CS 390E: Geometry Processing Assignment 01

Linear Algebra

Anna Frühstück

Sunday 5th February, 2017

1 Matrix norm

The matrix norm can be interpreted as the maximum amount of "stretching" that a matrix A can do to a vector $x \to Ax$. The operator norm corresponding to the vector norm $\|\cdot\|$ is defined as

$$\|A\|_p = \sup_{x\neq 0} \frac{\|Ax\|_p}{\|x\|_p} \text{ for } p \geq 1$$

1.1 Matrix norm inequalities

Subadditivity

Prove $\|A+B\| \leq \|A\| + \|B\|$*:*

$$\|A+B\| = \max \|(A+B)x\| = \max_{\|x=1\|} \|Ax+Bx\|$$

$$\max_{\|x=1\|} \|Ax+Bx\| \leq \max_{\|x=1\|} \|Ax\| + \|Bx\| \leq \max_{\|x=1\|} \|Ax\| + \max_{\|x=1\|} \|Bx\| = \|A\| + \|B\|$$

Subordinance

Prove $\|Ax\| \leq \|A\|\|x\|$*:*
Definition of the operator norm: $\|A\| = \sup \frac{\|Ax\|}{\|x\|}$
multiply right and left side by $\|x\|$
$\to$ for the x corresponding to the largest norm: $\|x\|\|A\| = \|Ax\|$
$\to$ in general: $\|x\|\|A\| \leq \|Ax\|$

Submultiplicativity

Prove $\|AB\| \leq \|A\|\|B\|$*:*

$$\|AB\| = \max_{x\neq 0} \frac{\|ABx\|}{\|x\|} = \max_{x\neq 0,\ Bx\neq 0} \frac{\|A(Bx)\|}{\|Bx\|}\frac{\|Bx\|}{\|x\|} \leq \max_{y\neq 0} \frac{\|Ay\|}{\|y\|} \max_{x\neq 0} \frac{\|Bx\|}{\|x\|} = \|A\|\|B\|$$

2 Orthogonality

Given a set of orthogonal vectors $\{v_1, v_2, ..., v_n\} \in \mathbb{R}_m$ *with* $m \geq n$*, i.e.,* $v_i^T v_j = 0$*, for* $i \neq j$*, then they are linearly independent. Why?*

If v_i^T and v_j are orthogonal, $v_i^T v_j = 0$.
Two vectors v_i and v_j are linearly independent if and only if the following property is true:

If c_1 and c_2 are scalars such that $c_1 v_i + c_2 v_j = 0$ then $c_1 = c_2 = 0$.

Suppose $c_1 v_1 + c_2 v_2 + ... + c_n v_n = 0$, then multiply equation with v_i:

$$c_1 \underbrace{v_1 \cdot v_i}_{=0} + c_2 \underbrace{v_2 \cdot v_i}_{=0} + ... + c_i v_i \cdot v_i + ... + c_n \underbrace{v_n \cdot v_i}_{=0} = 0 \cdot v_i$$

All terms on the left side go to zero except one:

$$c_i \|v_i\|^2 = 0$$

$\rightarrow c_i = 0$, therefore $c_1 = ... = c_n = 0$

3 Jordan Normal Form

The Jordan normal form is a special type of block matrix. The Jordan matrix has the eigenvalues on the main diagonal and 1s on the superdiagonal for each Jordan block. The shape of the Jordan normal form follows this structure, each subblock of the matrix shown here is called a Jordan block:

$$\begin{bmatrix}
\lambda_1 & 1 & & & & & & & \\
 & \lambda_1 & 1 & & & & & & \\
 & & \lambda_1 & & & & & & \\
 & & & \lambda_2 & 1 & & & & \\
 & & & & \lambda_2 & & & & \\
 & & & & & \lambda_3 & & & \\
 & & & & & & \ddots & & \\
 & & & & & & & \lambda_n & 1 \\
 & & & & & & & & \lambda_n
\end{bmatrix}$$

While not all $n \times n$ matrices are diagonalizable (they are only if A has n linearly independent eigenvectors), in general an invertible matrix P can be found such that $A = PJP^{-1}$, where J is in Jordan form, and as such "almost diagonal".

