Fundamentals of Linear Algebra, Part II

Class 2. 31 August 2009

Instructor: Bhiksha Raj

Administrivia

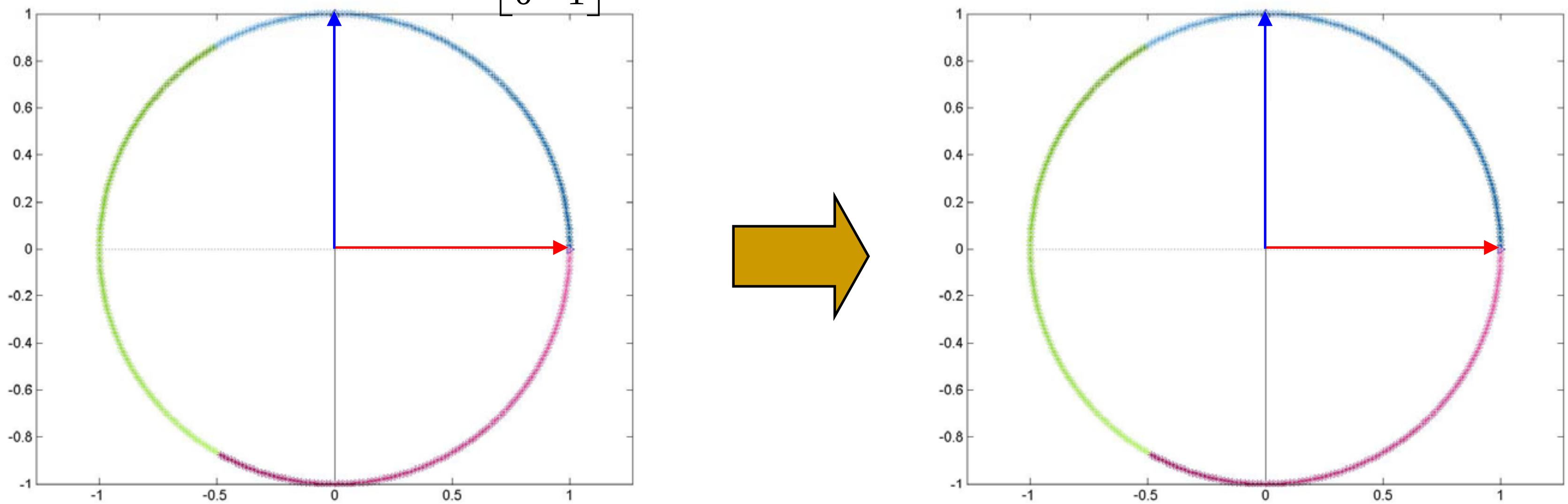
- Registration: Anyone on waitlist still?
- We have a second TA
  - Sohail Bahmani
  - sbahmani@andrew.cmu.edu
- Homework: Slightly delayed
  - Linear algebra
  - Adding some fun new problems.
  - Use the discussion lists on blackboard.andrew.cmu.edu
- **Blackboard – if you are not registered on blackboard please register**

Overview

- *Vectors and matrices*
- *Basic vector/matrix operations*
- *Vector products*
- *Matrix products*
- Various matrix types
- Matrix inversion
- Matrix interpretation
- Eigenanalysis
- Singular value decomposition

The Identity Matrix

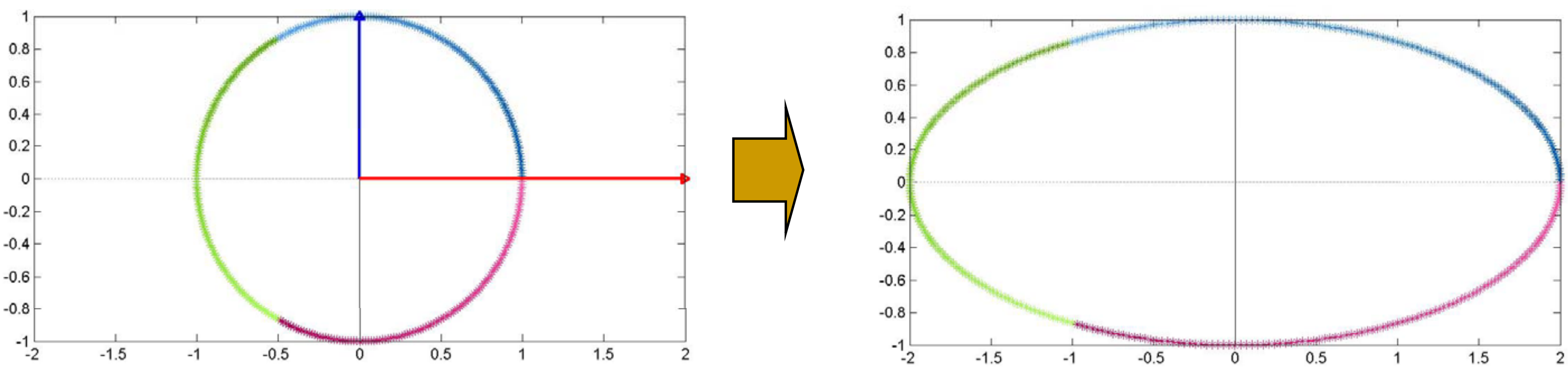
$$Y = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$



- An identity matrix is a square matrix where
  - All diagonal elements are 1.0
  - All off-diagonal elements are 0.0
- Multiplication by an identity matrix does not change vectors

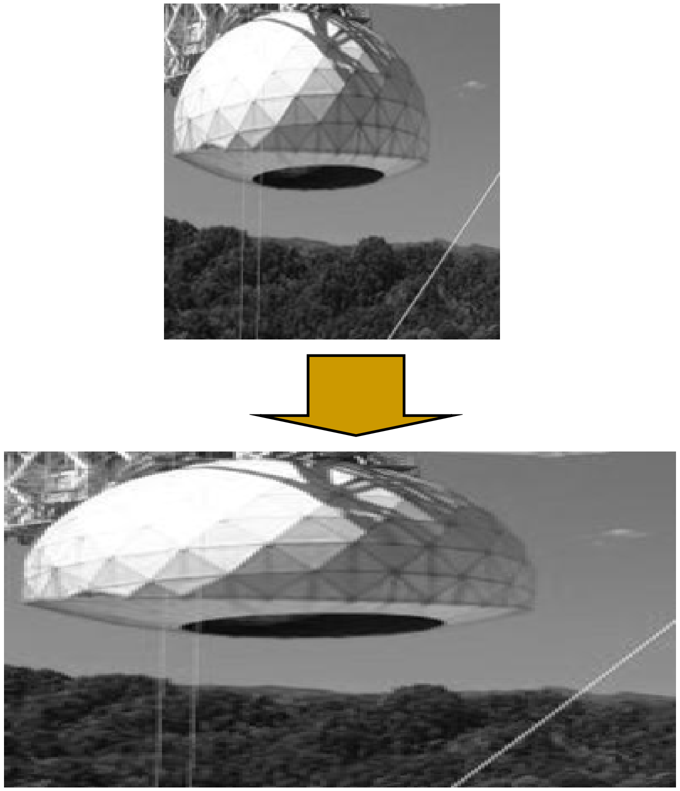
Diagonal Matrix

$$Y = \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix}$$



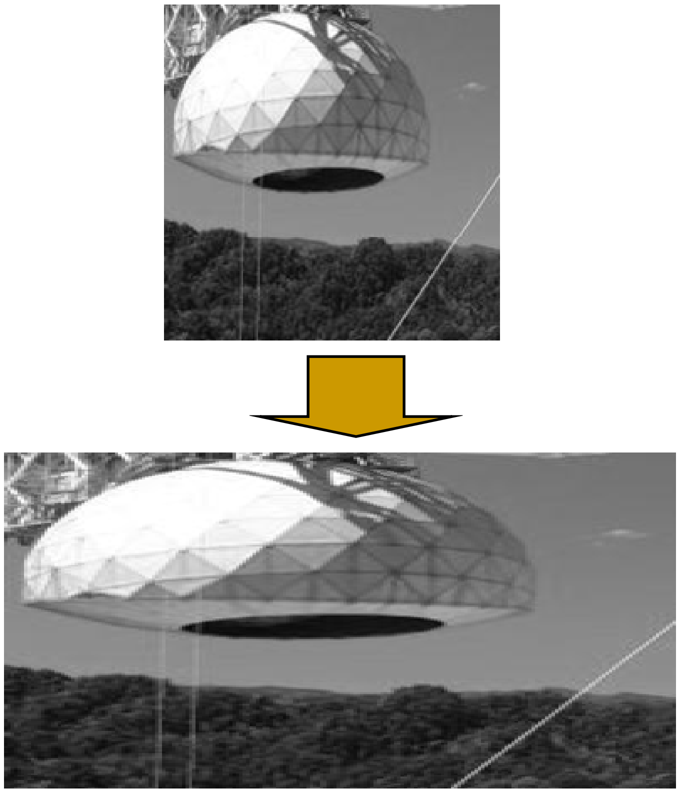
- All off-diagonal elements are zero
- Diagonal elements are non-zero
- Scales the axes
  - May flip axes

Diagonal matrix to transform images



- How?

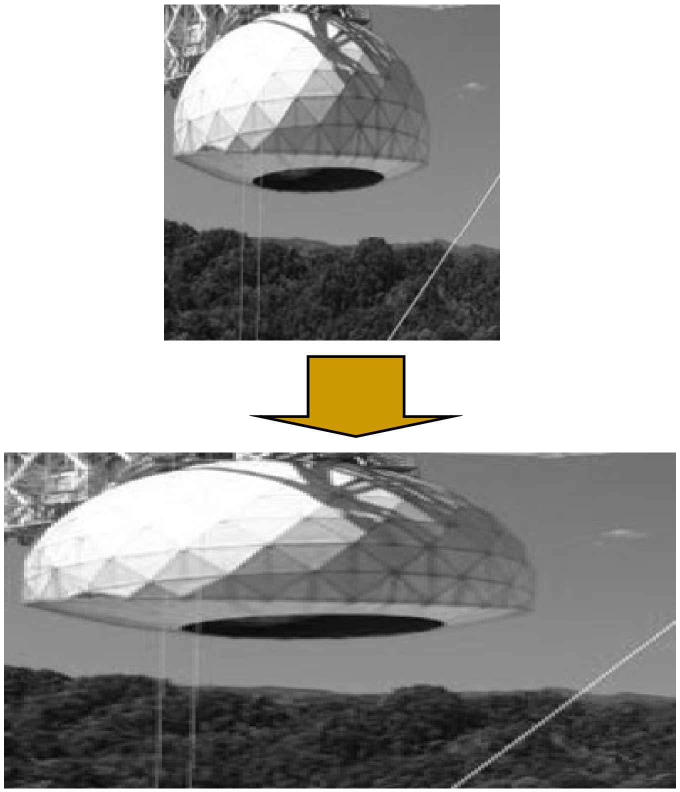
Stretching



$$\begin{bmatrix} 2 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 & . & 2 & . & 2 & 2 & . & 2 & . & 10 \\ 1 & 2 & . & 1 & . & 5 & 6 & . & 10 & . & 10 \\ 1 & 1 & . & 1 & . & 0 & 0 & . & 1 & . & 1 \end{bmatrix}$$

- Location-based representation
- Scaling matrix – only scales the X axis
  - The Y axis and pixel value are scaled by identity
- Not a good way of scaling.

Stretching



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

$$A = \begin{bmatrix} 1 & .5 & 0 & 0 & . \\ 0 & .5 & 1 & .5 & . \\ 0 & 0 & 0 & .5 & . \\ 0 & 0 & 0 & 0 & . \\ . & . & . & . & . \end{bmatrix} (N\mathrm{x}2N)$$

$$\text{Newpic} = DA$$

- Better way

Modifying color

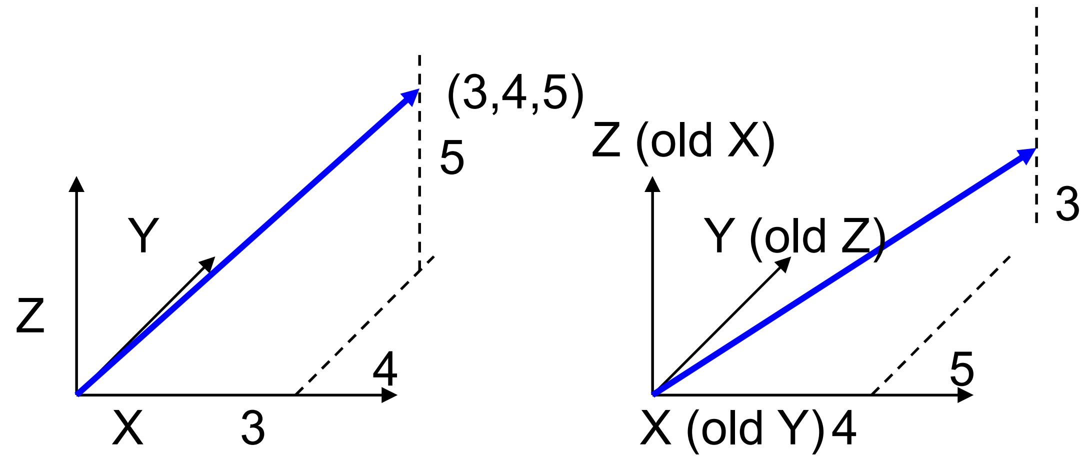
$$P = \begin{bmatrix} R & G & B \end{bmatrix}$$

$$Newpic = P \begin{bmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

- Scale only Green

Permutation Matrix

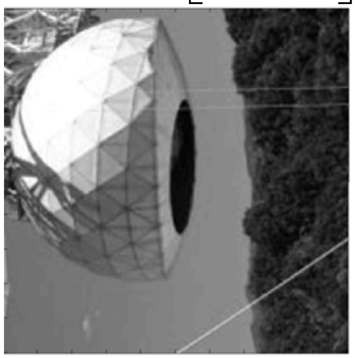
$$\begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} y \\ z \\ x \end{bmatrix}$$



- A permutation matrix simply rearranges the axes
  - The row entries are axis vectors in a different order
  - The result is a combination of rotations and reflections
- The permutation matrix effectively *permutes* the arrangement of the elements in a vector

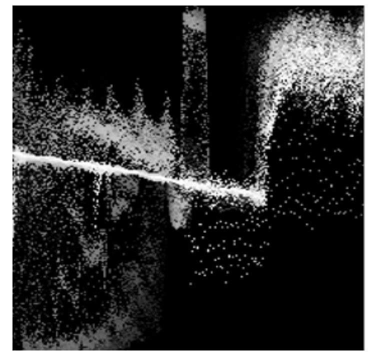
Permutation Matrix

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



$$\begin{bmatrix} 1 & 1 & . & 2 & . & 2 & 2 & . & 2 & . & 10 \\ 1 & 2 & . & 1 & . & 5 & 6 & . & 10 & . & 10 \\ 1 & 1 & . & 1 & . & 0 & 0 & . & 1 & . & 1 \end{bmatrix}$$

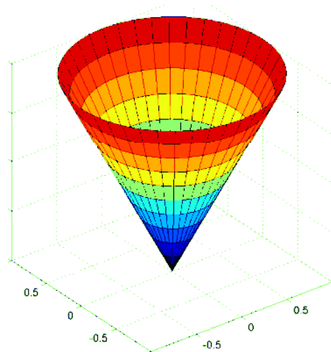
$$P = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}$$



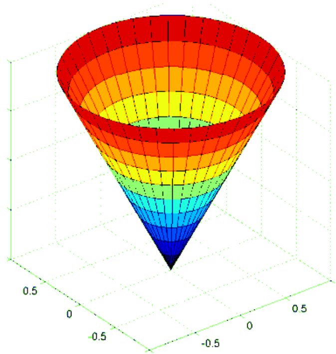
- Reflections and 90 degree rotations of images and objects

