In [1]:
```
#Use Python to solve the following system of equations. Hint: You can write this syste
#equation in in the form Ax = b, where the matrix A is invertible.
#27x1 – 10x2 + 4x3 – 29x4 = 1
#–16x1 + 5x2 – 2x3 + 18x4 = –1
#–17x1 + 4x2 – 2x3 + 20x4 = 0
#–7x1 + 2x2 – x3 + 8x4 = 1

#load np
import numpy as np


#write as matrix
#might be wrong? Do SoE
#matrix A
A = np.array([[27, -10, 4, -29], [-16, 5, -2, 18], [-17, 4, -2, 20], [-7, 2, -1, 8]])

#vector b
b = np.array([1, -1, 0, 1])

#system of equations
x = np.linalg.solve(A, b)
print("solution:")
print("x1 =", x[0])
print("x2 =", x[1])
print("x3 =", x[2])
print("x4 =", x[3])
```

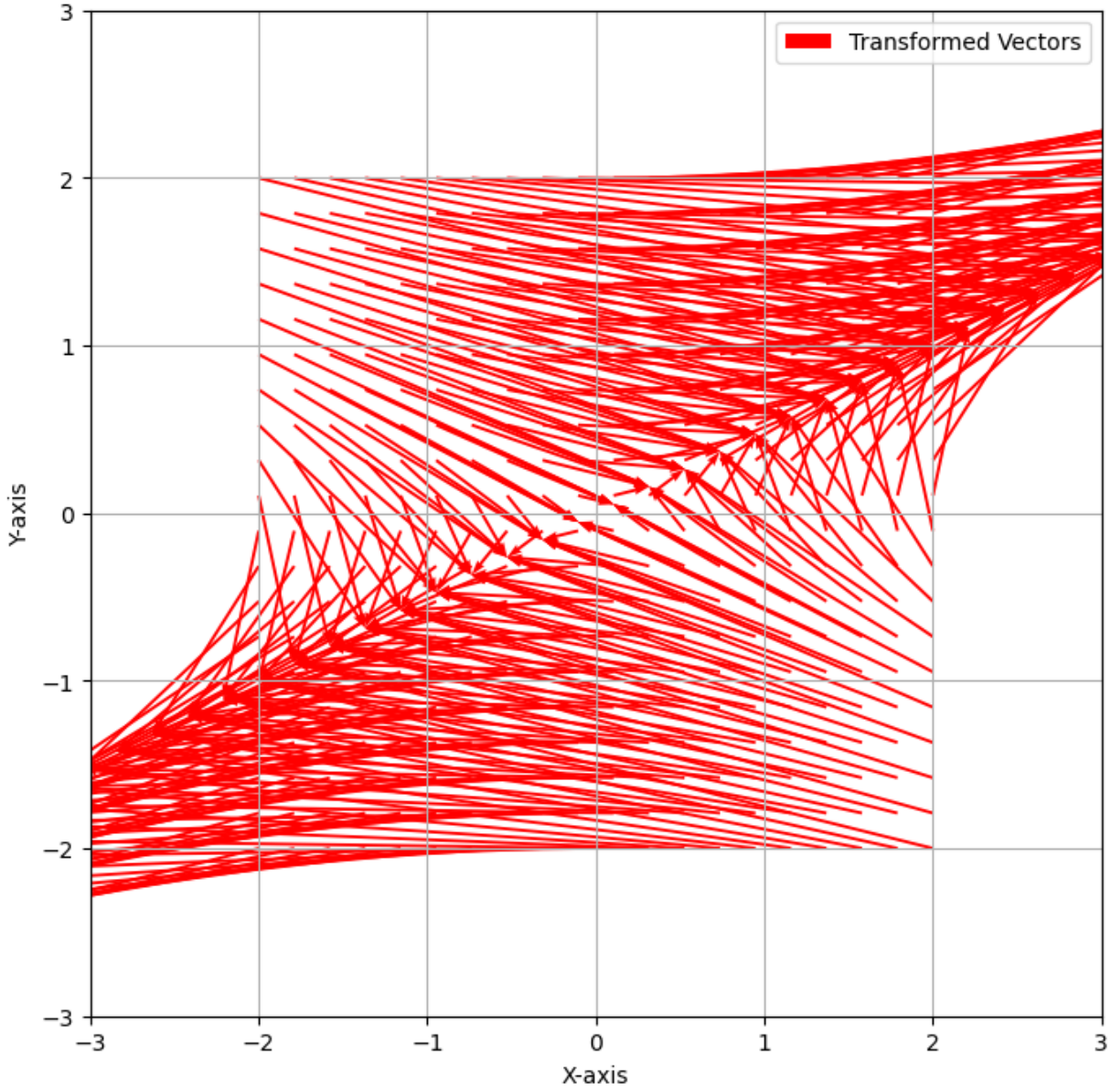
```
solution:
x1 = 10.000000000000053
x2 = 3.000000000000033
x3 = -9.000000000000018
x4 = 7.000000000000035
```

In [3]:
```
#Let T be the linear transformation that takes in a vector v in R2 and outputs the pro
#matrix A shown below and v.
# √ √  2/2 – 2/2 A = √ √ 2/2 2/2
#I.e., T is a function that takes in a vector v in R2 and outputs the vector T(v) = Av
#  2 (a) Find T(x), where x = . –2

#√ causes syntax,  replace with acutal value for matrix A

#load np
import numpy as np

#matrix A without square root symbols
A = np.array([[np.sqrt(2), np.sqrt(2)], [2/2, 2/2]])

#vector x
x = np.array([2, -2])

#T(x) by multiplying A and x
Tx = np.dot(A, x)
print("T(x) =", Tx)
```

```
T(x) = [0. 0.]
```

In [4]:
```
#Mimic the code in Section 7.1.3 of Essential Math for Data Science to visualize the l

#load np, plt
```

```
import numpy as np
import matplotlib.pyplot as plt

#matrix A
A = np.array([[1, 2], [0.5, 1]])

#x and y coordinates, create grid using meshgrid
x = np.linspace(-2, 2, 20)
y = np.linspace(-2, 2, 20)
X, Y = np.meshgrid(x, y)

#linear transformation to grid points
X_transformed = A[0, 0] * X + A[0, 1] * Y
Y_transformed = A[1, 0] * X + A[1, 1] * Y

#use quiver for visualization
plt.figure(figsize=(8, 8))
plt.quiver(X, Y, X_transformed - X, Y_transformed - Y, angles='xy', scale_units='xy',

#plot axis
plt.xlim(-3, 3)
plt.ylim(-3, 3)
plt.xlabel('X-axis')
plt.ylabel('Y-axis')

#plot
plt.legend()
plt.grid()
plt.show()
```



Using our visualization , T transforms vectors in general by scaling along the first basis vector. In the grid, vectors are spread out based on their positions along the x axis by a factor of 1, and 0.5 for the y axis. This will lead to vectors being stretched or compressed horizontally depending on their inital positions. The further right the vector is, the more it is stretched horizontally, while the further left are more compressed.