Math 425 Computation Linear Algebra

HW3, Part B (Question 9)

Brent A. Thorne

brentathorne@gmail.com

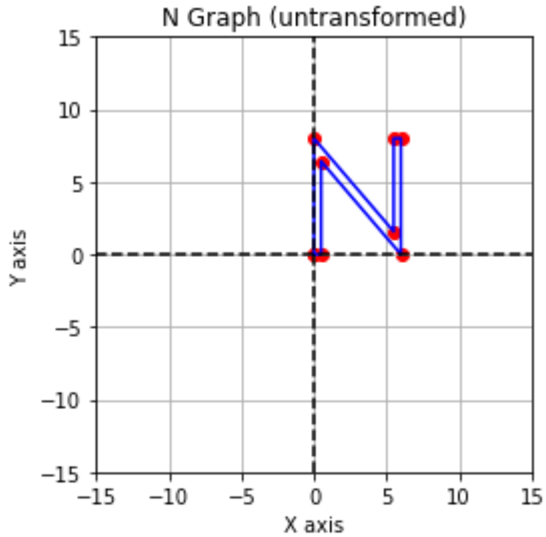
Uniqueness, linear transformations, range and domain.

In [33]:
```
# environment setup, try to make it clear which library I'm using for what
import numpy as np  # nice arrays and other stuff
import sympy as sym # symbollic maths
from sympy.matrices import Matrix # pretty matrices
from sympy import Eq # pretty equations
from sympy.physics.quantum.dagger import Dagger # we'll want this later...
from math import e, pi, sqrt # Mathy math math
from mpl_toolkits.mplot3d import Axes3D # we like 3d quivers for tutorials
import matplotlib.pyplot as plt # old standby for plotting like a villian
from IPython.display import display, Math, Latex # used to display formatted re
sults in the console
sym.init_printing()  # initialize pretty printing
```

9. Find the 3×3 matrices that produce the described composite 2D transformations, using homogeneous coordinates. Apply the transformations to the 'letter N' data, ``letterN.pny'' and submit the corresponding plots as well.

In [138]:

```
class letter:  # totally overkill
    """
    import numpy data and return letter object
    provides functions transform and plot
    assumes numpy data is 2-rows (2xm)
    recall: An m×n matrix has m rows and n columns.
    """
    def __init__(self, filename):
        assert isinstance(filename, str)
        self.filename = filename
        self.T = sym.eye(3)  # add feature to set transform on creation
        self.D = Matrix(np.load(filename))
        self.D = self.D.col_join(sym.ones(1,D.cols))

    def eye(self):
        self.T = sym.eye(3)

    def plot(self, title = 'Letter Plot'):
        lim=15 # consider feature to sets limits based on origin and average po
ints
        DD = self.T * self.D  # do inner product at plotting
        plt.title(f"{title}"); plt.xlabel("X axis"); plt.ylabel("Y axis")
        plt.scatter(list(DD.row(0)), list(DD.row(1)), color ="red")
        plt.plot(list(DD.row(0)), list(DD.row(1)), color ="blue")
        plt.xlim(-lim,lim); plt.ylim(-lim,lim)
        plt.grid(); plt.gca().set_aspect("equal") # square grids are pretty
        plt.axhline(0, color='black', linestyle='--')
        plt.axvline(0, color='black', linestyle='--')
        plt.show()

    def __mul__(self, other): #dot the transform
        if isinstance(other, Matrix):
            self.T = other * self.T
        else:
            return NotImplemented

    def dot(self, other): # dot yourself
        return letter.__mul__(self, other)

    def report(self): # so pretty
        display(Latex(f'$TD={sym.latex(self.T)}\
        {sym.latex(self.D.n(2))}$'))
        display(Latex(f'$TD^*={sym.latex(Matrix(self.T*self.D).n(2))}\
        $ $^*TD$ rounded to two decimal points'))

N = letter('letterN.npy')
N.plot('N Graph (untransformed)')
display(Latex(f'$T={sym.latex(N.T)}$'))
N.report()
```