Permutation Matrix

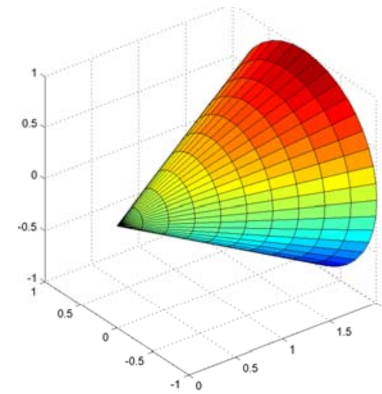
$$P = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$



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



$$\begin{bmatrix} x_1 & x_2 & . & . & x_N \\ y_1 & y_2 & . & . & y_N \\ z_1 & z_2 & . & . & z_N \end{bmatrix}$$



- Reflections and 90 degree rotations of images and objects
  - Object represented as a matrix of 3-Dimensional “position” vectors
  - Positions identify each point on the surface

Rotation Matrix

$$x' = x\cos\theta - y\sin\theta$$

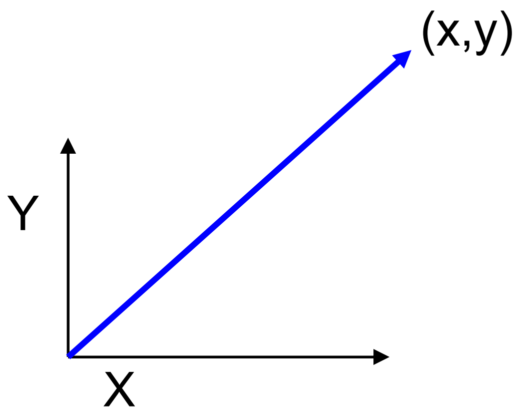
$$y' = x\sin\theta + y\cos\theta$$

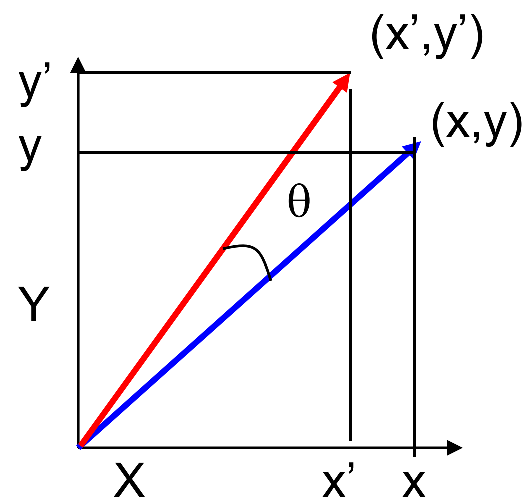
$$\mathbf{R}_\theta = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

$$X = \begin{bmatrix} x \\ y \end{bmatrix}$$

$$X_{new} = \begin{bmatrix} x' \\ y' \end{bmatrix}$$

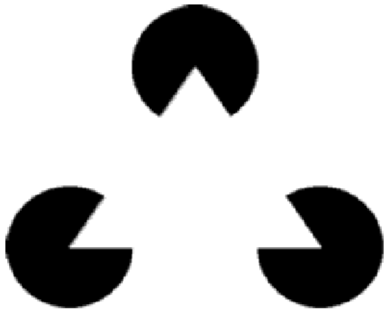
$$R_\theta X = X_{new}$$



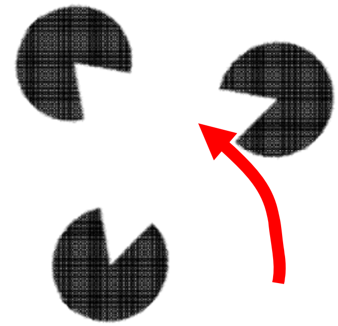


- A rotation matrix *rotates* the vector by some angle θ
- Alternately viewed, it rotates the axes
  - The new axes are at an angle θ to the old one

Rotating a picture



$$R = \begin{bmatrix} \cos 45 & -\sin 45 & 0 \\ \sin 45 & \cos 45 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

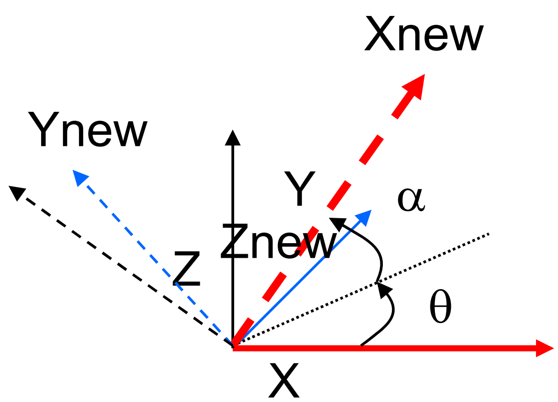


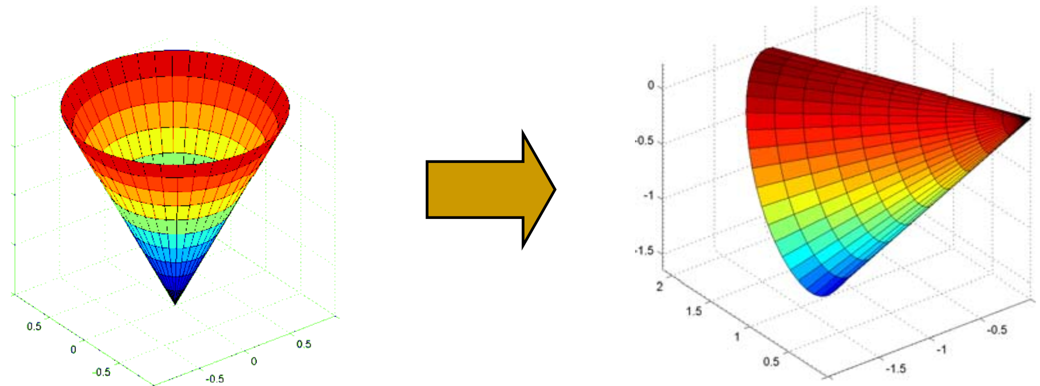
$$\begin{bmatrix} 1 & 1 & . & 2 & . & 2 & 2 & . & 2 & . & . \\ 1 & 2 & . & 1 & . & 5 & 6 & . & 10 & . & . \\ 1 & 1 & . & 1 & . & 0 & 0 & . & 1 & . & 1 \end{bmatrix}$$

$$\begin{bmatrix} 0 & -\sqrt{2} & . & \sqrt{2} & . & -3\sqrt{2} & -4\sqrt{2} & . & -8\sqrt{2} & . & . \\ \sqrt{2} & 3\sqrt{2} & . & 3\sqrt{2} & . & 7\sqrt{2} & 8\sqrt{2} & . & 12\sqrt{2} & . & . \\ 1 & 1 & . & 1 & . & 0 & 0 & . & 1 & . & 1 \end{bmatrix}$$

- Note the representation: 3-row matrix
  - Rotation only applies on the “coordinate” rows
  - The value does not change
  - Why is pacman grainy?

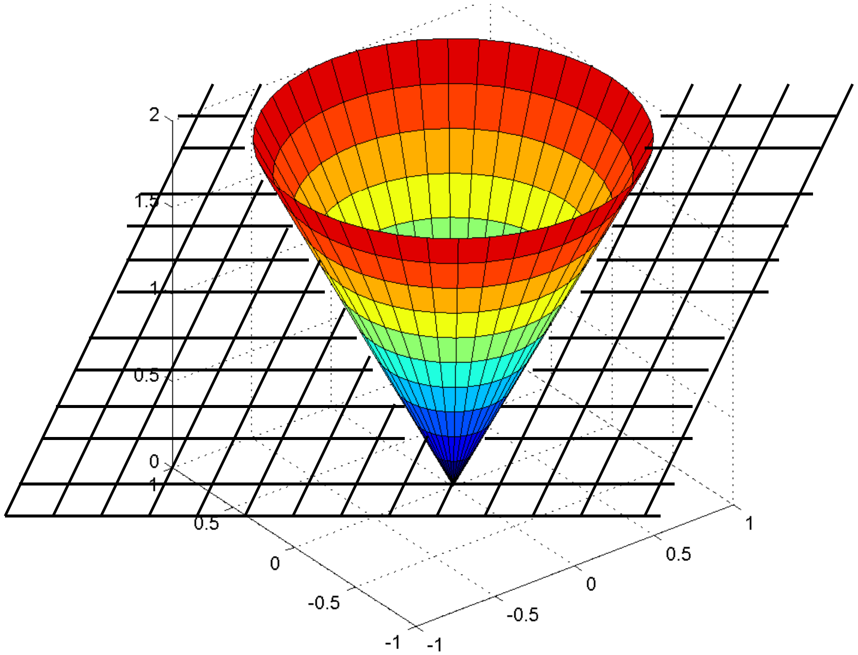
3-D Rotation

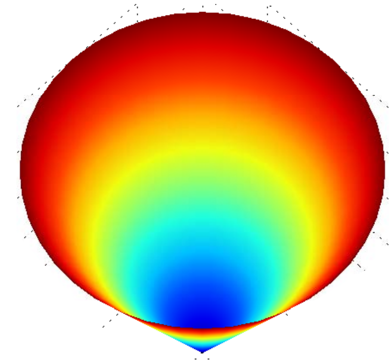




- 2 degrees of freedom
  - 2 separate angles
- What will the rotation matrix be?

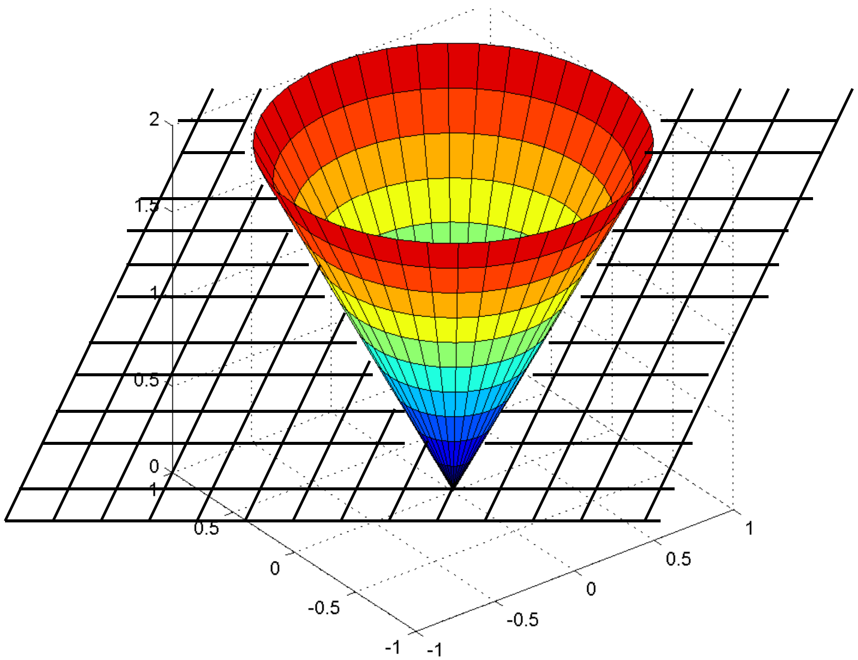
Projections

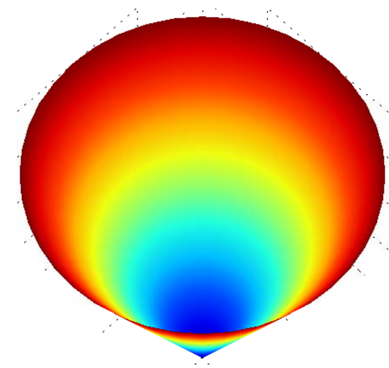




- What would we see if the cone to the left were transparent if we looked at it along the normal to the plane
  - The plane goes through the origin
  - Answer: the figure to the right
- How do we get this? Projection

