In-Depth Notes on Linear Transformations with Working NumPy Implementations

1 Definition of Linear Transformation

A **linear transformation** T between vector spaces V and W over the same field (usually $\mathbb{R}$) is a function

$$T : V \to W$$

that satisfies for all $\mathbf{u}, \mathbf{v} \in V$ and scalars $c \in \mathbb{R}$:

$$\begin{cases} T(\mathbf{u} + \mathbf{v}) = T(\mathbf{u}) + T(\mathbf{v}) \\ T(c\mathbf{v}) = cT(\mathbf{v}) \end{cases}$$

This means T preserves vector addition and scalar multiplication.

NumPy Implementation: Basic Linear Transformation

Represent vectors and matrices as NumPy arrays and apply $T(\mathbf{x}) = A\mathbf{x}$ via matrix multiplication:

```
import numpy as np

# Define transformation matrix A (e.g., 2x2)
A = np.array([[2, -1],
              [3,  4]])

# Define vector x in R^2
x = np.array([1, 2])

# Apply linear transformation T(x) = A @ x
Tx = A @ x  # or np.matmul(A, x)

print("T(x) =", Tx)
```

2 Matrix Representation of a Linear Transformation

Given $T : \mathbb{R}^n \to \mathbb{R}^m$, T can be represented by an $m \times n$ matrix A:

$$T(\mathbf{x}) = A\mathbf{x}.$$

The columns of A are the images of the standard basis vectors of $\mathbb{R}^n$.

Example: Constructing A from Basis Images

If

$$T(\mathbf{e}_1) = \begin{bmatrix} 2 \\ 3 \end{bmatrix}, \quad T(\mathbf{e}_2) = \begin{bmatrix} -1 \\ 4 \end{bmatrix},$$

then

$$A = \begin{bmatrix} 2 & -1 \\ 3 & 4 \end{bmatrix}.$$

3 Properties of Linear Transformations

3.1 Preservation of Structure

Linear transformations preserve the operations of vector addition and scalar multiplication, thus preserving:

- **Zero vector:** $T(\mathbf{0}) = \mathbf{0}$.
- **Lines and planes:** Images of lines and planes are lines and planes (or points).
- **Linear dependence and independence:** If vectors are linearly dependent, their images are also linearly dependent.

3.2 Kernel and Image

- **Kernel (Null space):**
$$\ker(T) = \{\mathbf{v} \in V : T(\mathbf{v}) = \mathbf{0}\}.$$
It is a subspace of V.
- **Image (Range):**
$$\mathrm{Im}(T) = \{T(\mathbf{v}) : \mathbf{v} \in V\} \subseteq W,$$
also a subspace of W.

3.3 Rank-Nullity Theorem

For finite-dimensional V,

$$\dim(V) = \dim(\ker(T)) + \dim(\mathrm{Im}(T)).$$

4 Examples of Linear Transformations

4.1 Scaling

Scaling by factors k_x, k_y in $\mathbb{R}^2$:

$$T(x, y) = (k_x x, k_y y), \quad A = \begin{bmatrix} k_x & 0 \\ 0 & k_y \end{bmatrix}.$$

```
kx, ky = 1.5, 0.5
A = np.array([[kx, 0],
              [0, ky]])

v = np.array([2, 3])
Tv = A @ v
print("Scaled vector:", Tv)
```

4.2 Rotation

Rotation by angle θ in $\mathbb{R}^2$:

$$R_\theta = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}.$$

```python
theta = np.pi / 4  # 45 degrees
R = np.array([[np.cos(theta), -np.sin(theta)],
              [np.sin(theta),  np.cos(theta)]])

v = np.array([1, 0])
rotated_v = R @ v
print("Rotated vector:", rotated_v)
```

4.3 Reflection about x-axis

$$A = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}.$$

```python
A = np.array([[1, 0],
              [0, -1]])

v = np.array([3, 4])
reflected_v = A @ v
print("Reflected vector:", reflected_v)
```

5 Composition of Linear Transformations

If $T_1(\mathbf{x}) = A\mathbf{x}$ and $T_2(\mathbf{x}) = B\mathbf{x}$, then

$$(T_2 \circ T_1)(\mathbf{x}) = B(A\mathbf{x}) = (BA)\mathbf{x}.$$

```python
A = np.array([[1, 2],
              [3, 4]])
B = np.array([[0, 1],
              [1, 0]])

x = np.array([1, 1])

# Apply T1 then T2
result = B @ (A @ x)
print("Result of T2(T1(x)):", result)

# Equivalent to (B @ A) @ x
composed_matrix = B @ A
result2 = composed_matrix @ x
print("Result using composed matrix:", result2)
```

6 Summary

- Linear transformations preserve vector addition and scalar multiplication.
- They can be represented by matrices, and applied via matrix multiplication.
- Kernel and image provide key structural insights.

- Common transformations include scaling, rotation, and reflection.
- Composition corresponds to matrix multiplication.