$$T = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

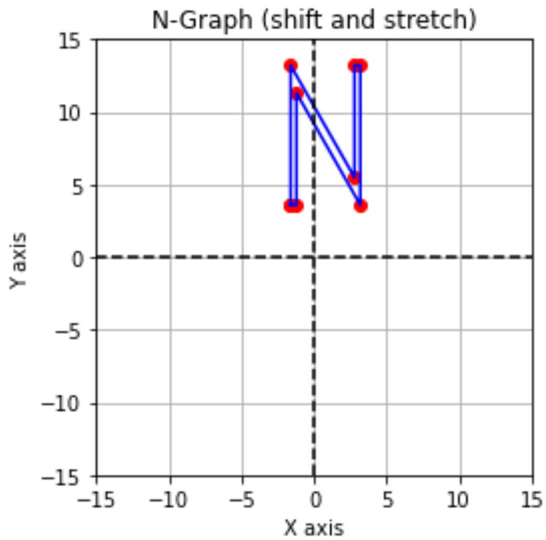
$$TD = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 0.5 & 0.5 & 6.0 & 6.0 & 5.5 & 5.5 & 0 & 0 \\ 0 & 0 & 6.4 & 0 & 8.0 & 8.0 & 1.6 & 8.0 & 0 \\ 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 \end{bmatrix}$$

$$TD^* = \begin{bmatrix} 0 & 0.5 & 0.5 & 6.0 & 6.0 & 5.5 & 5.5 & 0 & 0 \\ 0 & 0 & 6.4 & 0 & 8.0 & 8.0 & 1.6 & 8.0 & 0 \\ 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 \end{bmatrix}$$ $^*TD$ rounded to two decimal points

(a) Translate by $(-2, 3)$, and then scale the x-coordinate by 0.8 and the y-coordinate by 1.2

In [137]:

```
# see also: Ch2.7 P4E in Lay text
T1 = Matrix([[1,0,-2],[0,1,3],[0,0,1]])
T2 = Matrix([[0.8,0,0],[0,1.2,0],[0,0,1]])

N.eye() # clear transforms
N.dot(T1)
N.dot(T2)
N.plot('N-Graph (shift and stretch)')
display(Latex(f'$T={sym.latex(T2)}{sym.latex(T1)}={sym.latex(T2*T1)}$'))
N.report()
```



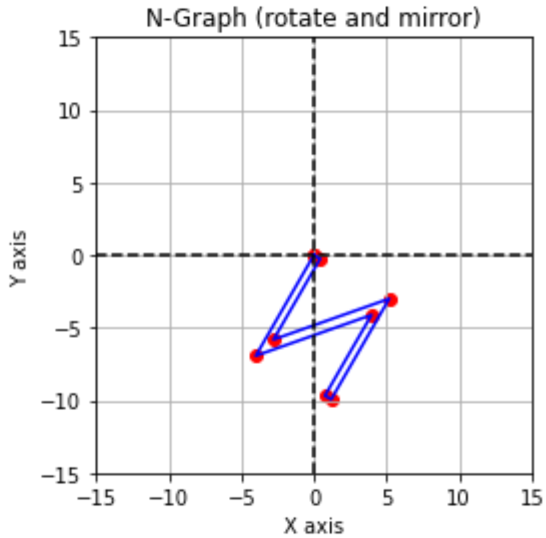
$$T=\begin{bmatrix}0.8 & 0 & 0\\ 0 & 1.2 & 0\\ 0 & 0 & 1\end{bmatrix}\begin{bmatrix}1 & 0 & -2\\ 0 & 1 & 3\\ 0 & 0 & 1\end{bmatrix}=\begin{bmatrix}0.8 & 0 & -1.6\\ 0 & 1.2 & 3.6\\ 0 & 0 & 1\end{bmatrix}$$

$$TD=\begin{bmatrix}0.8 & 0 & -1.6\\ 0 & 1.2 & 3.6\\ 0 & 0 & 1\end{bmatrix}\begin{bmatrix}0 & 0.5 & 0.5 & 6.0 & 6.0 & 5.5 & 5.5 & 0 & 0\\ 0 & 0 & 6.4 & 0 & 8.0 & 8.0 & 1.6 & 8.0 & 0\\ 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0\end{bmatrix}$$