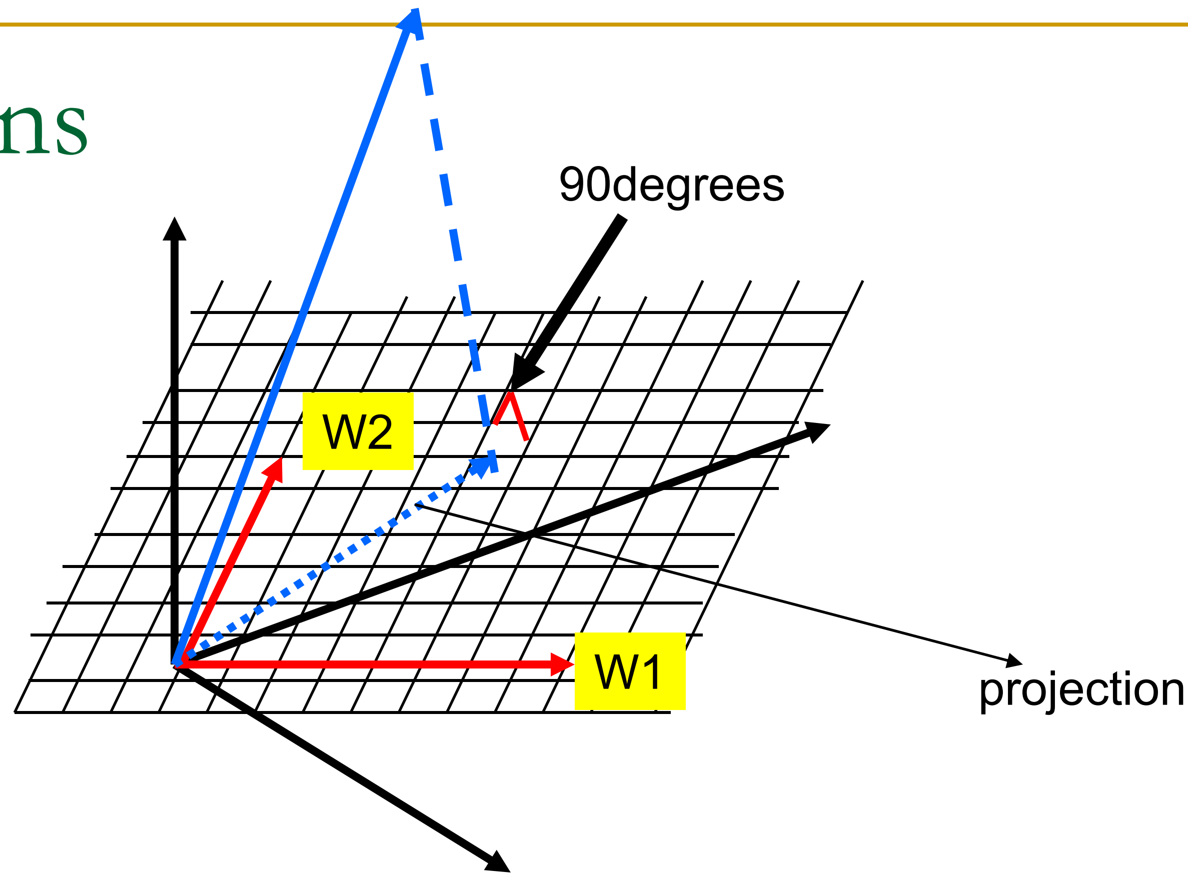
Projections





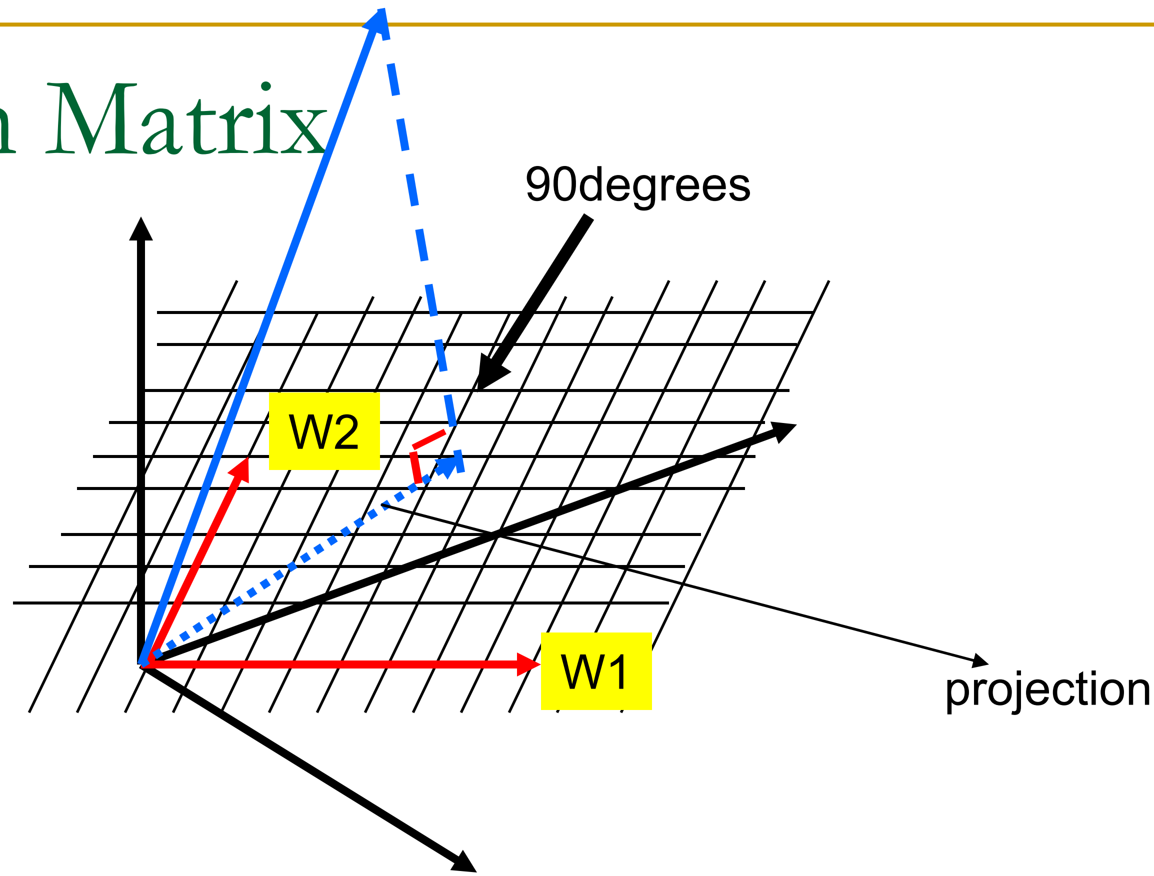
- Each pixel in the cone to the left is mapped onto to its “shadow” on the plane in the figure to the right
- The location of the pixel’s “shadow” is obtained by multiplying the vector *V* representing the pixel’s location in the first figure by a matrix *A*
    - *Shadow* (*V*)= *A V*
- The matrix *A* is a projection matrix

Projections



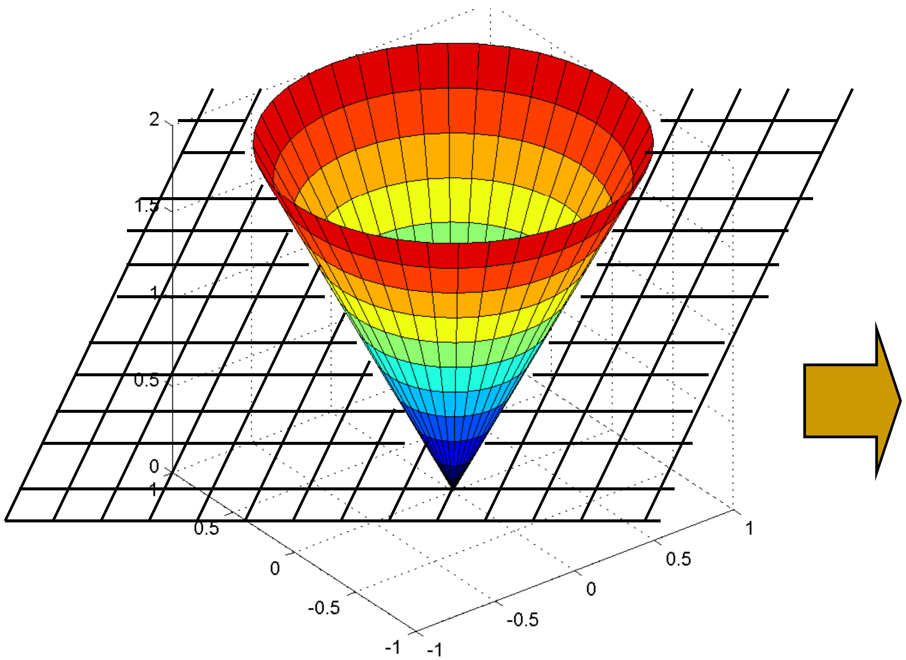
- Consider any plane specified by a set of vectors W_1, W_2..
  - Or matrix $[W_1\ W_2\ ..]$
- Any vector can be projected onto this plane by multiplying it with the projection matrix for the plane
  - The projection is the shadow

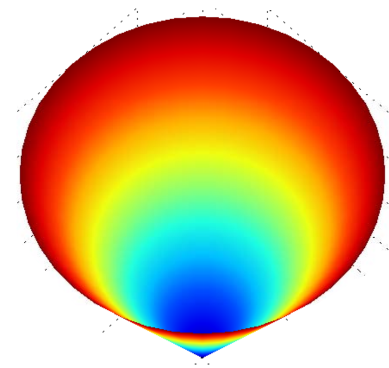
Projection Matrix



- Given a set of vectors W1, W2, which form a matrix W = [W1 W2..]
- The projection matrix that transforms any vector X to its projection on the plane is
  - $P = W (W^T W)^{-1} W^T$
    - We will visit matrix inversion shortly
- Magic – any set of vectors from the same plane that are expressed as a matrix will give you the same projection matrix
  - $P = V (V^T V)^{-1} V^T$

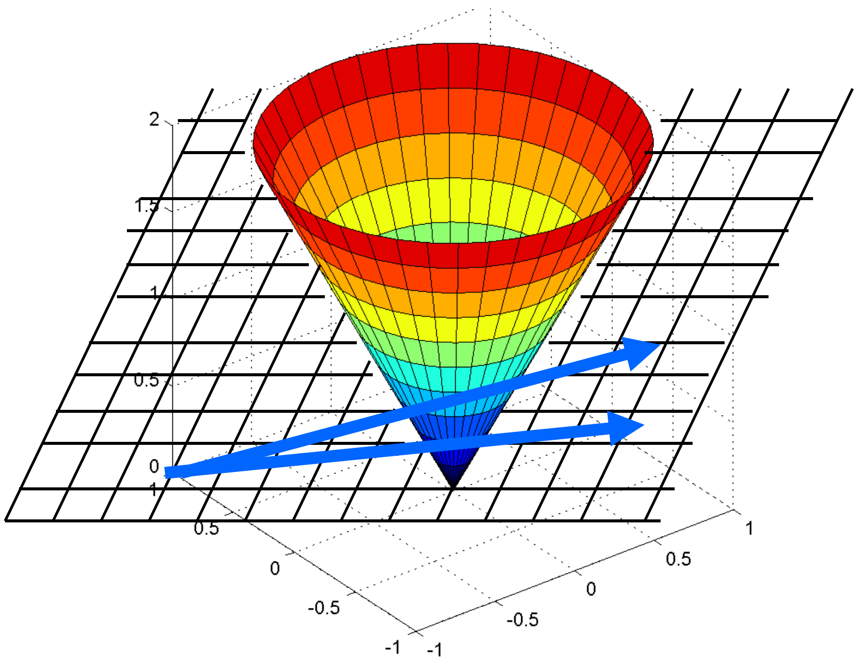
Projections

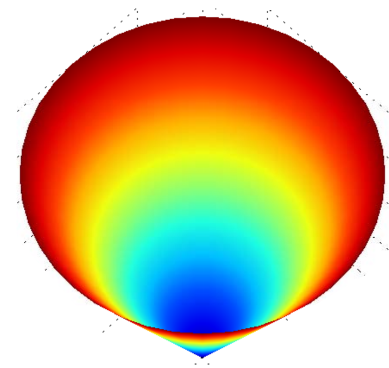




- HOW?

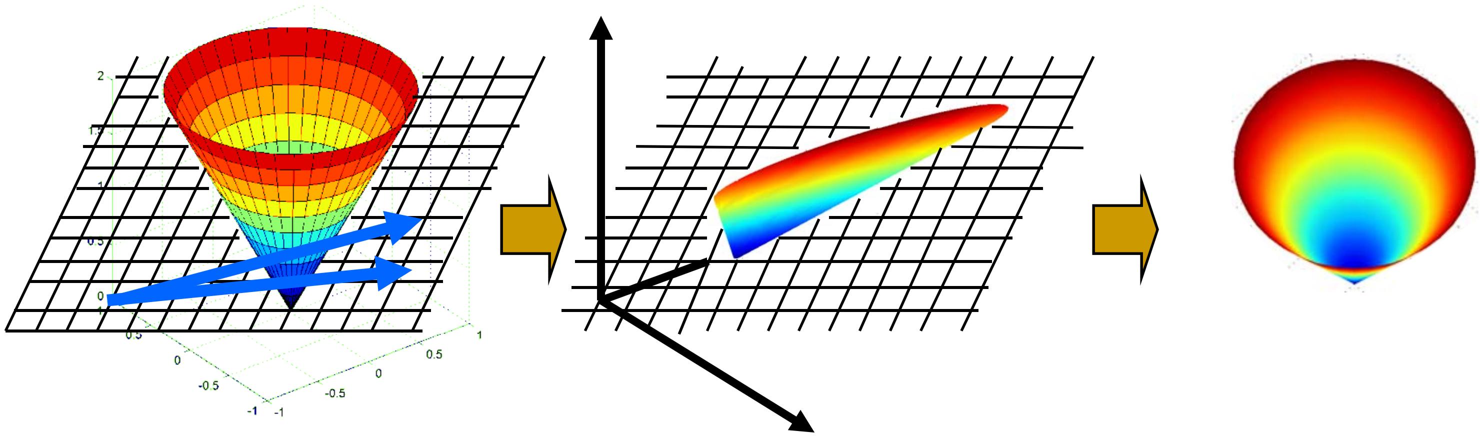
Projections





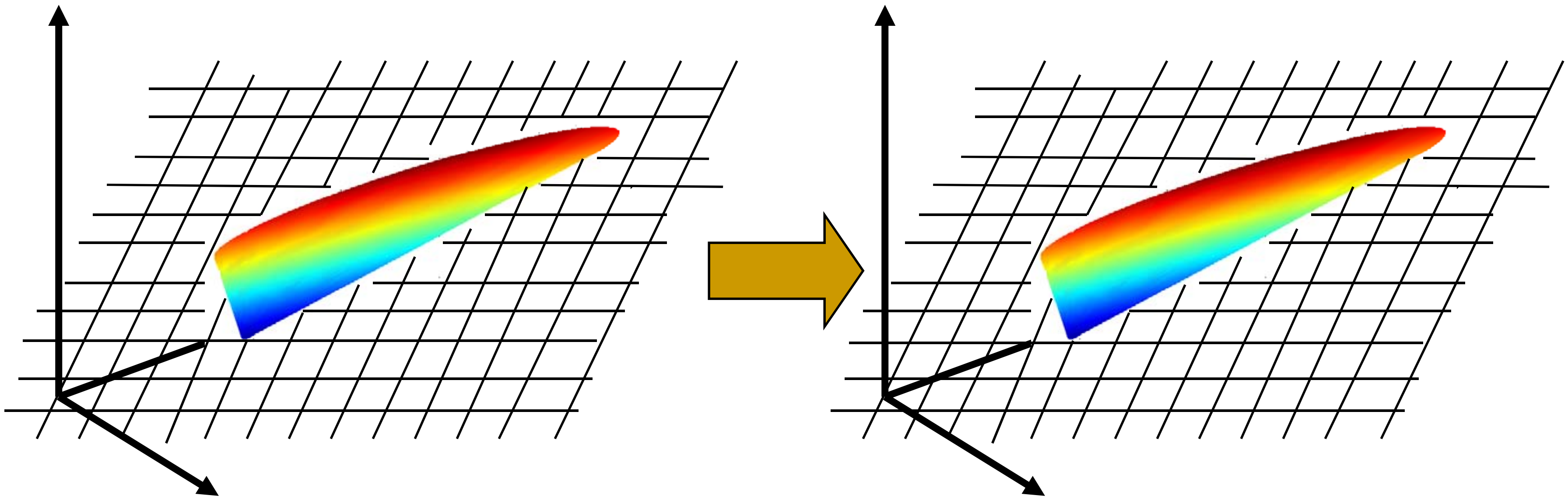
- Draw any two vectors W1 and W2 that lie on the plane
  - ***ANY two*** **so long as they have different angles**
- Compose a matrix W = [W1 W2]
- Compose the projection matrix $P = W (W^T W)^{-1} W^T$
- Multiply every point on the cone by P to get its projection
- View it ☺
  - I'm missing a step here – what is it?

Projections



- The projection actually projects it onto the plane, but you're still seeing the plane in 3D
  - The result of the projection is a 3-D vector
  - $P = W (W^TW)^{-1} W^T = 3x3$, $P*Vector = 3x1$
  - The image must be rotated till the plane is in the plane of the paper
    - The Z axis in this case will always be zero and can be ignored
    - How will you rotate it? (remember you know W1 and W2)

Projection matrix properties



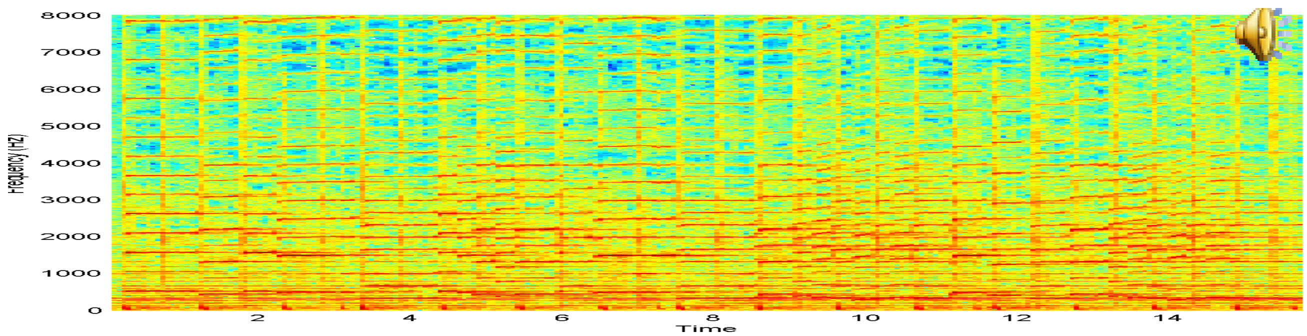
- The projection of any vector that is already on the plane is the vector itself
  - Px = x if x is on the plane
  - If the object is already on the plane, there is no further projection to be performed
- The projection of a projection is the projection
  - P (Px) = Px
  - That is because Px is already on the plane
- Projection matrices are *idempotent*
  - $P^2 = P$
    - Follows from the above

Projections: A more physical meaning

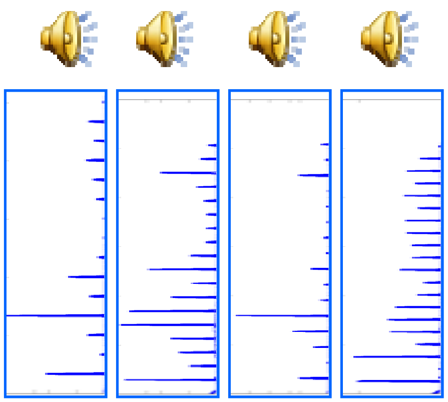
- Let W_1, W_2 .. W_k be "bases"
- We want to explain our data in terms of these "bases"
  - We often cannot do so
  - But we can explain a significant portion of it

