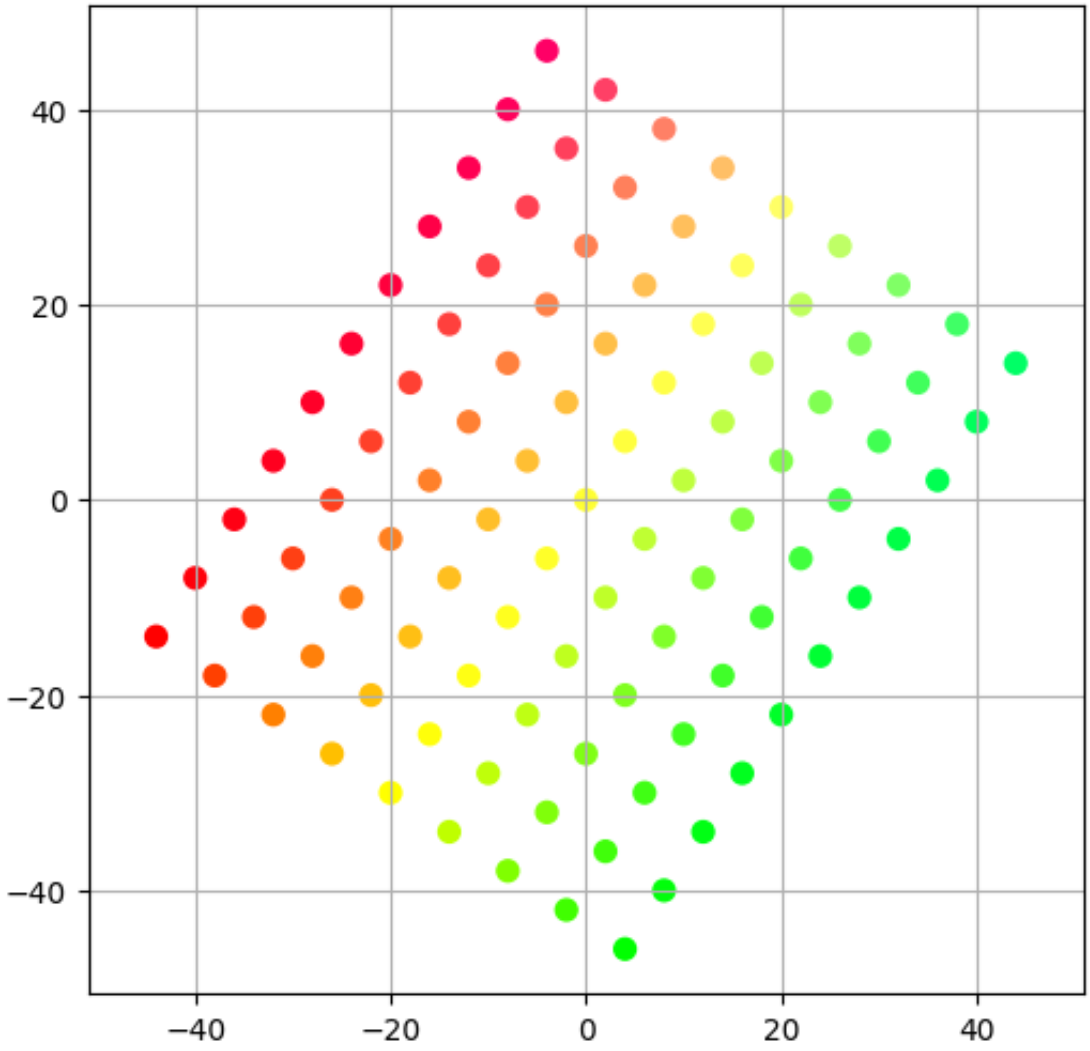
```
[[6 0]
 [0 1]]
```



11. Отображение, у которого собственные вектора перпендикулярны, и ни один из них не лежит на прямой y = 0 или y = x:

In [17]:
```
eleventh_matrix = np.array([
    [4, 6],
    [6, -4]
])

xy_new = np.dot(eleventh_matrix, xy_old)

plt.figure(figsize=(6, 6), facecolor="w")
plt.scatter(xy_new[0], xy_new[1], s = 50, c = colors)
plt.axis('equal')
plt.grid(True)
print(tenth_matrix)
```