- The eigenvalues of J (and therefore A) are on the diagonal.

- The geometric multiplicity of an eigenvalue λ_i (the maximum number of linearly independent eigenvectors associated with this eigenvalue) is equal to the number of Jordan blocks corresponding to λ_i.
- The sum of the sizes of all Jordan blocks corresponding to an eigenvalue λ_i equals to the algebraic multiplicity of J (i.e. the repeated occurrence of λ_i in the characteristic polynomial of A).

4 2D Mesh viewer

Start MATLAB implementation from `matrix_transformations\main.m` (for custom exploration) or `matrix_transformations\auto2D.m` (for animated demonstration).
Special matrices can be input using the functions in `matrix.m` by specifying the dimensionality, the matrix type and the matrix parameters e.g. `m = matrix(2, 'rotation', 0.8)` to generate a 2D rotation matrix. Note that homogeneous transformation matrices require a dimensionality of 3 for 2D transformations.
Any matrix transformation can be applied to the current data stored in the `points` variable by calling the function `points = transformPoints(view, points, m)` where `view` is the current figure that needs to be updated after the transformation, `points` is the matrix storing the data and `m` is the matrix transformation to be applied to each data point.
Apply the function `analyzeMatrix(m)` to any matrix to inspect its determinant, eigenvalues, eigenvectors, Jordan matrix and condition number.

Rotation

$$\begin{bmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{bmatrix}$$

```
m = matrix(2, 'rotation', a)
```

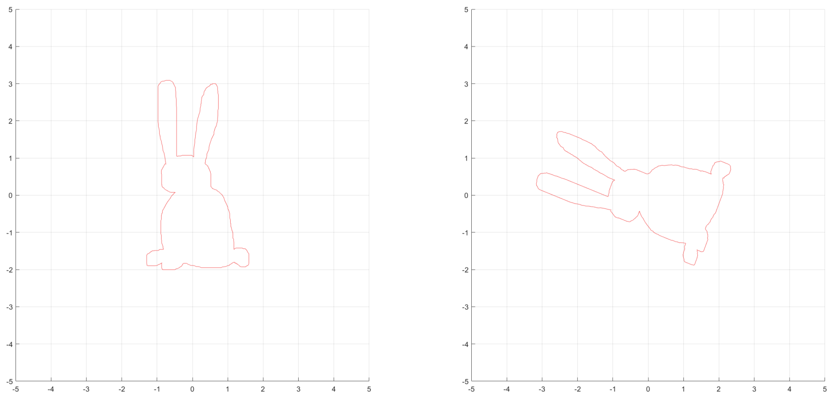

Figure 1: Rotation by an angle of 1.2 radians around the center

Shear

Shear along x axis *Shear along y axis*

$$\begin{bmatrix} 1 & f \\ 0 & 1 \end{bmatrix} \qquad \begin{bmatrix} 1 & 0 \\ f & 1 \end{bmatrix}$$

```
m = matrix(2, 'shear', axis, f)
```