- The portion of the data that can be expressed in terms of our vectors W_1, W_2, .. W_k, is the projection of the data on the W_1 .. W_k (hyper) plane
  - In our previous example, the "data" were all the points on a cone
  - The interpretation for volumetric data is obvious

Projection : an example with sounds



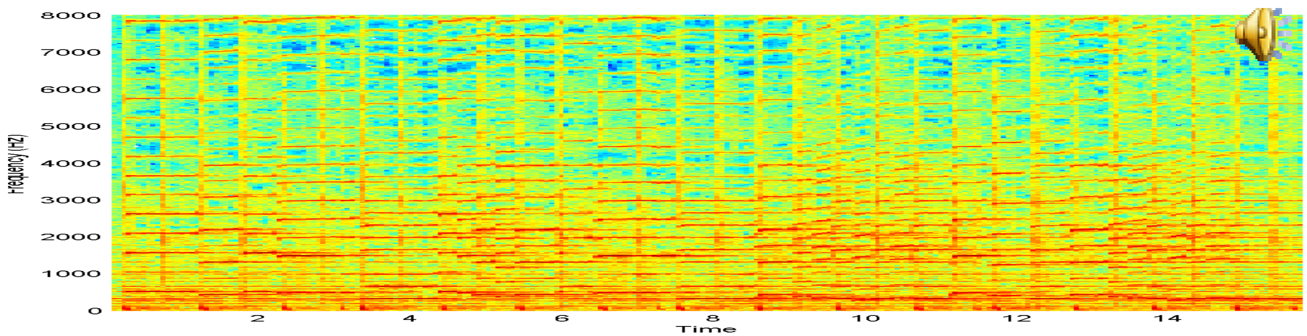
- The spectrogram (matrix) of a piece of music



- How much of the above music was composed of the above notes
  - I.e. how much can it be explained by the notes

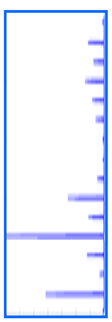
Projection: one note

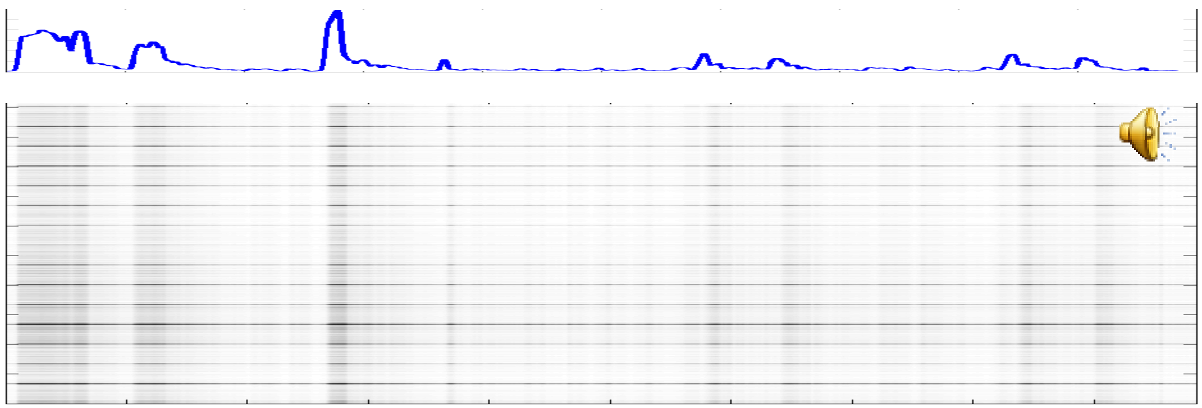
M =



- The spectrogram (matrix) of a piece of music

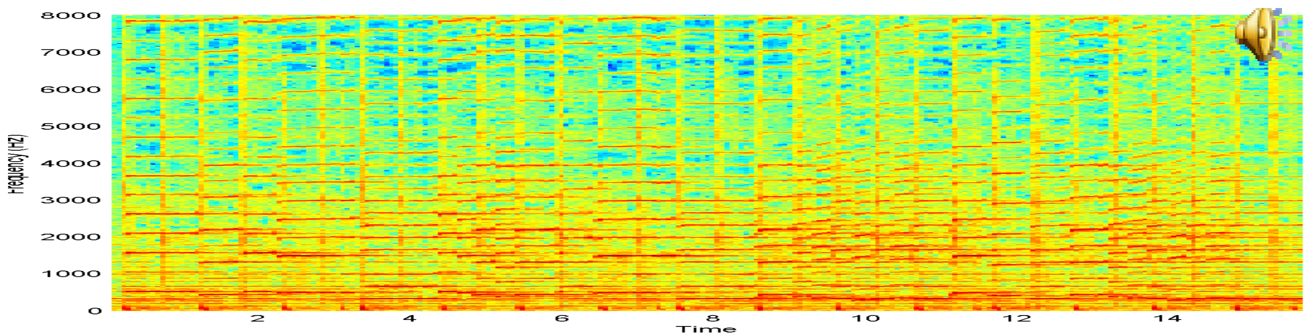
W =



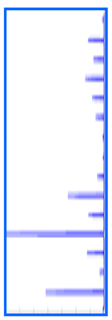


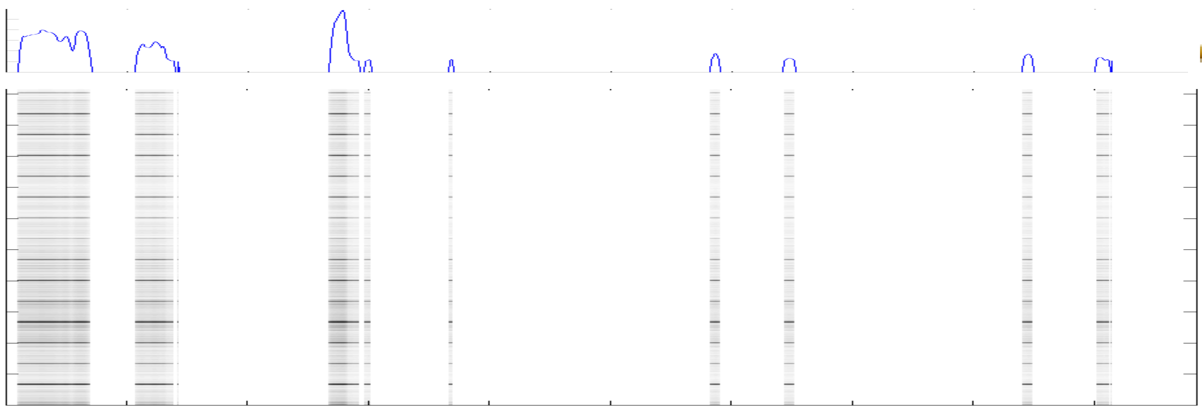
- M = spectrogram;   W = note
- P = W $(W^TW)^{-1}$ W^T
- Projected Spectrogram = P * M

Projection: one note – cleaned up

M = 

- The spectrogram (matrix) of a piece of music

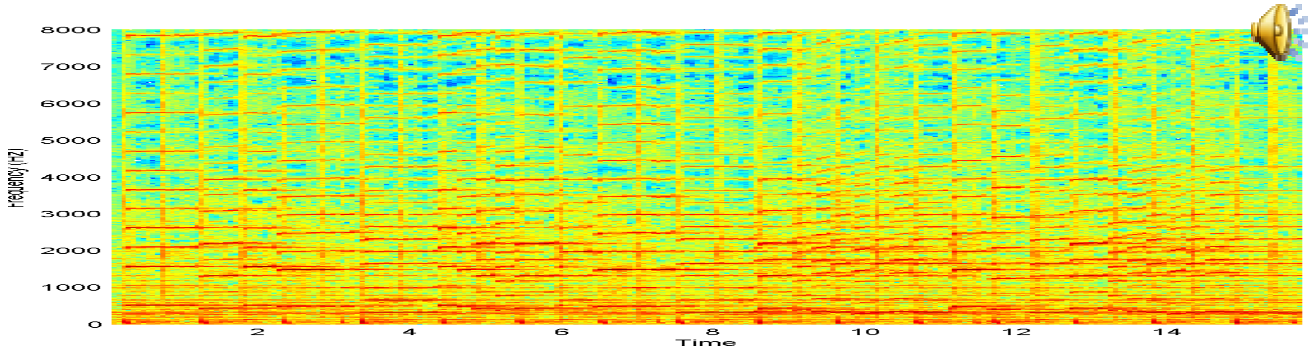
W = 



- Floored all matrix values below a threshold to zero

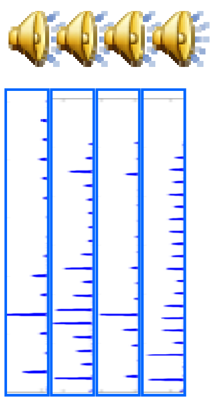
Projection: multiple notes

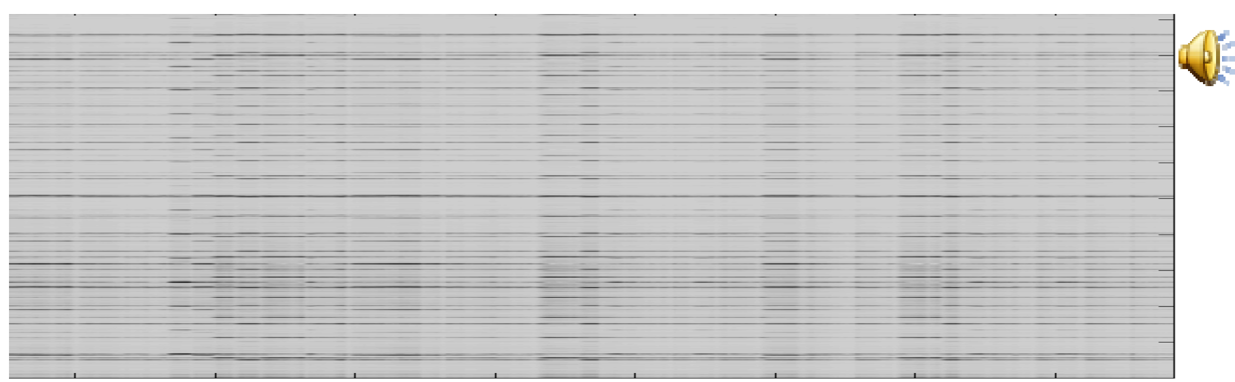
M =



- The spectrogram (matrix) of a piece of music

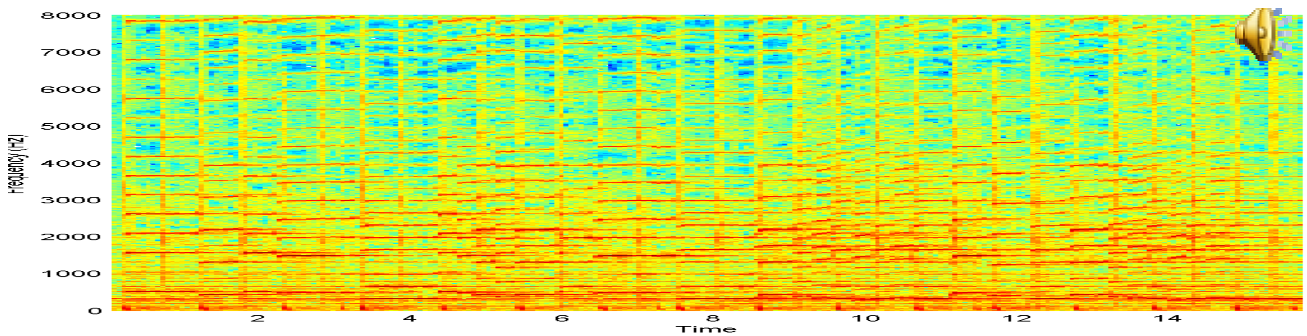
W =



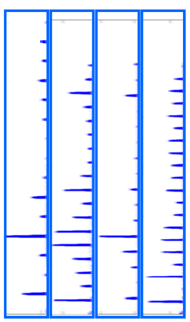


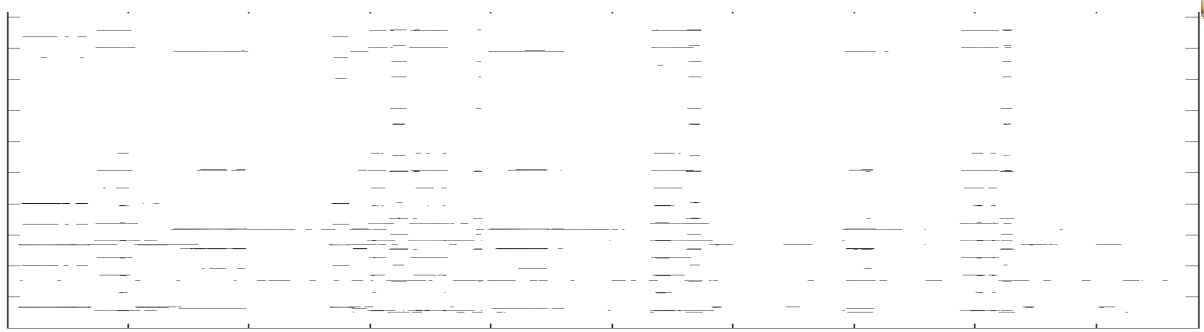
- $P = W (W^T W)^{-1} W^T$
- Projected Spectrogram = P * M

Projection: multiple notes, cleaned up

M = 

- The spectrogram (matrix) of a piece of music

W = 



- $P = W (W^T W)^{-1} W^T$
- Projected Spectrogram = P * M

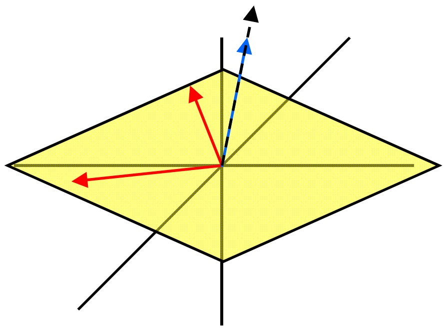
Projection and Least Squares

- Projection actually computes a *least squared error* estimate
- For each vector V in the music spectrogram matrix
  - Approximation: V_{approx} = a*note1 + b*note2 + c*note3..

$$V_{approx} = \begin{bmatrix} \text{note1} & \text{note2} & \text{note3} \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix}$$

  - Error vector E = V – V_{approx}
  - Squared error energy for V   e(V) = norm(E)^2
  - Total error = sum_over_all_V { e(V) } = Σ_V e(V)
- Projection computes V_{approx} for all vectors such that Total error is minimized
  - It does not give you "a", "b", "c".. Though
    - That needs a different operation – the inverse / pseudo inverse

Orthogonal and Orthonormal matrices



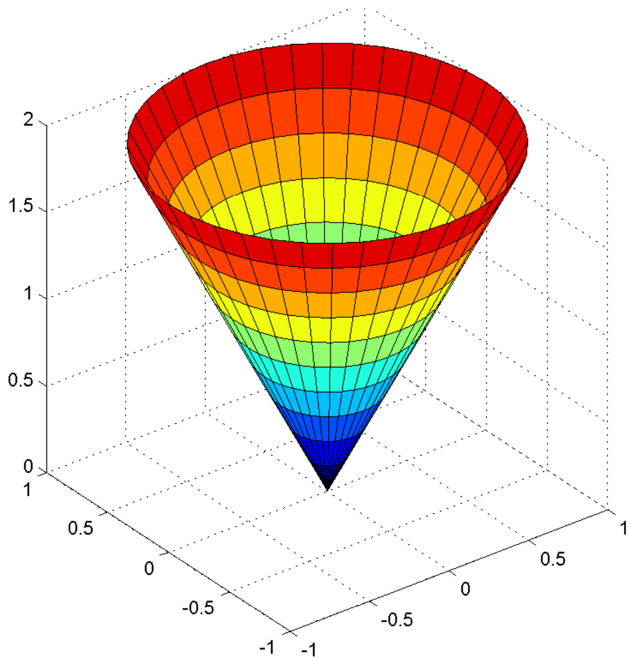
$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \begin{bmatrix} 0.707 & -0.354 & 0.612 \\ 0.707 & 0.354 & -0.612 \\ 0 & 0.866 & 0.5 \end{bmatrix}$$