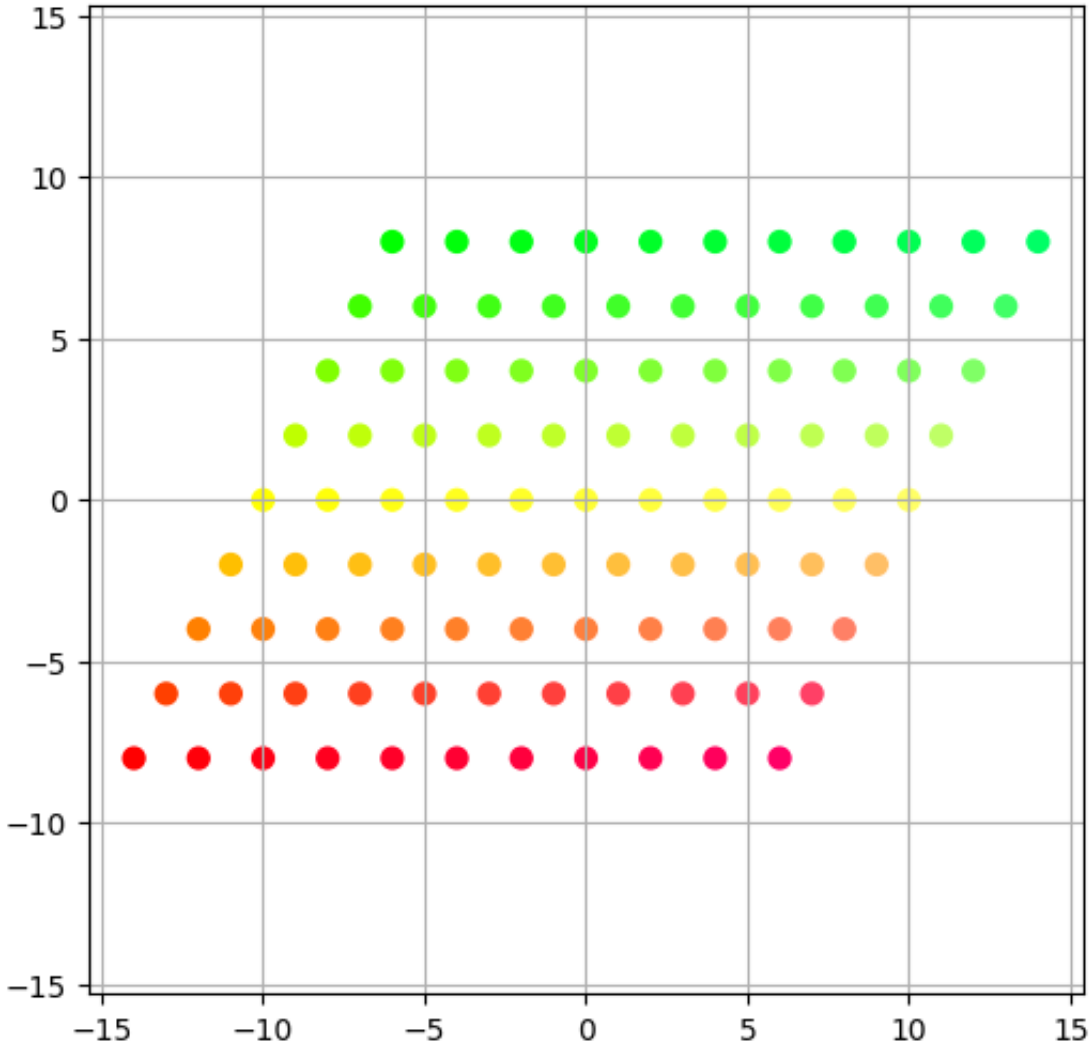
```
[[6 0]
 [0 1]]
```



12. Отображение, у которого нет двух неколлинеарных собственных векторов:

In [18]:
```
twelfth_matrix = np.array([
    [2, 1],
    [0, 2]
])
xy_new = np.dot(twelfth_matrix, xy_old)

plt.figure(figsize=(6, 6), facecolor="w")
plt.scatter(xy_new[0], xy_new[1], s = 50, c = colors)
plt.axis('equal')
plt.grid(True)
print(twelfth_matrix)
```