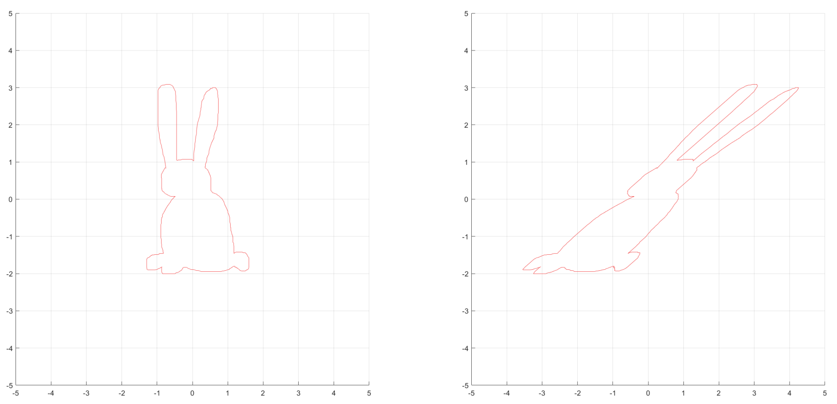

Figure 2: Shear by a factor of 1.2 along the x axis

Mirror

for reflection along axis defined by unit vector $[x, y]$

$$\begin{bmatrix} x^2-y^2 & 2xy \\ 2xy & y^2-x^2 \end{bmatrix}$$

```
m = matrix(2, 'reflection', [x y])
```

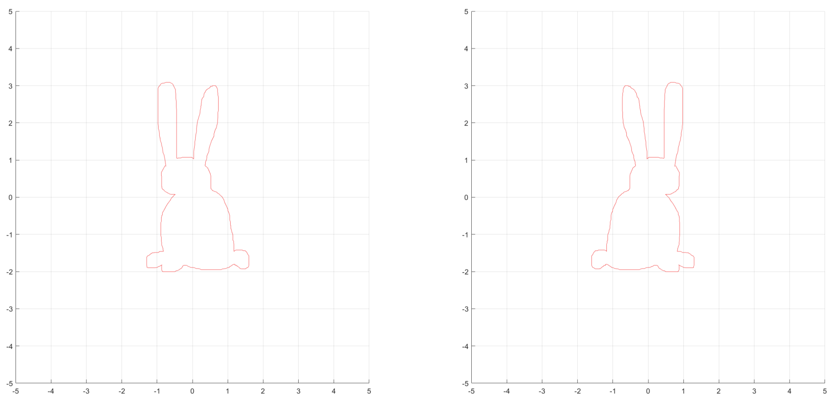

Figure 3: Reflect across y axis (vector $[0, 1]$)

Scaling

$$\begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix}$$

```
m = matrix(2, 'scale', s_x, s_y)
```

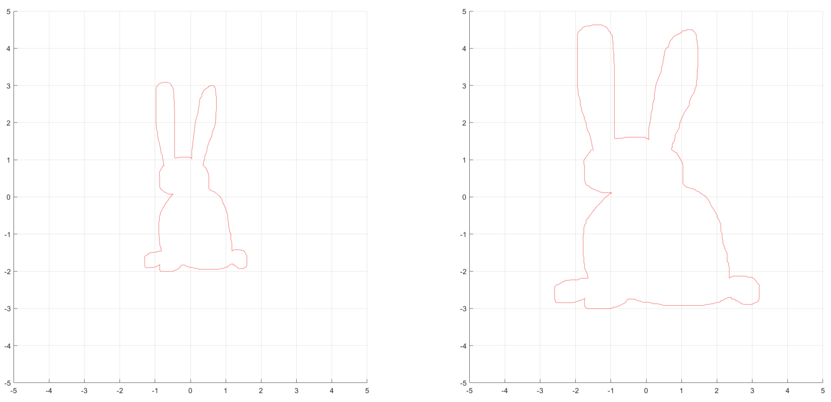

Figure 4: Anisotropic scale by factors $s_x = 2$ and $s_y = 1.5$

Translation

requires 3×3 homogeneous coordinates.

$$\begin{bmatrix} 1 & 0 & d_x \\ 0 & 1 & d_y \\ 0 & 0 & 1 \end{bmatrix}$$

```
m = matrix(3, 'translation', d_x, d_y)
```