- Orthogonal Matrix : AA^T = diagonal
  - Each row vector lies exactly along the normal to the plane specified by the rest of the vectors in the matrix
- Orthonormal Matrix: $AA^T = A^TA = I$
  - In additional to be orthogonal, each vector has length exactly = 1.0
  - Interesting observation: In a square matrix if the length of the row vectors is 1.0, the length of the column vectors is also 1.0

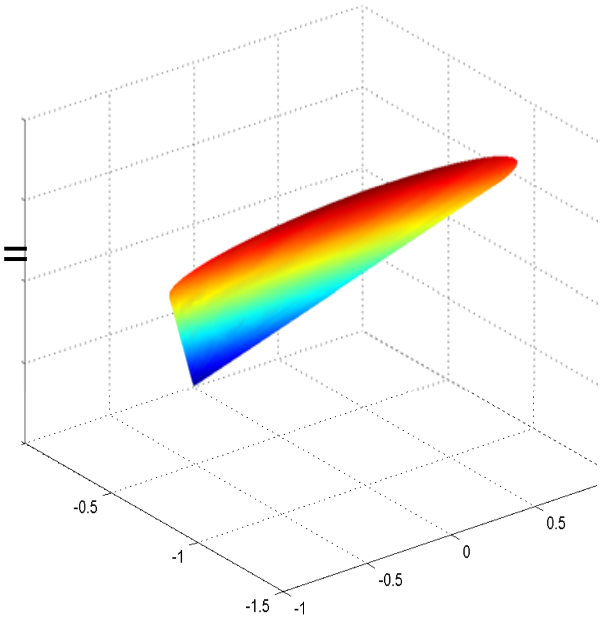
Orthogonal and Orthonormal Matrices

- Orthonormal matrices will retain the relative angles between transformed vectors
  - Essentially, they are combinations of rotations, reflections and permutations
  - Rotation matrices and permutation matrices are all orthonormal matrices
  - The vectors in an orthonormal matrix are at 90degrees to one another.
- Orthogonal matrices are like Orthonormal matrices with stretching
  - The product of a diagonal matrix and an orthonormal matrix

Matrix Rank and Rank-Deficient Matrices



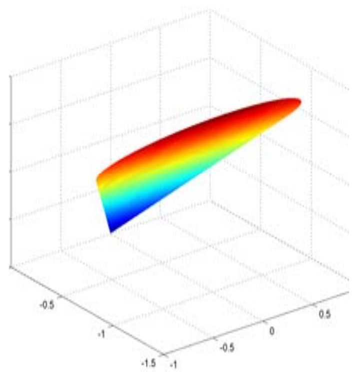
P * Cone =



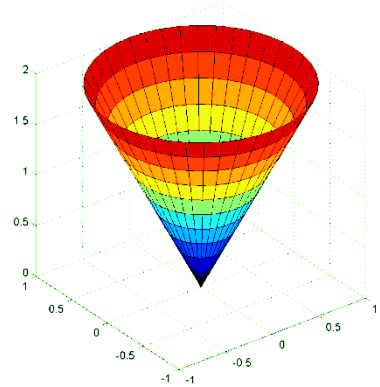
- Some matrices will eliminate one or more dimensions during transformation
  - These are *rank deficient* matrices
  - The rank of the matrix is the dimensionality of the trasnsformed version of a full-dimensional object

Matrix Rank and Rank-Deficient Matrices

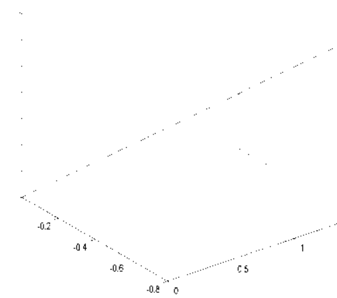
```
P =

    1.0000         0         0
         0    0.2500   -0.4330
         0   -0.4330    0.7500
```



Rank = 2



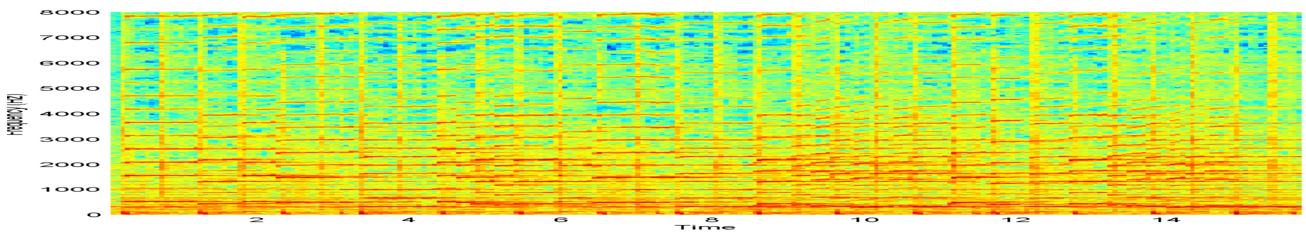
```
P2 =

    0.5000   -0.2500    0.4330
   -0.2500    0.1250   -0.2165
    0.4330   -0.2165    0.3750
```

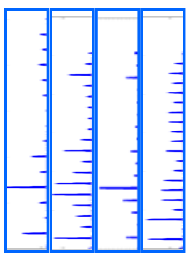


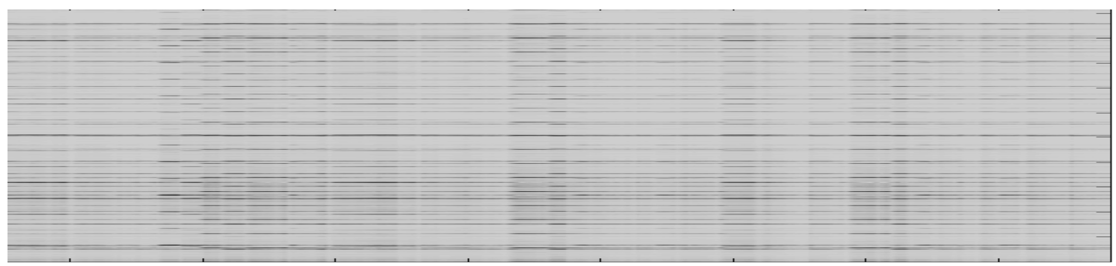
Rank = 1

- Some matrices will eliminate one or more dimensions during transformation
  - These are *rank deficient* matrices
  - The rank of the matrix is the dimensionality of the transformed version of a full-dimensional object

Projections are often examples of rank-deficient transforms

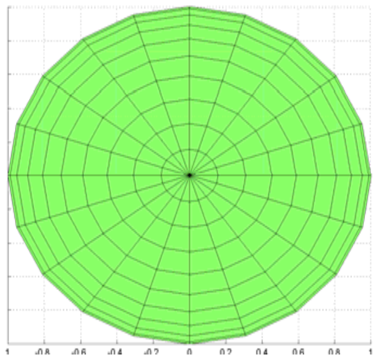
M = 

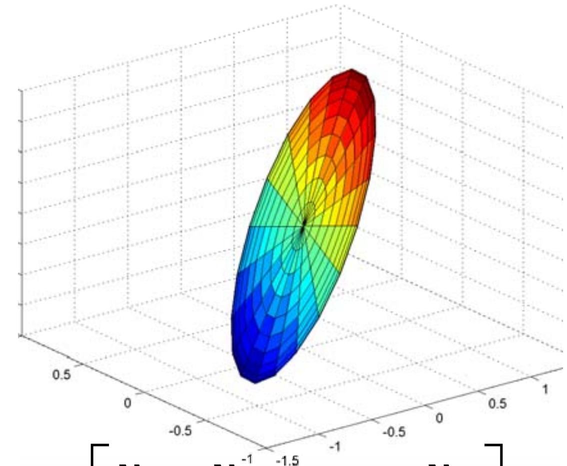
W = 



- $P = W (W^T W)^{-1} W^T$; Projected Spectrogram = P * M
- The original spectrogram can never be recovered
  - P is rank deficient
- P explains all vectors in the new spectrogram as a mixture of only the 4 vectors in W
  - There are only 4 ***independent*** bases
  - Rank of P is 4

Non-square Matrices





$$\begin{bmatrix} x_1 & x_2 & . & . & x_N \\ y_1 & y_2 & . & . & y_N \end{bmatrix}$$

X = 2D data

$$\begin{bmatrix} .8 & .9 \\ .1 & .9 \\ .6 & 0 \end{bmatrix}$$

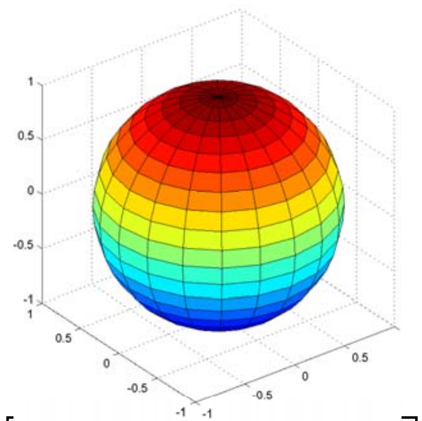
P = transform

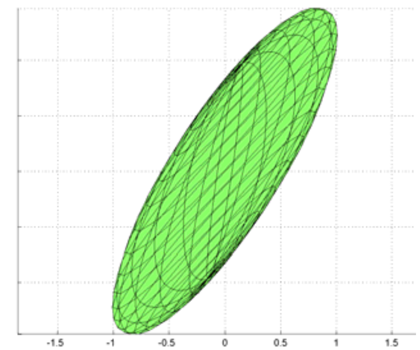
$$\begin{bmatrix} x_1 & x_2 & . & . & x_N \\ y_1 & y_2 & . & . & y_N \\ z_1 & z_2 & . & . & z_N \end{bmatrix}$$

PX = 3D, rank 2

- Non-square matrices add or subtract axes
  - More rows than columns → add axes
    - But does not increase the dimensionality of the data

Non-square Matrices





$$\begin{bmatrix} x_1 & x_2 & . & . & x_N \\ y_1 & y_2 & . & . & y_N \\ z_1 & z_2 & . & . & z_N \end{bmatrix}$$

X = 3D data, rank 3

$$\begin{bmatrix} .3 & 1 & .2 \\ .5 & 1 & 1 \end{bmatrix}$$

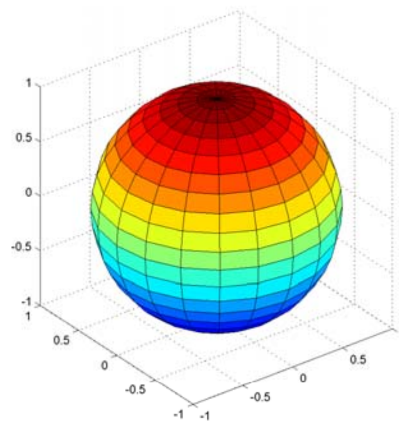
P = transform

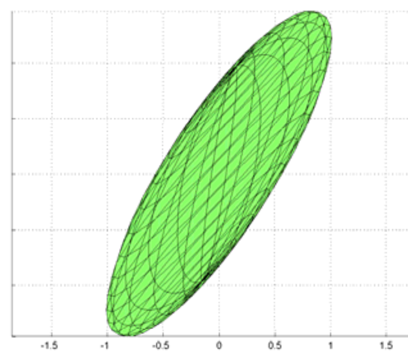
$$\begin{bmatrix} x_1 & x_2 & . & . & x_N \\ y_1 & y_2 & . & . & y_N \end{bmatrix}$$

PX = 2D, rank 2

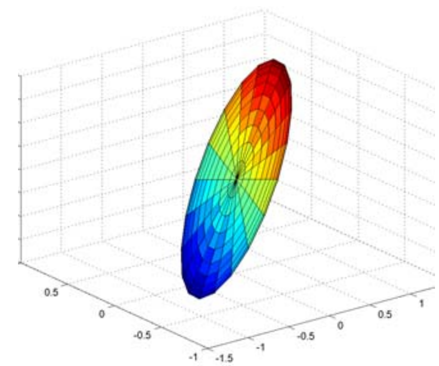
- Non-square matrices add or subtract axes
  - Fewer rows than columns → reduce axes
    - May reduce dimensionality of the data

The Rank of a Matrix





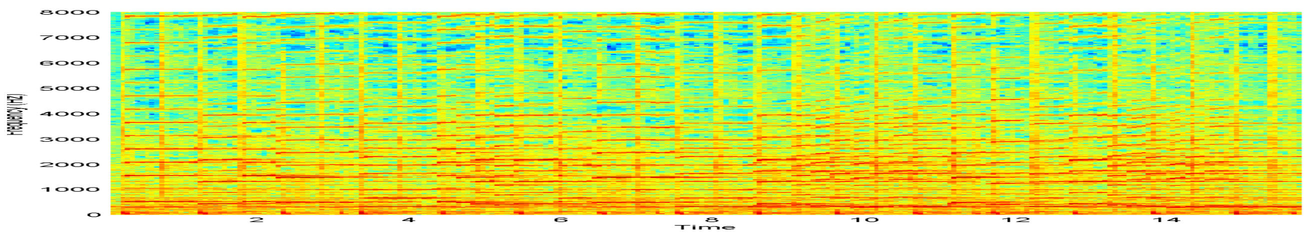
$$\begin{bmatrix} .3 & 1 & .2 \\ .5 & 1 & 1 \end{bmatrix}$$

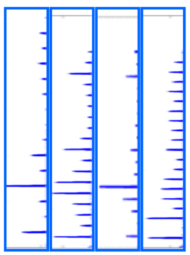


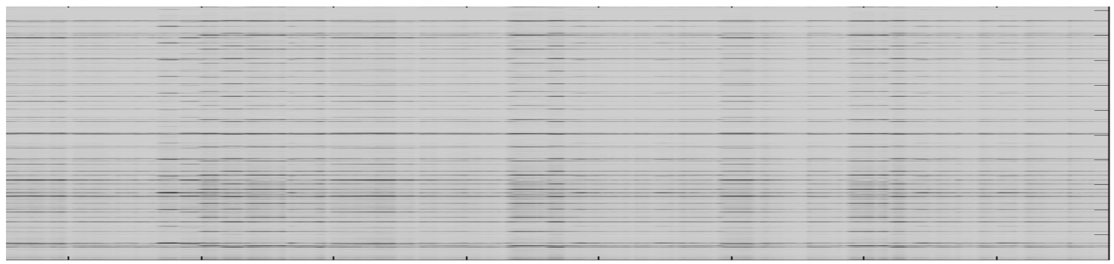
$$\begin{bmatrix} .8 & .9 \\ .1 & .9 \\ .6 & 0 \end{bmatrix}$$

- The matrix rank is the dimensionality of the transformation of a full-dimensioned object in the original space
- The matrix can never *increase* dimensions
  - Cannot convert a circle to a sphere or a line to a circle
- The rank of a matrix can never be greater than the lower of its two dimensions

The Rank of Matrix

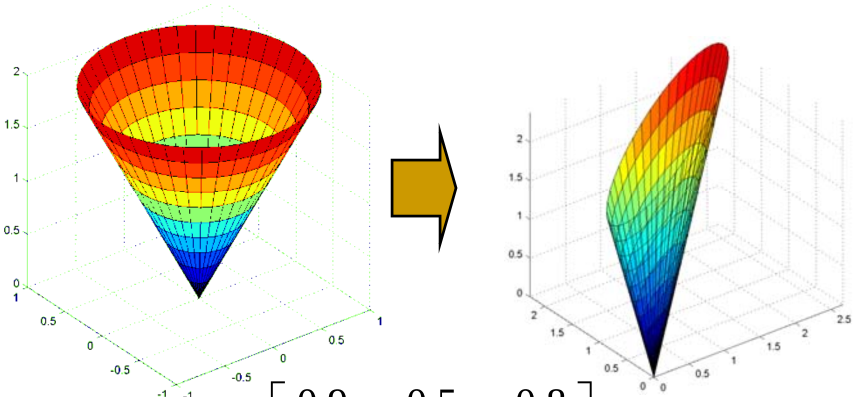
M = 



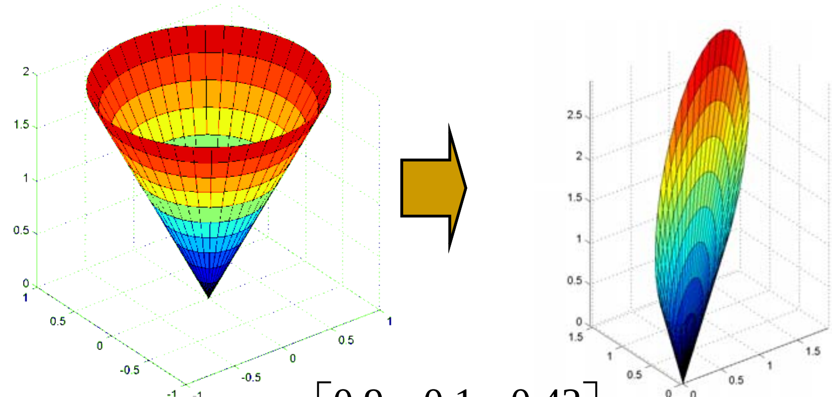


- Projected Spectrogram = P * M
  - Every vector in it is a combination of only 4 bases
- The rank of the matrix is the *smallest* no. of bases required to describe the output
  - E.g. if note no. 4 in P could be expressed as a combination of notes 1,2 and 3, it provides no additional information
  - Eliminating note no. 4 would give us the same projection
  - The rank of P would be 3!