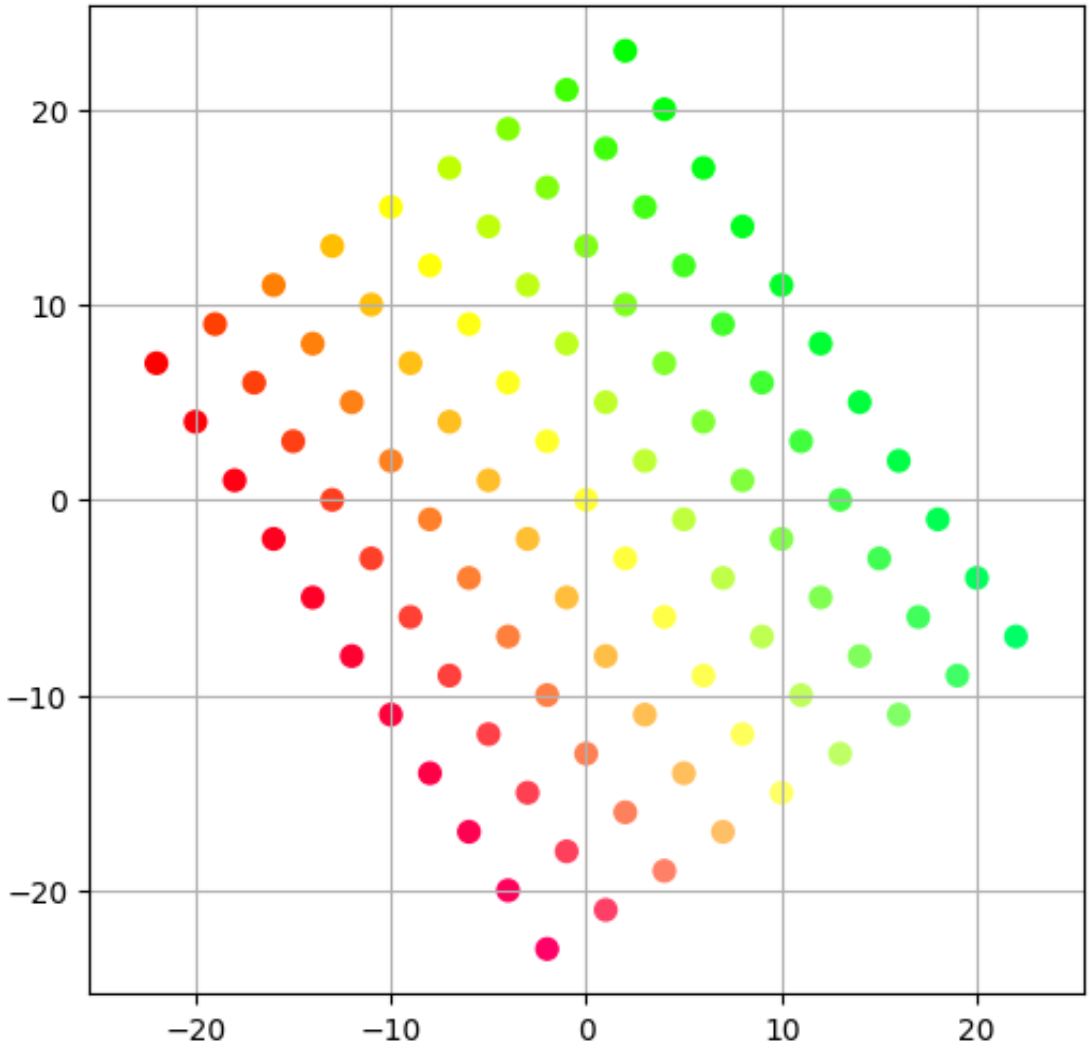
```
[[2 1]
 [0 2]]
```



13. Отображение, у которого нет ни одного вещественного собственного вектора (но при этом само отображение задаётся вещественной матрицей):

In [19]:
```
thirteenth_matrix = np.array([
    [2, 3],
    [-3, 2]
])

xy_new = np.dot(thirteenth_matrix, xy_old)

plt.figure(figsize=(6, 6), facecolor="w")
plt.scatter(xy_new[0], xy_new[1], s = 50, c = colors)
plt.axis('equal')
plt.grid(True)
print(thirteenth_matrix)
```

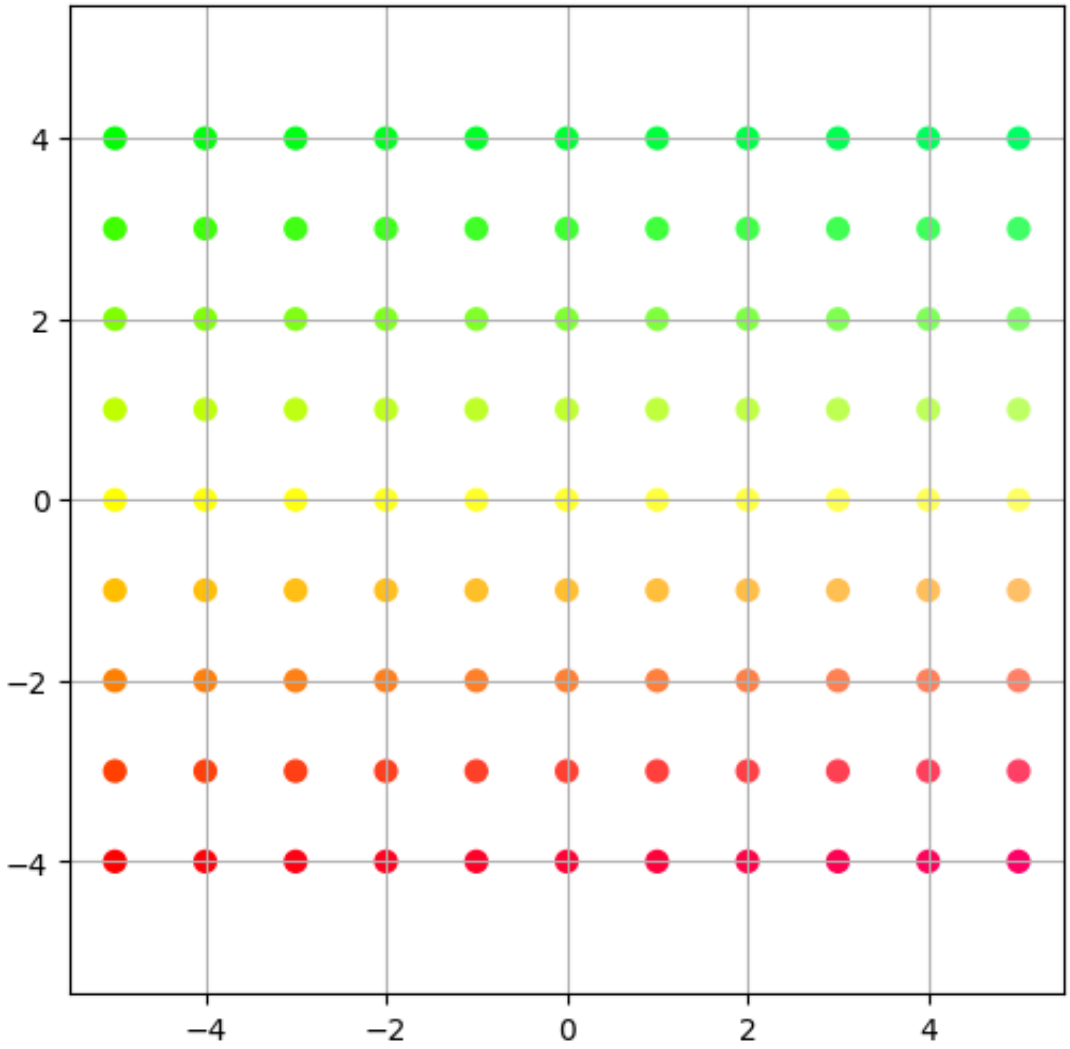
```
[[ 2  3]
 [-3  2]]
```



14. Отображение, для которого любой ненулевой вектор является собственным:

In [20]:
```
fourteenth_matrix = np.array([
    [1, 0],
    [0, 1]
])

xy_new = np.dot(fourteenth_matrix, xy_old)

plt.figure(figsize=(6, 6), facecolor="w")
plt.scatter(xy_new[0], xy_new[1], s = 50, c = colors)
plt.axis('equal')
plt.grid(True)
print(fourteenth_matrix)
```

```
[[1 0]
 [0 1]]
```

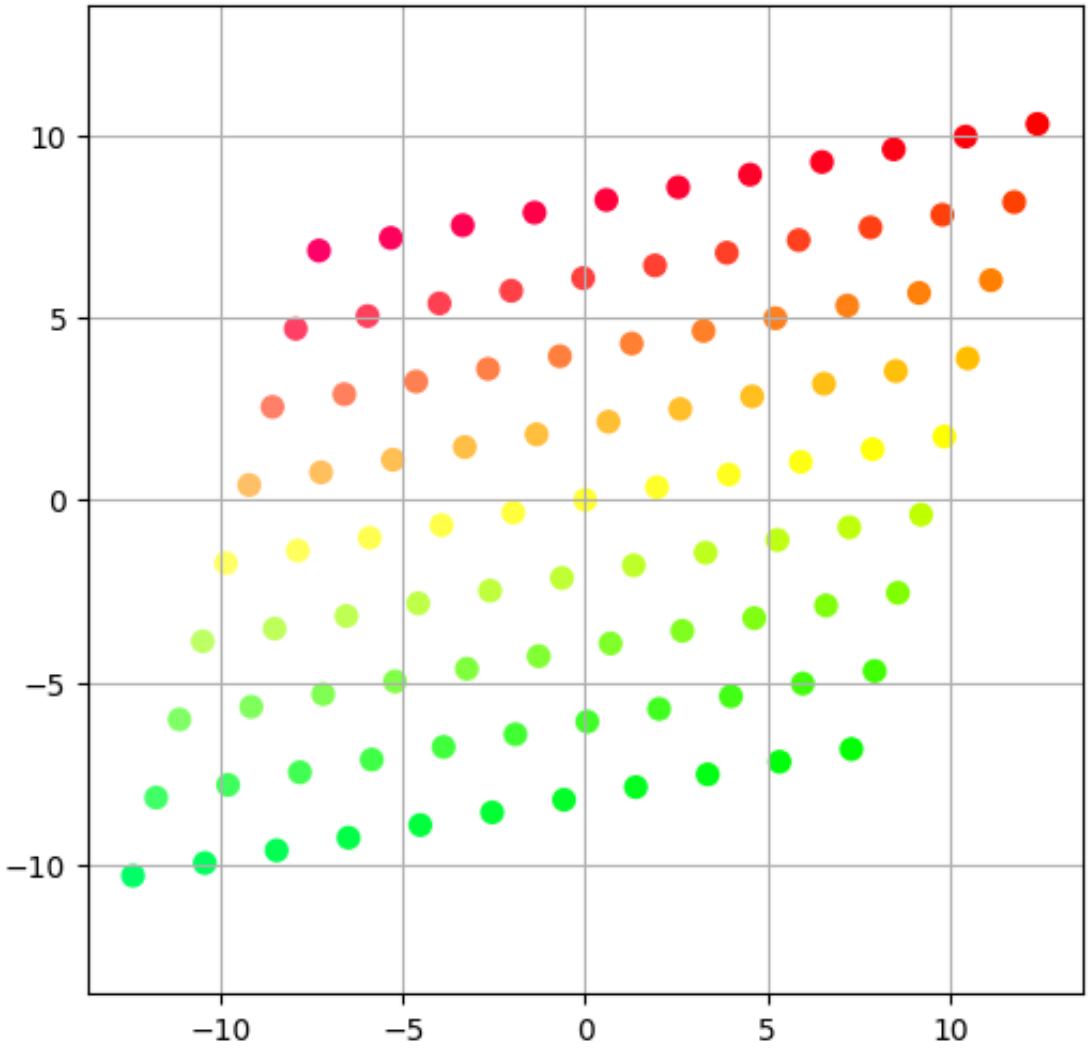


15. Пару отображений, последовательное применение которых даёт различные результаты в зависимости от порядка: AB != BA. Сделайте визуализацию всех рассматриваемых отображений, а именно: A, B, AB и BA:

15.1 Первое отображение:

In [21]:
```
fifteenth_matrix_1 = np.dot(third_matrix, twelfth_matrix)

xy_new = np.dot(fifteenth_matrix_1 , xy_old)

plt.figure(figsize=(6, 6), facecolor="w")
plt.scatter(xy_new[0], xy_new[1], s = 50, c = colors)
plt.axis('equal')
plt.grid(True)
print(fifteenth_matrix_1)
```