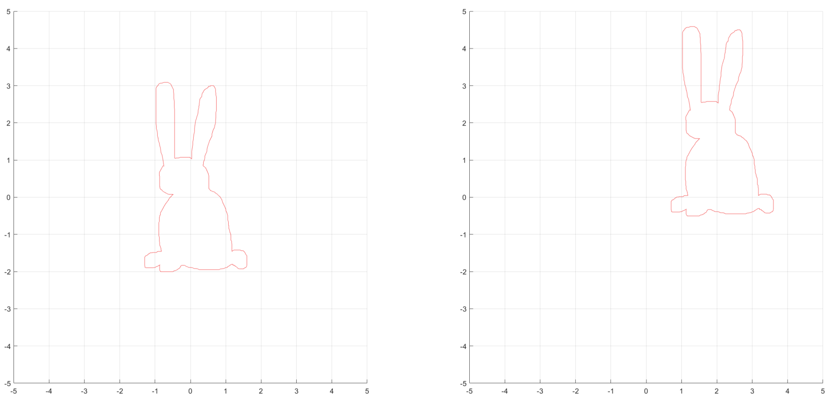

Figure 5: Translation by offsets $d_x = 2$ and $d_y = 1.5$

5 3D Mesh viewer

Start MATLAB implementation from `matrix_transformations\main.m` (for custom exploration) or `matrix_transformations\auto3D.m` (for animated demonstration). Special matrices can be input using the functions in `matrix.m` by specifying the dimensionality, the matrix type and the matrix parameters e.g. `m = matrix(3, 'rotation', 'x', 0.8)` to generate a 3D rotation matrix. Note that homogeneous transformation matrices require a dimensionality of 4 for 3D transformations.
Any matrix transformation can be applied to the current data stored in the `points` variable by calling the function `points = transformPoints(view, points, m)` where `view` is the current figure that needs to be updated after the transformation, `points` is the matrix storing the data and `m` is the matrix transformation to be applied to each data point.
Apply the function `analyzeMatrix(m)` to any matrix to inspect its determinant, eigenvalues, eigenvectors, Jordan matrix and condition number.

Rotation

Rotate around x axis

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{bmatrix}$$

Rotate around y axis

$$\begin{bmatrix} \cos\alpha & 0 & \sin\alpha \\ 0 & 1 & 0 \\ -\sin\alpha & 0 & \cos\alpha \end{bmatrix}$$

Rotate around z axis

$$\begin{bmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

```
m = matrix(3, 'rotation', axis, a)
```

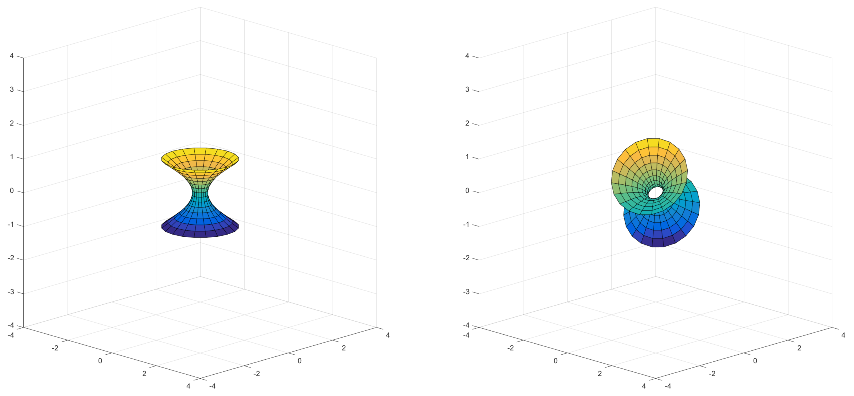

Figure 6: Rotation around x axis by 0.7 radians and y axis by 0.5 radians.

Shear

Shear along x axis *Shear along y axis* *Shear along z axis*

$$\begin{bmatrix} 1 & 0 & 0 \\ f_1 & 1 & 0 \\ f_2 & 0 & 1 \end{bmatrix} \qquad \begin{bmatrix} 1 & f_1 & 0 \\ 0 & 1 & 0 \\ 0 & f_2 & 1 \end{bmatrix} \qquad \begin{bmatrix} 1 & 0 & f_1 \\ 0 & 1 & f_2 \\ 0 & 0 & 1 \end{bmatrix}$$

```
m = matrix(3, ’shear’, axis, f_1, f_2)
```