Matrix rank is unchanged by transposition



$$\begin{bmatrix} 0.9 & 0.5 & 0.8 \\ 0.1 & 0.4 & 0.9 \\ 0.42 & 0.44 & 0.86 \end{bmatrix}$$

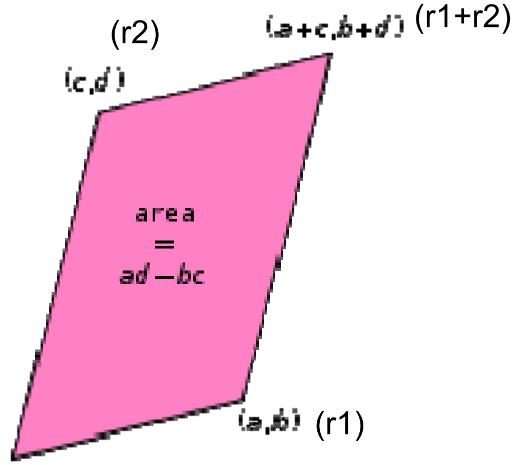


$$\begin{bmatrix} 0.9 & 0.1 & 0.42 \\ 0.5 & 0.4 & 0.44 \\ 0.8 & 0.9 & 0.86 \end{bmatrix}$$

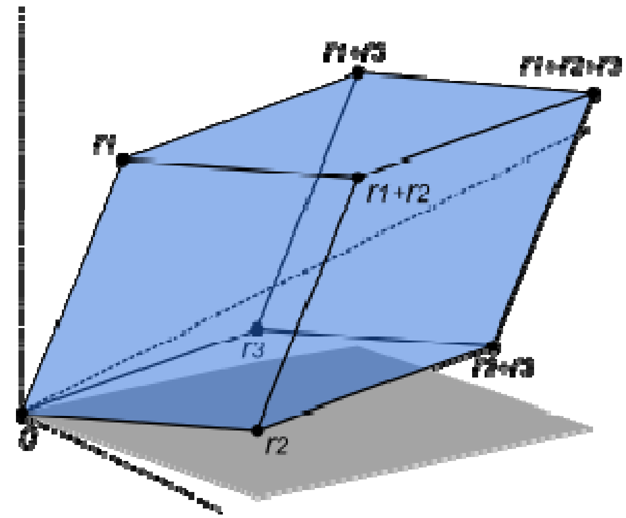
- If an N-D object is compressed to a K-D object by a matrix, it will also be compressed to a K-D object by the transpose of the matrix

Matrix Determinant

$$A = \begin{array}{c} \begin{bmatrix} a & b \\ c & d \end{bmatrix} \end{array} \begin{array}{l} \text{(r1)} \\ \text{(r2)} \end{array}$$



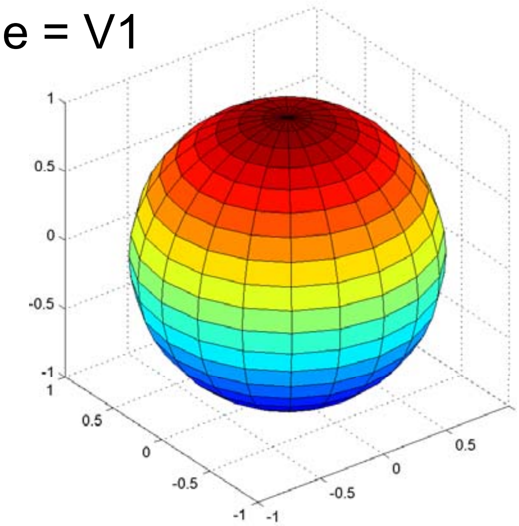
$$A = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix}$$



- The determinant is the “volume” of a matrix
- Actually the volume of a parallelepiped formed from its row vectors
  - Also the volume of the parallelepiped formed from its column vectors
- Standard formula for determinant: in text book

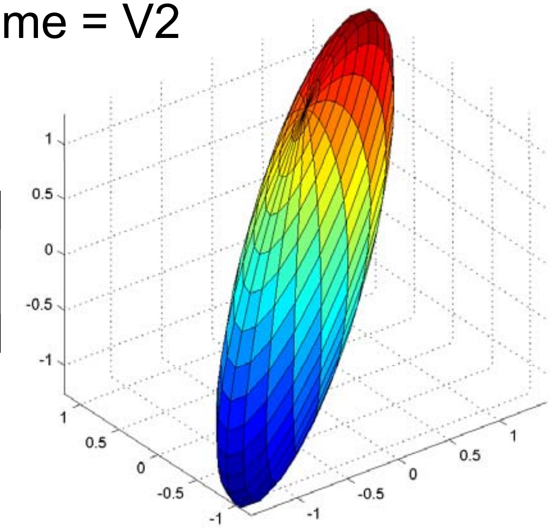
Matrix Determinant: Another Perspective

Volume = V1



$$\begin{bmatrix} 0.8 & 0 & 0.7 \\ 1.0 & 0.8 & 0.8 \\ 0.7 & 0.9 & 0.7 \end{bmatrix}$$

Volume = V2



- The determinant is the ratio of N-volumes
  - If V_1 is the volume of an N-dimensional object “O” in N-dimensional space
    - O is the complete set of points or vertices that specify the object
  - If V_2 is the volume of the N-dimensional object specified by A*O, where A is a matrix that transforms the space
  - $|A| = V_2 / V_1$

Matrix Determinants

- Matrix determinants are *only defined for square matrices*
  - They characterize volumes in linearly transformed space of the same dimensionality as the vectors
- Rank deficient matrices have determinant 0
  - Since they compress full-volumed N-D objects into zero-volume N-D objects
    - E.g. a 3-D sphere into a 2-D ellipse: The ellipse has 0 volume (although it does have area)
- Conversely, all matrices of determinant 0 are rank deficient
  - Since they compress full-volumed N-D objects into zero-volume objects

Multiplication properties

- Properties of vector/matrix products
  - Associative

$$\mathbf{A} \cdot (\mathbf{B} \cdot \mathbf{C}) = (\mathbf{A} \cdot \mathbf{B}) \cdot \mathbf{C}$$

  - Distributive

$$\mathbf{A} \cdot (\mathbf{B} + \mathbf{C}) = \mathbf{A} \cdot \mathbf{B} + \mathbf{A} \cdot \mathbf{C}$$

  - NOT commutative!!!

$$\mathbf{A} \cdot \mathbf{B} \neq \mathbf{B} \cdot \mathbf{A}$$

    - *left multiplications ≠ right multiplications*
  - Transposition

$$(\mathbf{A} \cdot \mathbf{B})^T = \mathbf{B}^T \cdot \mathbf{A}^T$$

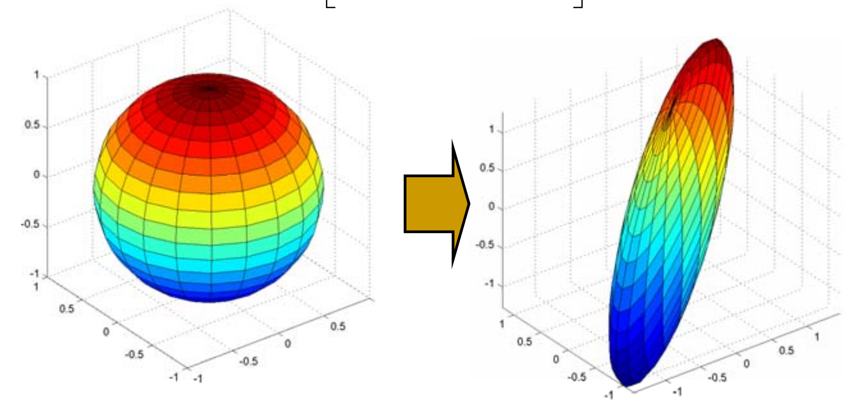
Determinant properties

- Associative for square matrices $$|\mathbf{A} \cdot \mathbf{B} \cdot \mathbf{C}| = |\mathbf{A}| \cdot |\mathbf{B}| \cdot |\mathbf{C}|$$
  - Scaling volume sequentially by several matrices is equal to scaling once by the product of the matrices
- Volume of sum != sum of Volumes $$|(\mathbf{B} + \mathbf{C})| \neq |\mathbf{B}| + |\mathbf{C}|$$
  - The volume of the parallelepiped formed by row vectors of the sum of two matrices is not the sum of the volumes of the parallelepipeds formed by the original matrices
- Commutative for square matrices!!!
$$|\mathbf{A} \cdot \mathbf{B}| = |\mathbf{B} \cdot \mathbf{A}| = |\mathbf{A}| \cdot |\mathbf{B}|$$
  - The order in which you scale the volume of an object is irrelevant

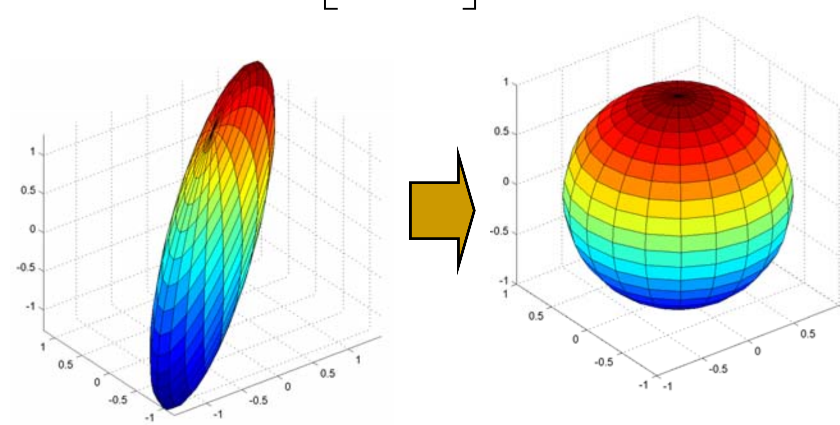
Matrix Inversion

- A matrix transforms an N-D object to a different N-D object
- What transforms the new object back to the original?
  - The *inverse transformation*
- The inverse transformation is called the matrix inverse

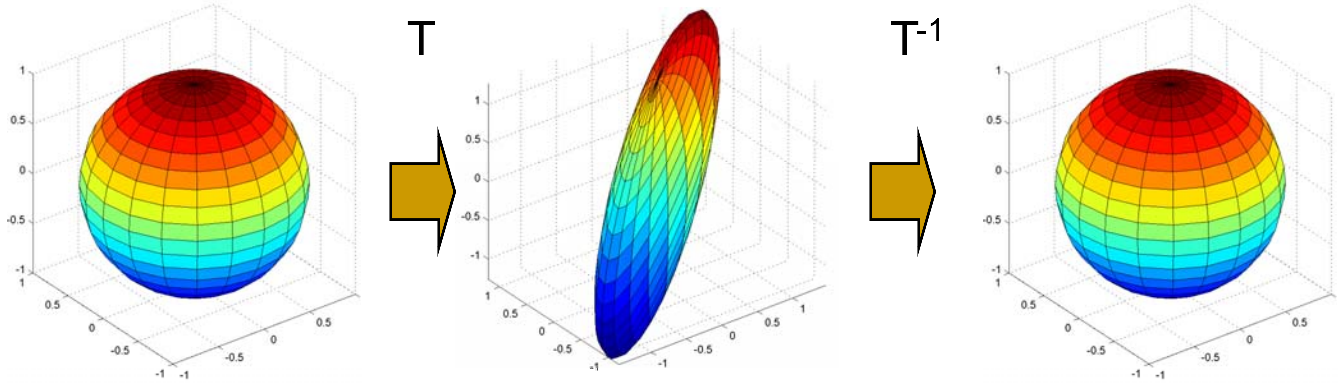
$$T = \begin{bmatrix} 0.8 & 0 & 0.7 \\ 1.0 & 0.8 & 0.8 \\ 0.7 & 0.9 & 0.7 \end{bmatrix}$$



$$Q = \begin{bmatrix} ? & ? & ? \\ ? & ? & ? \\ ? & ? & ? \end{bmatrix} = T^{-1}$$



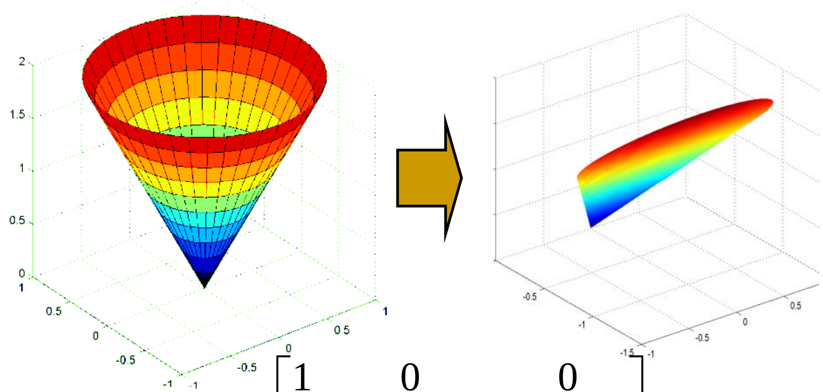
Matrix Inversion



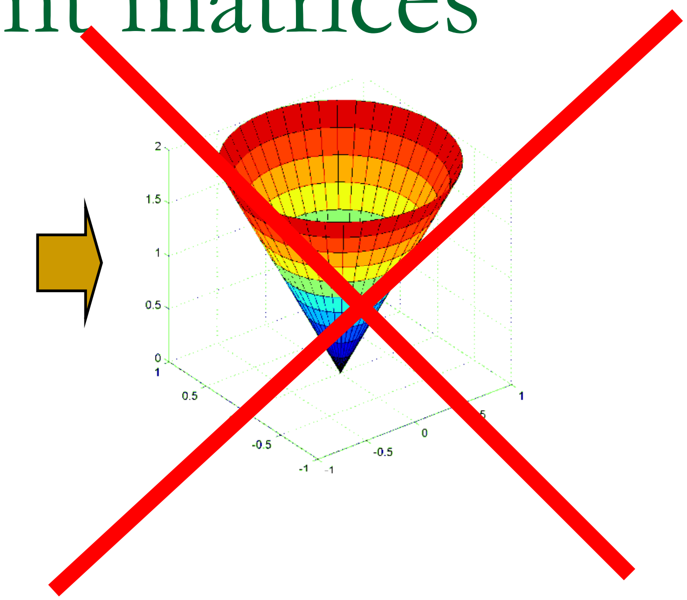
$T^{-1}T = I$

- The product of a matrix and its inverse is the identity matrix
  - Transforming an object, and then inverse transforming it gives us back the original object