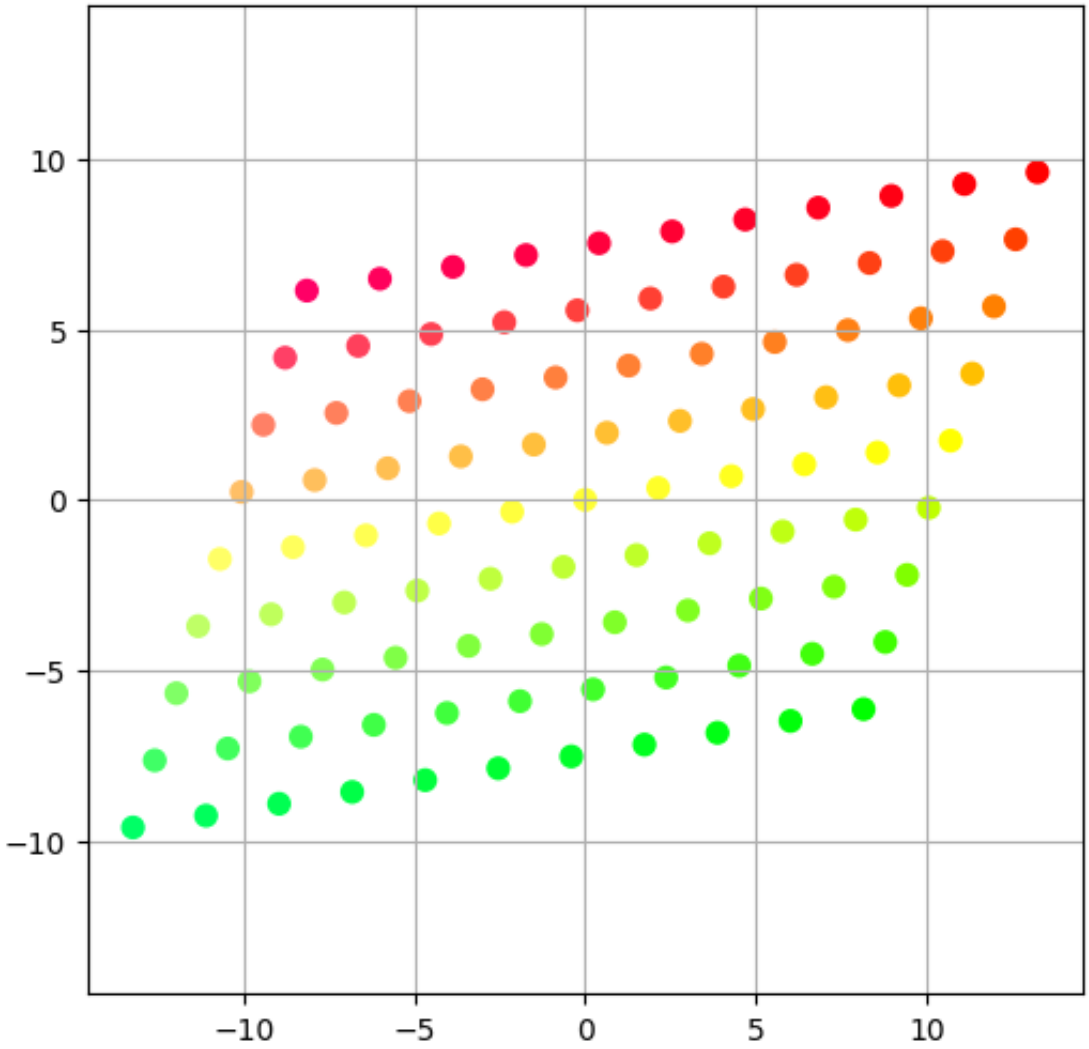
```
[[-1.96961551 -0.6375114 ]
 [-0.34729636 -2.14326368]]
```



15.2 Второе отображение:

In [22]:
```
fifteenth_matrix_2 = np.dot(twelfth_matrix, third_matrix)

xy_new = np.dot(fifteenth_matrix_2, xy_old)

plt.figure(figsize=(6, 6), facecolor="w")
plt.scatter(xy_new[0], xy_new[1], s = 50, c = colors)
plt.axis('equal')
plt.grid(True)
print(fifteenth_matrix_2)
```

```
[[-2.14326368 -0.6375114 ]
 [-0.34729636 -1.96961551]]
```