$TD^{*}=\begin{bmatrix}-1.6 & -1.2 & -1.2 & 3.2 & 3.2 & 2.8 & 2.8 & -1.6 & -1.6\\ 3.6 & 3.6 & 11.0 & 3.6 & 13.0 & 13.0 & 5.5 & 13.0 & 3.6\\ 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0\end{bmatrix}$ $^{*}TD$ rounded to two decimal points

(b) Rotate points $\frac{\pi}{6}$, and then reflect through the x-axis.

In [136]:
```
# use syms to make pretty
theta = sym.pi/6
T1 = Matrix([[sym.cos(theta), -sym.sin(theta),0], [sym.sin(theta), sym.cos(theta),0],[0,0,1]]) #rot
T2 = Matrix([[1,0,0],[0,-1,0],[0,0,1]]) # flip y

N.eye() # clear transforms
N.dot(T1) # stack on a transform
N.dot(T2) # and again
N.plot('N-Graph (rotate and mirror)')
display(Latex(f'$T={sym.latex(T2)}{sym.latex(T1)}={sym.latex(T2*T1)}$'))
N.report()
```



$T=\begin{bmatrix}1 & 0 & 0\\0 & -1 & 0\\0 & 0 & 1\end{bmatrix}\begin{bmatrix}\frac{\sqrt{3}}{2} & -\frac{1}{2} & 0\\\frac{1}{2} & \frac{\sqrt{3}}{2} & 0\\0 & 0 & 1\end{bmatrix}=\begin{bmatrix}\frac{\sqrt{3}}{2} & -\frac{1}{2} & 0\\-\frac{1}{2} & -\frac{\sqrt{3}}{2} & 0\\0 & 0 & 1\end{bmatrix}$

$TD=\begin{bmatrix}\frac{\sqrt{3}}{2} & -\frac{1}{2} & 0\\-\frac{1}{2} & -\frac{\sqrt{3}}{2} & 0\\0 & 0 & 1\end{bmatrix}\begin{bmatrix}0 & 0.5 & 0.5 & 6.0 & 6.0 & 5.5 & 5.5 & 0 & 0\\0 & 0 & 6.4 & 0 & 8.0 & 8.0 & 1.6 & 8.0 & 0\\1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0\end{bmatrix}$

$TD^{*}=\begin{bmatrix}0 & 0.43 & -2.8 & 5.2 & 1.2 & 0.76 & 4.0 & -4.0 & 0\\0 & -0.25 & -5.8 & -3.0 & -9.9 & -9.7 & -4.1 & -6.9 & 0\\1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0\end{bmatrix}$ $^{*}TD$ rounded to two decimal points

Appendix 0. The Matrix Alphabet

sym	matrix	sym	matrix
A	Any Matrix	P	Permutation Matrix
B	Basis Matrix	P	Projection Matrix
C	Cofactor Matrix	Q	Orthogonal Matrix
D	Diagonal Matrix	R	Upper Triangular Matrix
E	Elimination Matrix	R	Reduced Echelon Matrix
F	Fourier Matrix	S	Symmetric Matrix
H	Hadamard Matrix	T	Linear Transformation
I	Identity Matrix	U	Upper Triangular Matrix
J	Jordan Matrix	U	Left Singular Vectors
K	Stiffness Matrix	V	Right Singular Vectors
L	Lower Triangular Matrix	X	Eigenvector Matrix
M	Markov Matrix	Λ	Eigenvalue Matrix
N	Nullspace Matrix	Σ	Singular Value Matrix

Linear Algebra by Gilbert Strang