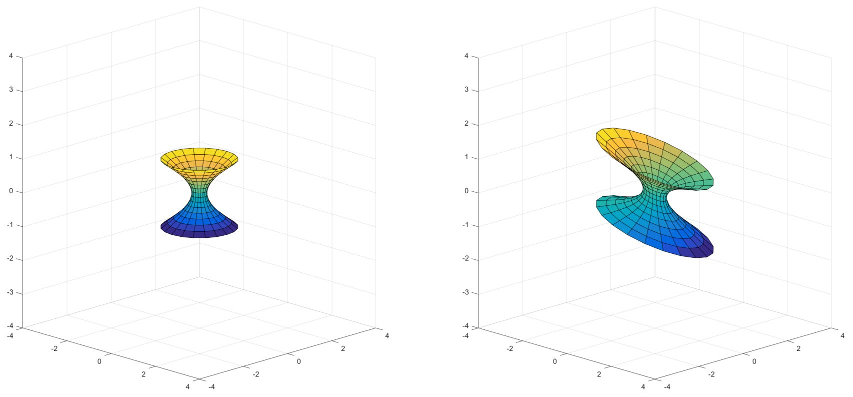

Figure 7: Shear along x axis by factors $f_1 = 0.9$ and $f_2 = -0.7$

Mirror

Reflect x component *Reflect y component* *Reflect z component*

$$\begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{bmatrix}$$

```
m = matrix(3, ’reflection’, axis)
```

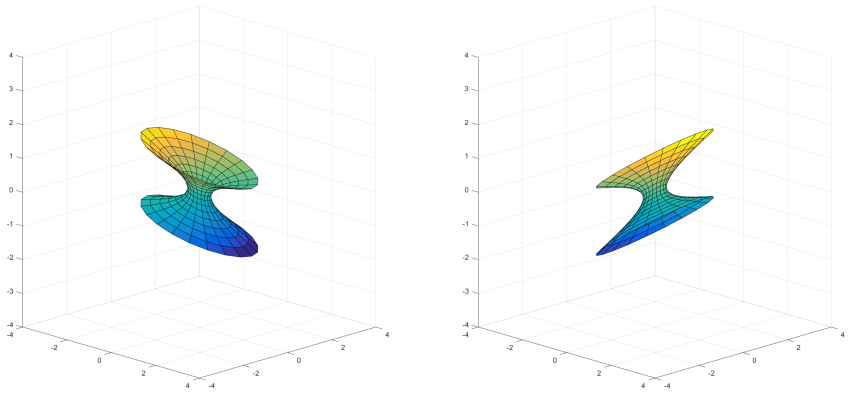

Figure 8: Reflect sheared body across z axis

Scaling

Any diagonal matrix applies an (anisotropic) scaling transformation.

$$\begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & s_z \end{bmatrix}$$

```
m = matrix(3, 'scale', s_x, s_y, s_z)
```

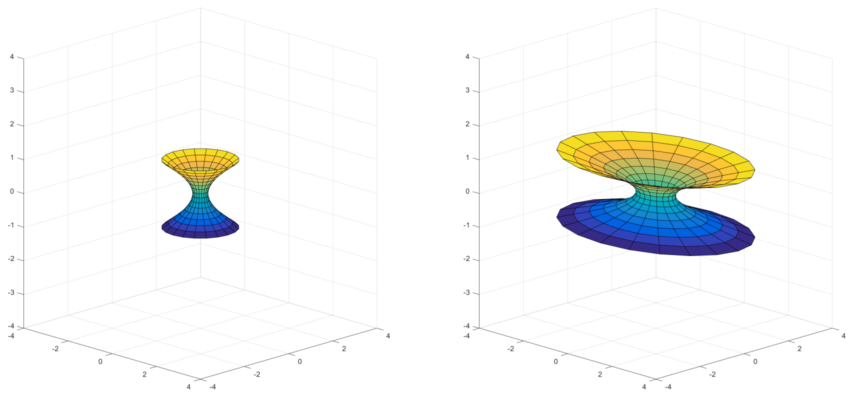

Figure 9: Anisotropic scaling by factors $s_x = 3$, $s_y = 2$ and $s_z = 1$

Translation

requires 4×4 homogeneous coordinates.