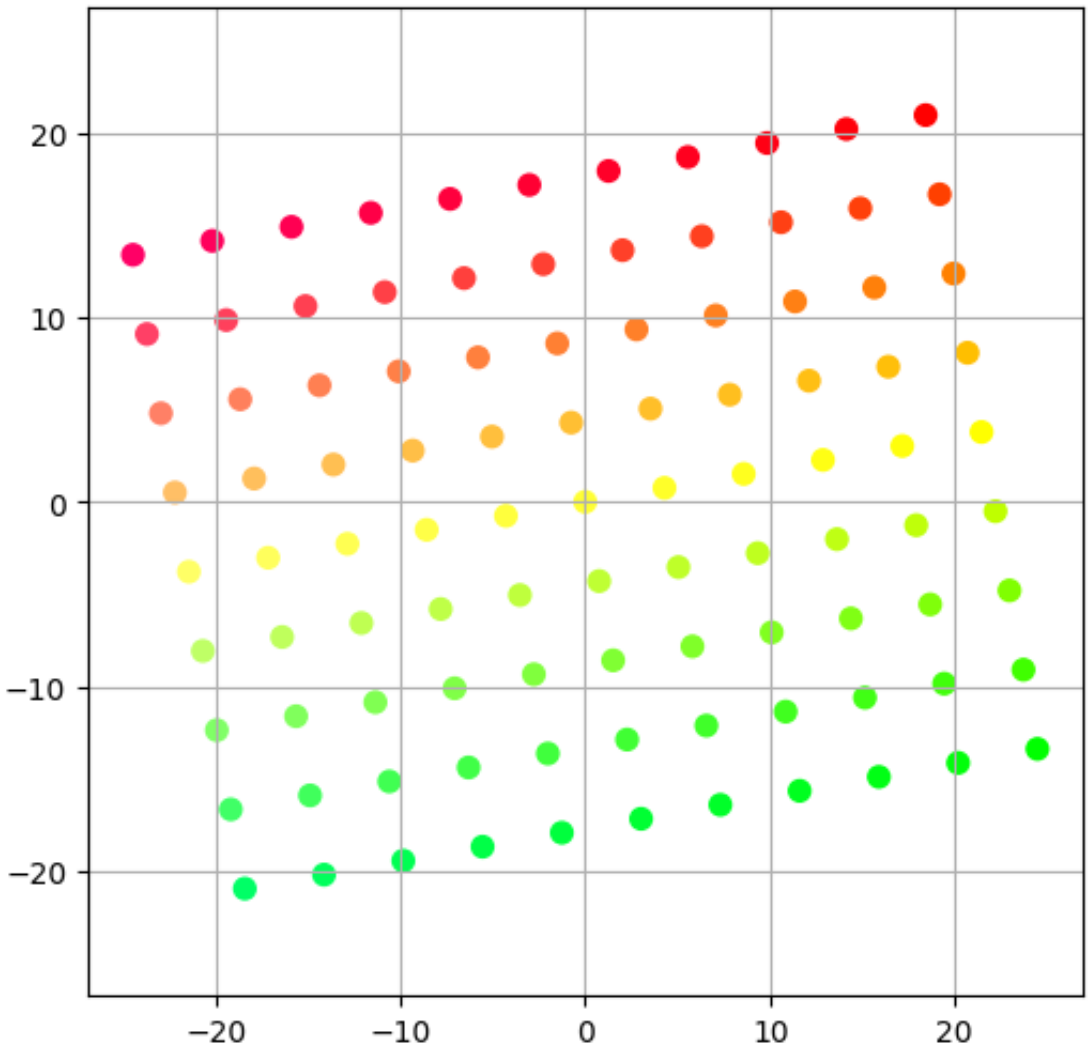


16. Пару отображений, последовательное применение которых даёт одинаковый результат независимо от порядка: AB = BA. Постарайтесь, чтобы матрицы A и B были максимально непохожими друг на друга. Сделайте визуализацию, аналогичную передыдущему пункту:

16.1 Первое отображение:

In [23]:
```
sixteenth_matrix_1 = np.dot(third_matrix, ninth_matrix)

xy_new = np.dot(sixteenth_matrix_1, xy_old)

plt.figure(figsize=(6, 6), facecolor="w")
plt.scatter(xy_new[0], xy_new[1], s = 50, c = colors)
plt.axis('equal')
plt.grid(True)
print(sixteenth_matrix_1)
```

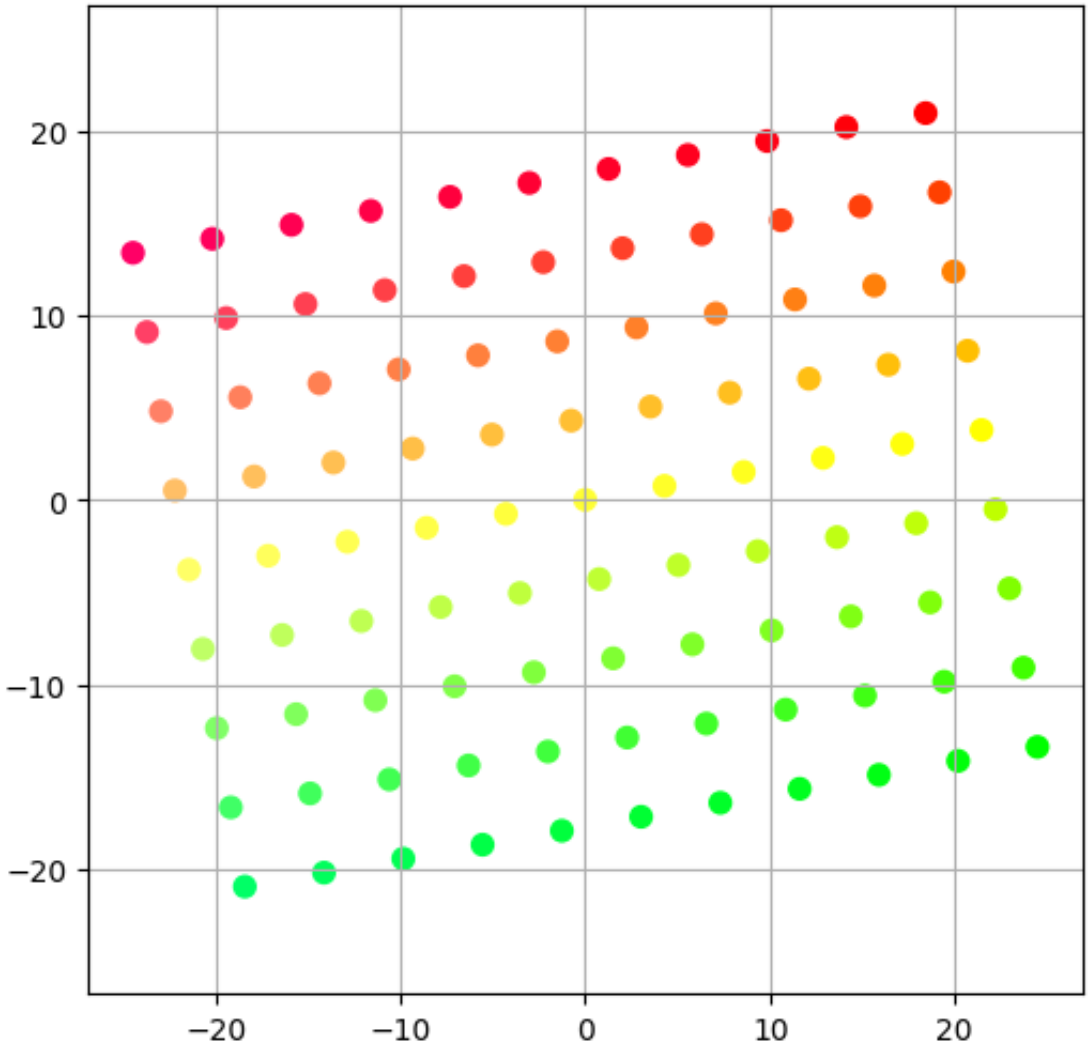
```
[[-4.29267747  0.75691486]
 [-0.75691486 -4.29267747]]
```