Inverting rank-deficient matrices



$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & .25 & -0.433 \\ 0 & -0.433 & 0.75 \end{bmatrix}$$



- Rank deficient matrices "flatten" objects
  - In the process, multiple points in the original object get mapped to the same point in the transformed object
- It is not possible to go "back" from the flattened object to the original object
  - Because of the many-to-one forward mapping
- Rank deficient matrices have no inverse

Revisiting Projections and Least Squares

- Projection computes a *least squared error* estimate
- For each vector V in the music spectrogram matrix
  - Approximation: V_{approx} = a*note1 + b*note2 + c*note3..

$$W = \begin{bmatrix} \text{note1} & \text{note2} & \text{note3} \end{bmatrix} \qquad V_{approx} = W \begin{bmatrix} a \\ b \\ c \end{bmatrix}$$

  - Error vector E = $V - V_{approx}$
  - Squared error energy for V $e(V) = \text{norm}(E)^2$
  - Total error = Total error + e(V)
- Projection computes V_{approx} for all vectors such that Total error is minimized
- **But WHAT ARE "a" "b" and "c"?**

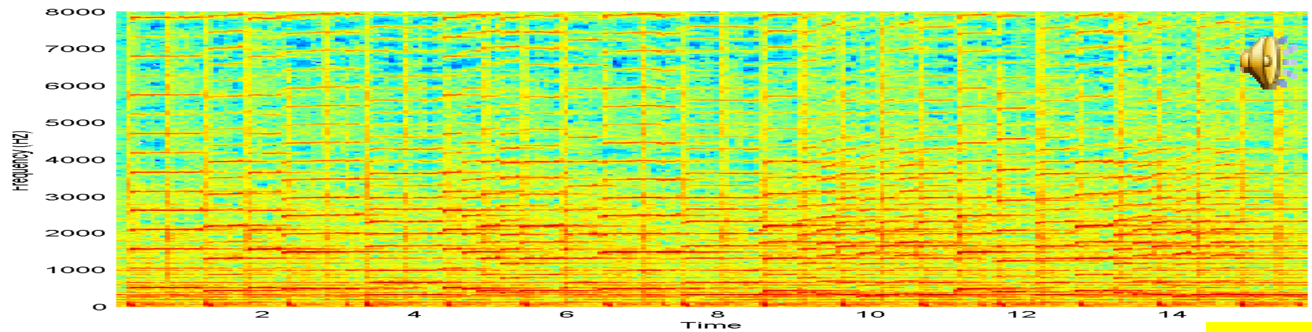
The Pseudo Inverse (PINV)

$$V_{approx} = W \begin{bmatrix} a \\ b \\ c \end{bmatrix} \Rightarrow V \approx W \begin{bmatrix} a \\ b \\ c \end{bmatrix} \Rightarrow \begin{bmatrix} a \\ b \\ c \end{bmatrix} = PINV(W) * V$$

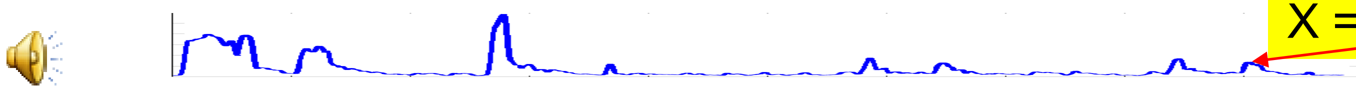
- We are approximating spectral vectors V as the transformation of the vector $[a\ b\ c]^T$
  - Note – we're viewing the collection of bases in W as a transformation
- The solution is obtained using the *pseudo inverse*
  - This give us a *LEAST SQUARES* solution
    - If W were square and invertible Pinv(W) = W^{-1}, and $V=V_{approx}$

Explaining music with one note

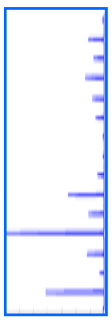
M =

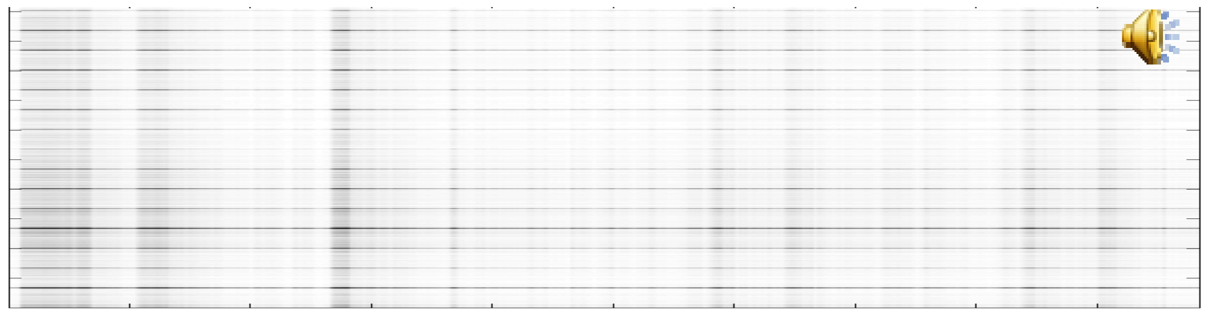


X =PINV(W)*M



W =





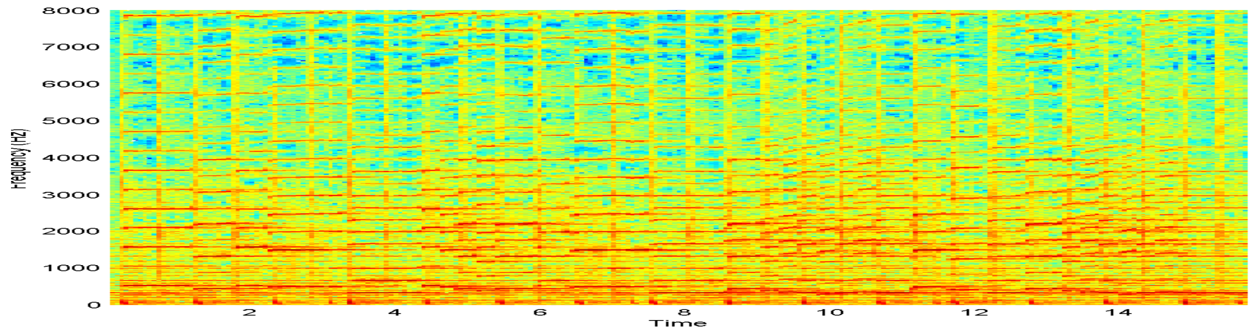
- Recap: P = W $(W^TW)^{-1}$ W^T, Projected Spectrogram = P*M

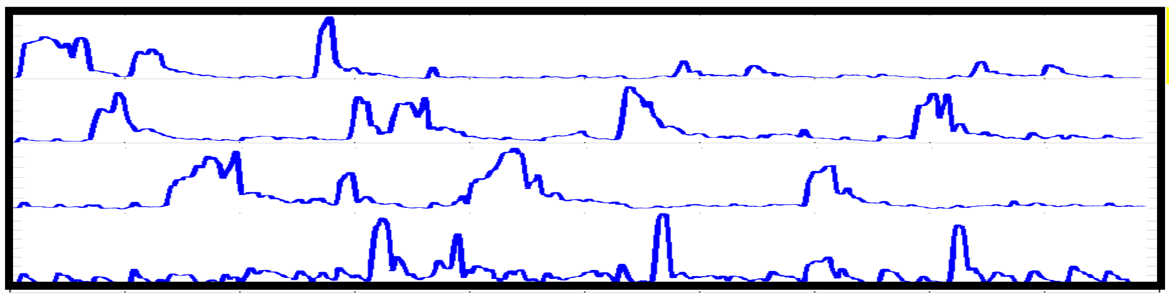
- **Approximation: M ≈ W*X**
- The amount of W in each vector = X = PINV(W)*M
- W*Pinv(W)*M = Projected Spectrogram = P*M
  - W*Pinv(W) = Projection matrix = W $(W^TW)^{-1}$ W.

PINV(W) = $(W^TW)^{-1}W^T$

Explanation with multiple notes

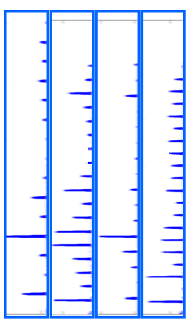
M =

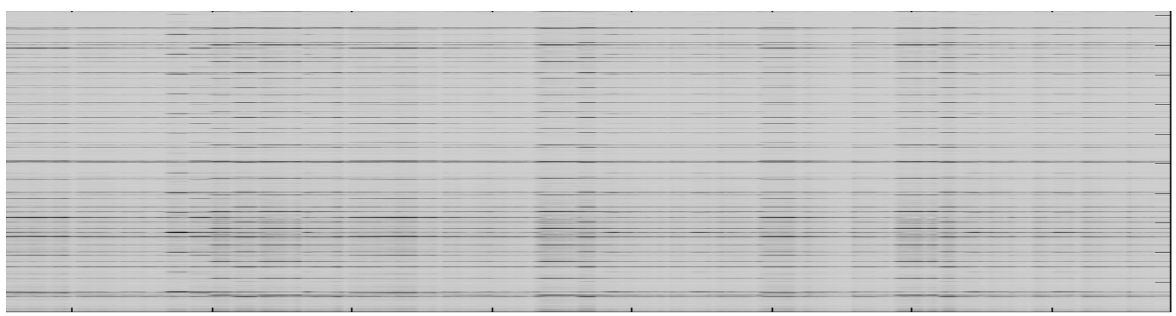




X=PINV(W)*M

W =

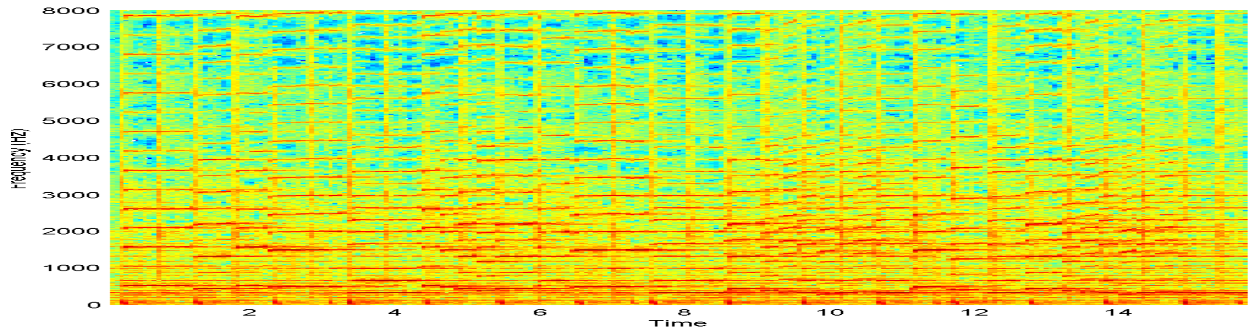




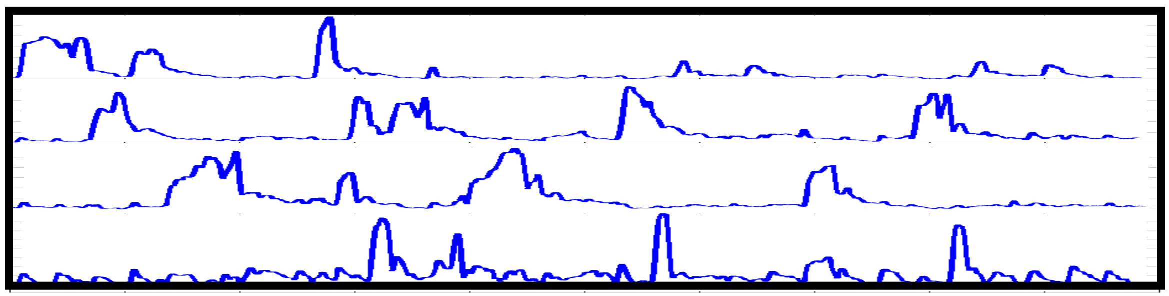
- X = Pinv(W) * M; Projected matrix = W*X = W*Pinv(W)*M

How about the other way?

M =



V =



W = ?

U = ?

- WV \approx M W = M * Pinv(V) U = WV

Pseudo-inverse (PINV)

- Pinv() applies to non-square matrices
- Pinv (Pinv (A))) = A
- A*Pinv(A)= projection matrix!
  - Projection onto the columns of A

- If A = K x N matrix and K > N, A projects N-D vectors into a higher-dimensional K-D space
- Pinv(A)*A = I in this case

Matrix inversion (division)

- The inverse of matrix multiplication
  - Not element-wise division!!
- Provides a way to “undo” a linear transformation
  - Inverse of the unit matrix is itself
  - Inverse of a diagonal is diagonal
  - Inverse of a rotation is a (counter)rotation (its transpose!)
  - Inverse of a rank deficient matrix does not exist!
    - But pseudoinverse exists
- Pay attention to multiplication side!

$$\mathbf{A}\cdot\mathbf{B}=\mathbf{C},\ \ \mathbf{A}=\mathbf{C}\cdot\mathbf{B}^{-1},\ \ \mathbf{B}=\mathbf{A}^{-1}\cdot\mathbf{C}$$

- Matrix inverses defined for square matrices only
  - If matrix not square use a matrix pseudoinverse:

$$\mathbf{A}\cdot\mathbf{B}=\mathbf{C},\ \ \mathbf{A}=\mathbf{C}\cdot\mathbf{B}^{+},\ \ \mathbf{B}=\mathbf{A}^{+}\cdot\mathbf{C}$$

- MATLAB syntax: `inv(a), pinv(a)`

What is the Matrix ?

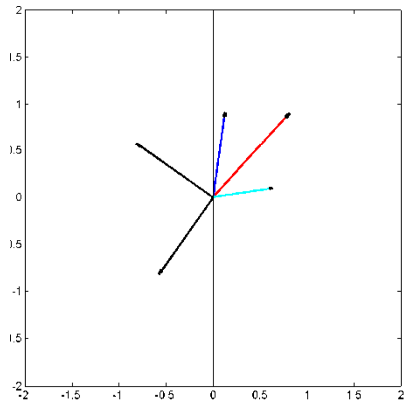
- Duality in terms of the matrix identity
  - Can be a container of data
    - An image, a set of vectors, a table, etc …
  - Can be a linear transformation
    - A process by which to transform data in another matrix
- We'll usually start with the first definition and then apply the second one on it
  - Very frequent operation
  - Room reverberations, mirror reflections, etc …
- Most of signal processing and machine learning are matrix operations!