$$\begin{bmatrix} 1 & 0 & 0 & d_x \\ 0 & 1 & 0 & d_y \\ 0 & 0 & 1 & d_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

```
m = matrix(4, 'translation', d_x, d_y, d_z)
```

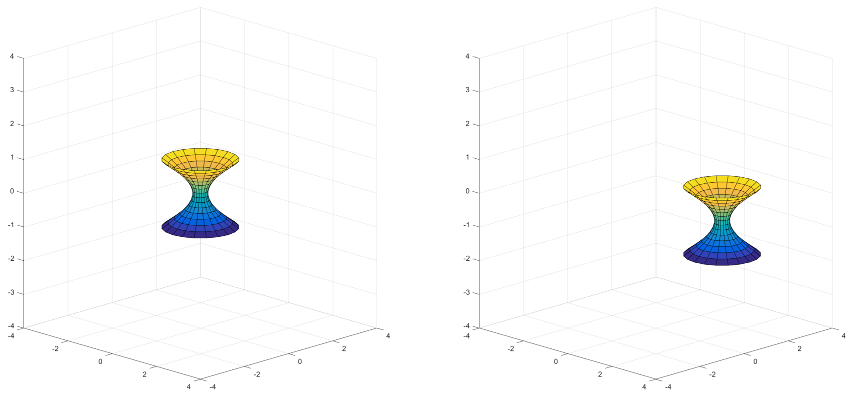

Figure 10: Translation by offsets $d_x = 1$, $d_y = 2$ and $d_z = -1$

Perspective Projection

requires 4×4 homogeneous coordinates.

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & \frac{1}{d} & 1 \end{bmatrix}$$

```
m = matrix(4, 'projection', d)
```

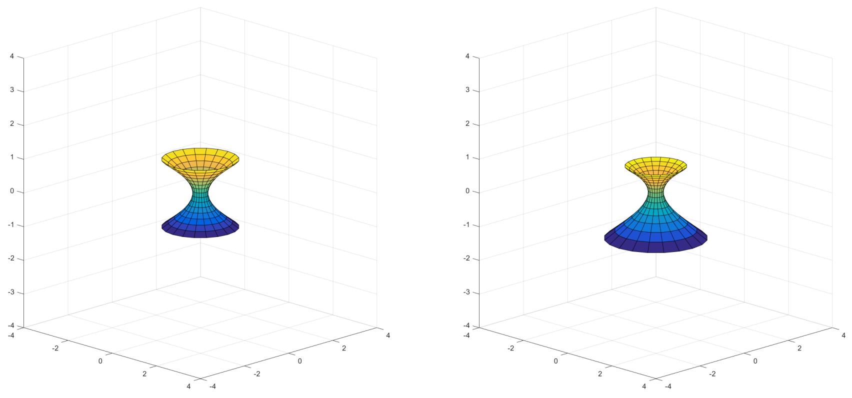

Figure 11: Projection by factor $d = 4$

Permutation Matrix

A permutation matrix contains exactly one 1 per row and column. When applied, a permutation matrix interchanges the axes of a vector. Example:

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

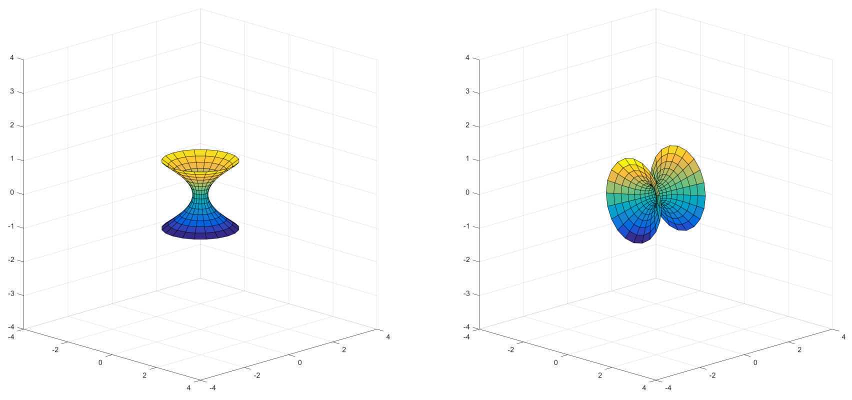

Figure 12: Permutation using above matrix

Stochastic Transition Matrix

In a stochastic transition matrix, the rows sum up to 1. Example:

$$\begin{bmatrix} 0 & \frac{1}{2} & \frac{1}{2} \\ \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ 1 & 0 & 0 \end{bmatrix}$$

Upper Triangular Matrix

An upper triangular matrix has only zero entries below the main diagonal, and arbitrary entries above. Example:

$$\begin{bmatrix} 2 & 3 & 2 \\ 0 & 1 & -2 \\ 0 & 0 & 4 \end{bmatrix}$$

Householder Matrix

A householder matrix describes the transformation of a vector when reflected through the origin with respect to a (hyper-)plane represented by its normal vector v. It can be constructed for a given (unit) vector v as follows:

$$P = I - 2 \cdot v \cdot v^T$$

. P is always symmetric since

$$P^T = (I - 2 \cdot v \cdot v^T)^T = I - 2 \cdot v \cdot v^T = P$$

and is its own inverse since

$$PP = (I - 2 \cdot v \cdot v^T)(I - 2 \cdot v \cdot v^T) = I - 4 \cdot v \cdot v^T + 4 \cdot v \cdot v^T v \cdot v^T = I - 4 \cdot v \cdot v^T + 4 \cdot v \cdot v^T = I$$

Thus, Householder matrices are orthogonal ($P^{-1} = P^T$).

6 Observations

Most of the listed transformations do not typically yield singular matrices, i.e. they can be reversed by applying the inverse of the matrix to the set of points. A few of the matrices can be made singular when setting all parameters to zero (e.g. a scaling matrix that consists of all zeros). The stochastic transition matrix chosen as an example above is in fact singular, because two columns were chosen such that are not linearly independent. This leads to a collapse in dimensionality in the visualized result (it becomes planar in 3D space) and can no longer be inverted.

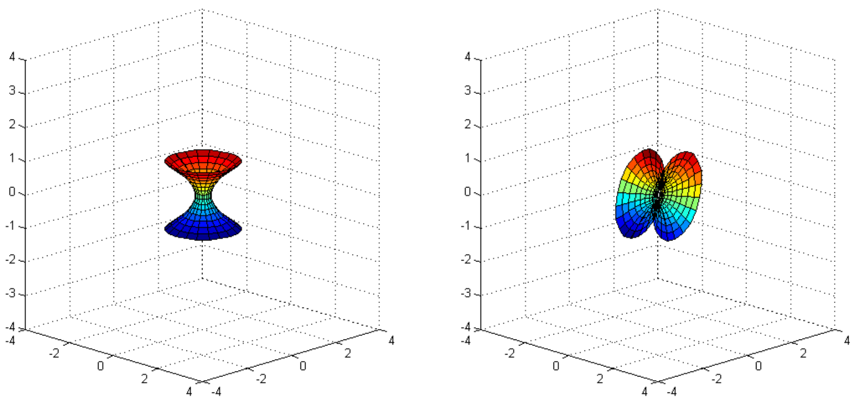

Figure 13: Applying a singular stochastic transition matrix to 3D data points.

The eigenvalues of the singular transformation matrix used in above illustration are 1, $-\frac{2}{3}$ and 0. (Therefore, the determinant is also zero.) The corresponding eigenvectors are $(-0.5774, -0.5774, -0.5774)$ $(-0.5523, -0.0921, 0.8285)$ and $(0.0000, -0.7071, 0.7071)$.