16.2 Второе отображение:

In [24]:
```
sixteenth_matrix_2 = np.dot(ninth_matrix, third_matrix)

xy_new = np.dot(sixteenth_matrix_2, xy_old)

plt.figure(figsize=(6, 6), facecolor="w")
plt.scatter(xy_new[0], xy_new[1], s = 50, c = colors)
plt.axis('equal')
plt.grid(True)
print(sixteenth_matrix_2)
```

```
[[-4.29267747  0.75691486]
 [-0.75691486 -4.29267747]]
```



Задание 2.

1. Найдите образ и ядро придуманных вами отображений из пунктов 1, 2, 13, 14:

In [25]:
```
matrix_list = [first_matrix, second_matrix, thirteenth_matrix, fourteenth
def find_image_and_kernel(matrix):
    matrix = np.array(matrix)
    rank = np.linalg.matrix_rank(matrix)
    U, S, Vt = np.linalg.svd(matrix)
    image = U[:, :rank]
    V = Vt.T
    kernel = V[:, rank:]
    return image, kernel

for matrix in matrix_list:
    print("Линейный оператор:\n")
    print(matrix)
    im, ker = find_image_and_kernel(matrix)
```

```
    print("Образ:\n")
    print(im)
    print("Ядро:\n")
    print(ker)
```

```
Линейный оператор:

[[ 0.99705449  0.0766965 ]
 [-0.0766965   0.99705449]]
Образ:

[[-0.99705449  0.0766965 ]
 [ 0.0766965   0.99705449]]
Ядро:

[]
Линейный оператор:

[[ 1  1]
 [20 20]]
Образ:

[[-0.04993762]
 [-0.99875234]]
Ядро:

[[-0.70710678]
 [ 0.70710678]]
Линейный оператор:

[[ 2  3]
 [-3  2]]
Образ:

[[-0.5547002   0.83205029]
 [ 0.83205029  0.5547002 ]]
Ядро:

[]
Линейный оператор:

[[1 0]
 [0 1]]
Образ:

[[1. 0.]
 [0. 1.]]
Ядро:

[]
```

2. Найдите собственные числа и собственные вектора придуманных вами отображений из пунктов 1, 2, 3, 4,

8, 11, 12, 13, 14, 15, 16:

In [26]:
```
matrix_list = [
    first_matrix, second_matrix, third_matrix, fourth_matrix, eight_matri
    eleventh_matrix, twelfth_matrix, thirteenth_matrix,
    fourth_matrix, fifteenth_matrix_1, fifteenth_matrix_2,
    sixteenth_matrix_1
]
def find_eigenvalues_and_eigenvectors(matrix):
    matrix = np.array(matrix)
    eigenvalues, eigenvectors = np.linalg.eig(matrix)

    return eigenvalues, eigenvectors
for matrix in matrix_list:
    print("Линейный оператор:\n")
    print(matrix)
    val, vec = find_eigenvalues_and_eigenvectors(matrix)
    print("Собственные числа:\n")
    print(val)
    print("Собственные вектора:\n")
    print(vec)
    print()
```

```
Линейный оператор:

[[ 0.99705449  0.0766965 ]
 [-0.0766965   0.99705449]]
Собственные числа:

[0.99705449+0.0766965j 0.99705449-0.0766965j]
Собственные вектора:

[[0.70710678+0.j         0.70710678-0.j        ]
 [0.        +0.70710678j 0.        -0.70710678j]]

Линейный оператор:

[[ 1  1]
 [20 20]]
Собственные числа:

[ 0. 21.]
Собственные вектора:

[[-0.70710678 -0.04993762]
 [ 0.70710678 -0.99875234]]

Линейный оператор:

[[-0.98480775  0.17364818]
 [-0.17364818 -0.98480775]]
Собственные числа:
```

```
[-0.98480775+0.17364818j -0.98480775-0.17364818j]
Собственные вектора:

[[0.70710678+0.j         0.70710678-0.j        ]
 [0.        +0.70710678j 0.        -0.70710678j]]

Линейный оператор:

[[-1.0000000e+00 -1.2246468e-16]
 [ 1.2246468e-16 -1.0000000e+00]]
Собственные числа:

[-1.+1.2246468e-16j -1.-1.2246468e-16j]
Собственные вектора:

[[0.70710678+0.j         0.70710678-0.j        ]
 [0.        -0.70710678j 0.        +0.70710678j]]

Линейный оператор:

[[ 1  0]
 [33 -1]]
Собственные числа:

[-1.  1.]
Собственные вектора:

[[0.         0.06049506]
 [1.         0.9981685 ]]

Линейный оператор:

[[ 4  6]
 [ 6 -4]]
Собственные числа:

[ 7.21110255 -7.21110255]
Собственные вектора:

[[ 0.8816746  -0.47185793]
 [ 0.47185793  0.8816746 ]]

Линейный оператор:

[[2 1]
 [0 2]]
Собственные числа:

[2. 2.]
Собственные вектора:

[[ 1.0000000e+00 -1.0000000e+00]
 [ 0.0000000e+00  4.4408921e-16]]
```

```
Линейный оператор:

[[ 2  3]
 [-3  2]]
Собственные числа:

[2.+3.j 2.-3.j]
Собственные вектора:

[[0.70710678+0.j         0.70710678-0.j        ]
 [0.        +0.70710678j 0.        -0.70710678j]]

Линейный оператор:

[[-1.0000000e+00 -1.2246468e-16]
 [ 1.2246468e-16 -1.0000000e+00]]
Собственные числа:

[-1.+1.2246468e-16j -1.-1.2246468e-16j]
Собственные вектора:

[[0.70710678+0.j         0.70710678-0.j        ]
 [0.        -0.70710678j 0.        +0.70710678j]]

Линейный оператор:

[[-1.96961551 -0.6375114 ]
 [-0.34729636 -2.14326368]]
Собственные числа:

[-1.57795887 -2.53492032]
Собственные вектора:

[[ 0.85205065  0.74820833]
 [-0.52345934  0.66346386]]

Линейный оператор:

[[-2.14326368 -0.6375114 ]
 [-0.34729636 -1.96961551]]
Собственные числа:

[-2.53492032 -1.57795887]
Собственные вектора:

[[-0.85205065  0.74820833]
 [-0.52345934 -0.66346386]]

Линейный оператор:

[[-4.29267747  0.75691486]
 [-0.75691486 -4.29267747]]
```

```
Собственные числа:

[-4.29267747+0.75691486j -4.29267747-0.75691486j]
Собственные вектора:

[[0.        -0.70710678j 0.        +0.70710678j]
 [0.70710678+0.j         0.70710678-0.j        ]]
```

3. Найдите определитель матриц из пунктов 1, 2, 3, 4, 5, 9, 10:

In [27]:
```
matrix_list = [
    first_matrix, second_matrix, third_matrix,
    fourth_matrix, fifth_matrix,
    ninth_matrix, tenth_matrix
]

for matrix in matrix_list:
    print("Линейный оператор:\n")
    print(matrix)
    print(f"Определитель: {np.linalg.det(matrix)}")
    print()
```

```
Линейный оператор:

[[ 0.99705449  0.0766965 ]
 [-0.0766965   0.99705449]]
Определитель: 1.0

Линейный оператор:

[[ 1  1]
 [20 20]]
Определитель: 0.0

Линейный оператор:

[[-0.98480775  0.17364818]
 [-0.17364818 -0.98480775]]
Определитель: 0.9999999999999999

Линейный оператор:

[[-1.0000000e+00 -1.2246468e-16]
 [ 1.2246468e-16 -1.0000000e+00]]
Определитель: 1.0

Линейный оператор:

[[ 0.49852724 -0.06642112]
 [-0.06642112  0.49852724]]
Определитель: 0.24411764705882363

Линейный оператор:

[[4.35889894 0.        ]
 [0.         4.35889894]]
Определитель: 19.000000000000004

Линейный оператор:

[[6 0]
 [0 1]]
Определитель: 6.0
```

4. В каких пунктах матрица обязательно получается симметричной?

In [28]:
```
matrix_list = [
    first_matrix,
    second_matrix,
    third_matrix,
    fourth_matrix,
    fifth_matrix,
```

```
    sixth_matrix,
    seventh_matrix,
    eight_matrix,
    ninth_matrix,
    tenth_matrix,
    eleventh_matrix,
    twelfth_matrix,
    thirteenth_matrix,
    fourth_matrix,
    fifteenth_matrix_1,
    fifteenth_matrix_2,
    sixth_matrix
]
for i in range(len(matrix_list)):
    print(f"Линейный оператор {i + 1}:\n")
    print(np.round(matrix_list[i],5))
    print()
print("Answer: 4 8 9 10 11 14 ")
```

```
Линейный оператор 1:

[[ 0.99705  0.0767 ]
 [-0.0767   0.99705]]

Линейный оператор 2:

[[ 1  1]
 [20 20]]

Линейный оператор 3:

[[-0.98481  0.17365]
 [-0.17365 -0.98481]]

Линейный оператор 4:

[[-1. -0.]
 [ 0. -1.]]

Линейный оператор 5:

[[ 0.49853 -0.06642]
 [-0.06642  0.49853]]

Линейный оператор 6:

[[ 1.    0.65]
 [13.   13.  ]]

Линейный оператор 7:

[[ 2.85714 -0.14286]
 [-2.85714  0.21978]]
```

```
Линейный оператор 8:

[[ 1  0]
 [33 -1]]

Линейный оператор 9:

[[4.3589 0.    ]
 [0.     4.3589]]

Линейный оператор 10:

[[6 0]
 [0 1]]

Линейный оператор 11:

[[ 4  6]
 [ 6 -4]]

Линейный оператор 12:

[[2 1]
 [0 2]]

Линейный оператор 13:

[[ 2  3]
 [-3  2]]

Линейный оператор 14:

[[-1. -0.]
 [ 0. -1.]]

Линейный оператор 15:

[[-1.96962 -0.63751]
 [-0.3473  -2.14326]]

Линейный оператор 16:

[[-2.14326 -0.63751]
 [-0.3473  -1.96962]]

Линейный оператор 17:

[[ 1.    0.65]
 [13.   13.  ]]

Answer: 4 8 9 10 11 14
```