Eigenanalysis

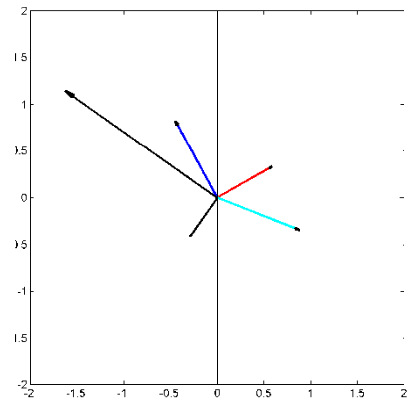
- If something can go through a process mostly unscathed in character it is an *eigen*-something
  - Sound example:
- A vector that can undergo a matrix multiplication and keep pointing the same way is an *eigenvector*
  - Its length can change though
- How much its length changes is expressed by its corresponding *eigenvalue*
  - Each eigenvector of a matrix has its eigenvalue
- Finding these "eigenthings" is called eigenanalysis

EigenVectors and EigenValues

Black vectors are eigen vectors



$$M = \begin{bmatrix} 1.5 & -0.7 \\ -0.7 & 1.0 \end{bmatrix}$$

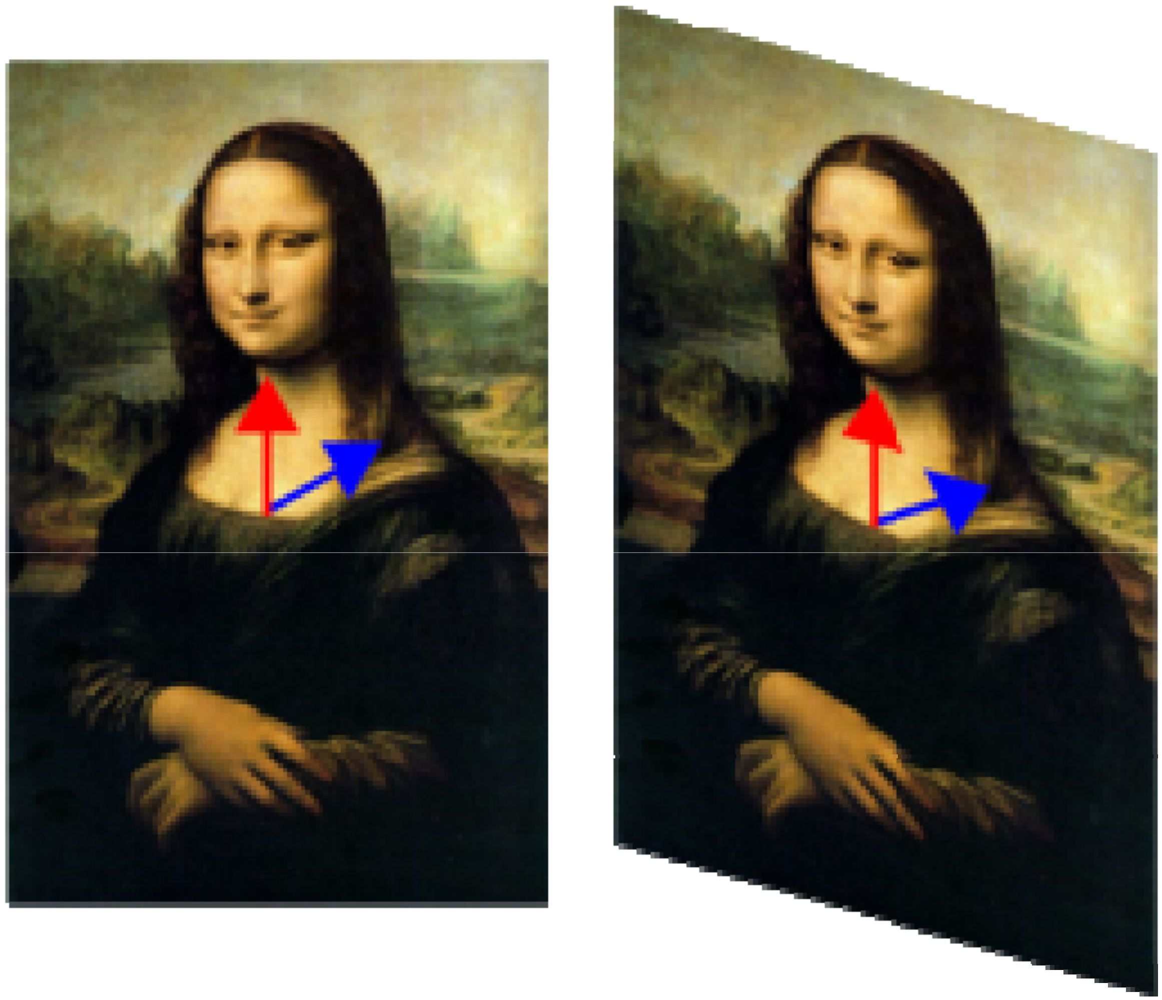


- Vectors that do not change angle upon transformation
  - They may change length

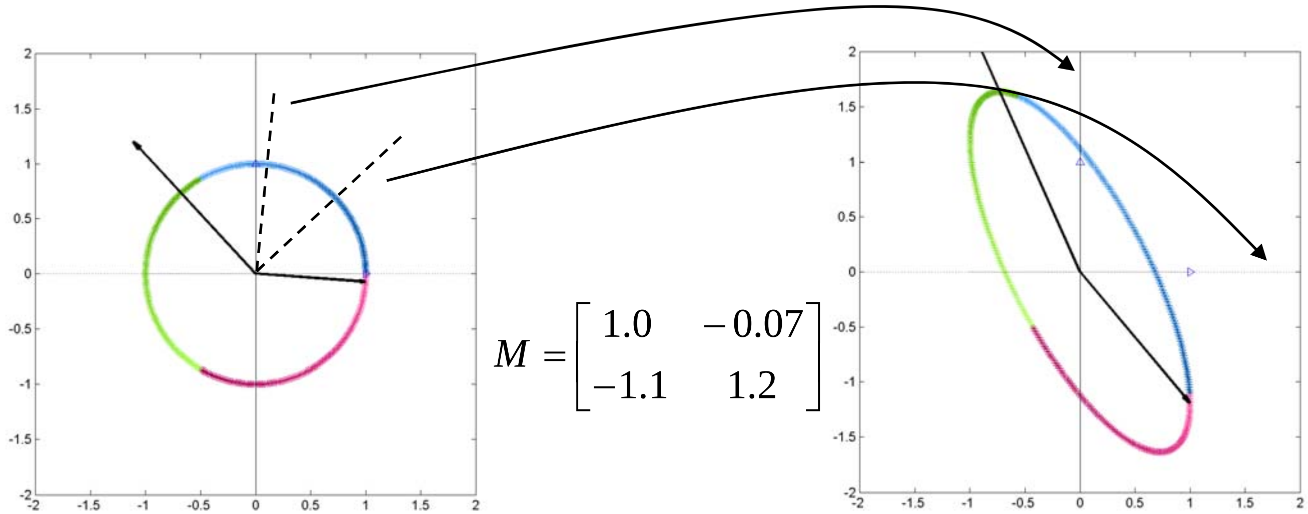
$$MV = \lambda V$$

  - V = eigen vector
  - λ = eigen value
  - `Matlab: [V, L] = eig(M)`
    - L is a diagonal matrix whose entries are the eigen values
    - V is a maxtrix whose columns are the eigen vectors

Eigen vector example



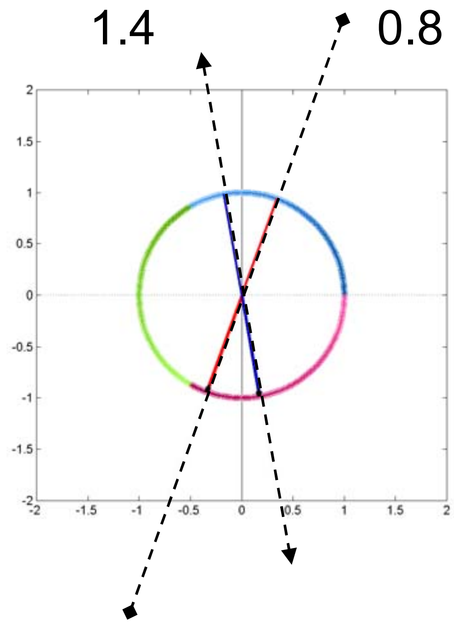
Matrix multiplication revisited



$$M = \begin{bmatrix} 1.0 & -0.07 \\ -1.1 & 1.2 \end{bmatrix}$$

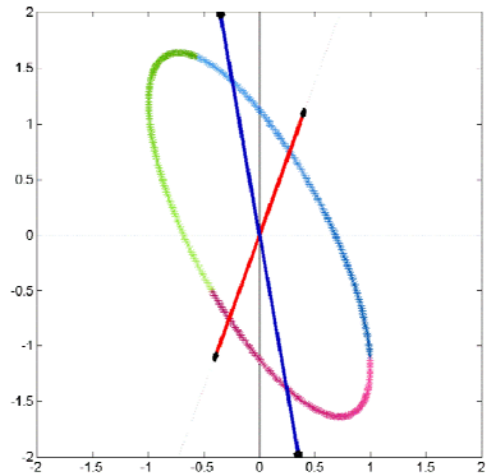
- Matrix transformation "transforms" the space
  - Warps the paper so that the normals to the two vectors now lie along the axes

A stretching operation



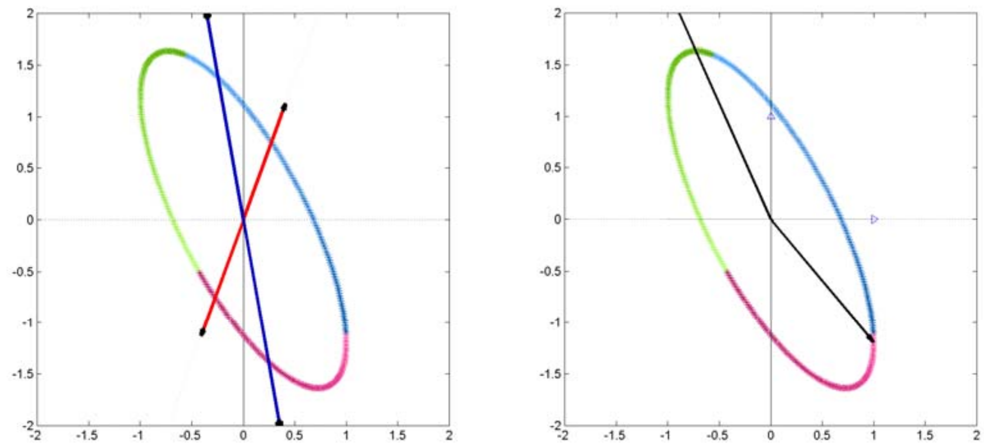
- Draw two lines
- Stretch / shrink the paper along these lines by factors λ_1 and λ_2
  - The factors could be negative – implies flipping the paper
- The result is a transformation of the space

A stretching operation



- Draw two lines
- Stretch / shrink the paper along these lines by factors λ_1 and λ_2
  - The factors could be negative – implies flipping the paper
- The result is a transformation of the space

Physical interpretation of eigen vector



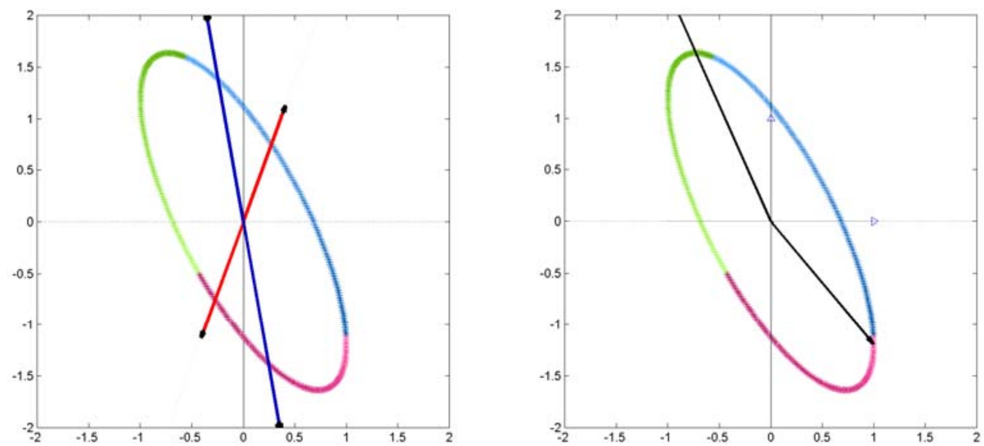
- The result of the stretching is exactly the same as transformation by a matrix
- The axes of stretching/shrinking are the eigenvectors
  - The degree of stretching/shrinking are the corresponding eigenvalues
- The EigenVectors and EigenValues convey all the information about the matrix

Physical interpretation of eigen vector

$$V = \begin{bmatrix} V_1 & V_2 \end{bmatrix}$$

$$L = \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix}$$

$$M = VLV^{-1}$$



- The result of the stretching is exactly the same as transformation by a matrix
- The axes of stretching/shrinking are the eigenvectors
  - The degree of stretching/shrinking are the corresponding eigenvalues
- The EigenVectors and EigenValues convey all the information about the matrix

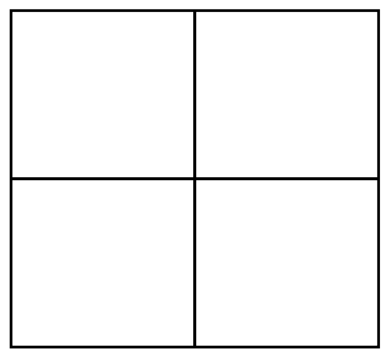
Eigen Analysis

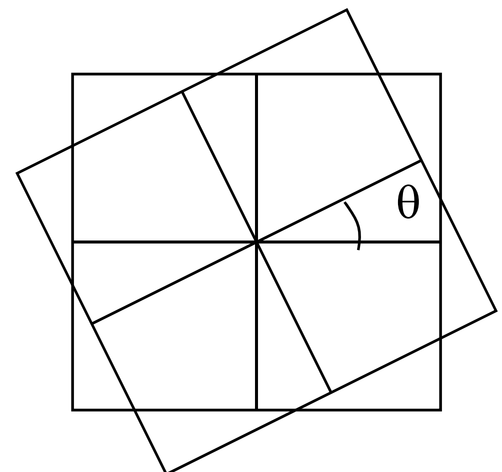
- Not all square matrices have nice eigen values and vectors
  - E.g. consider a rotation matrix

$$\mathbf{R}_\theta = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

$$X = \begin{bmatrix} x \\ y \end{bmatrix}$$

$$X_{new} = \begin{bmatrix} x' \\ y' \end{bmatrix}$$

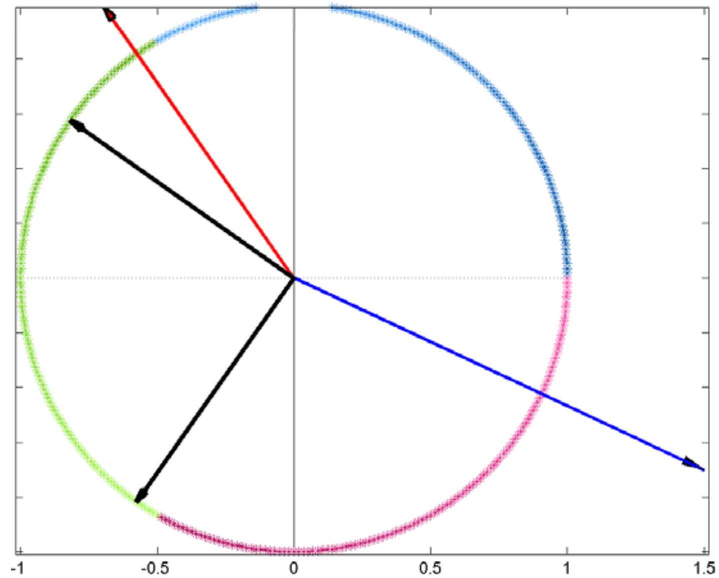




  - This rotates every vector in the plane
    - No vector that remains unchanged
- In these cases the Eigen vectors and values are complex
- Some matrices are special however..

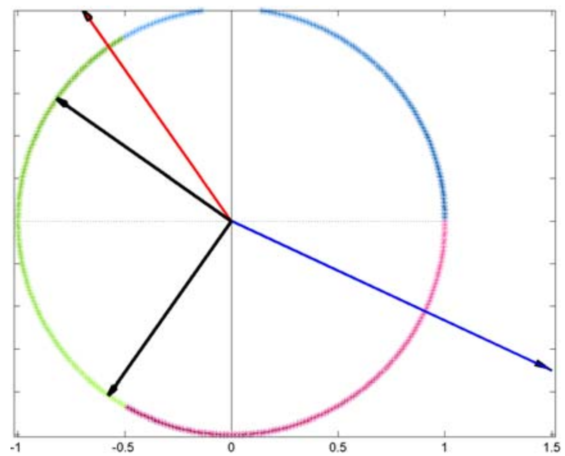
Symmetric Matrices

$$\begin{bmatrix} 1.5 & -0.7 \\ -0.7 & 1 \end{bmatrix}$$

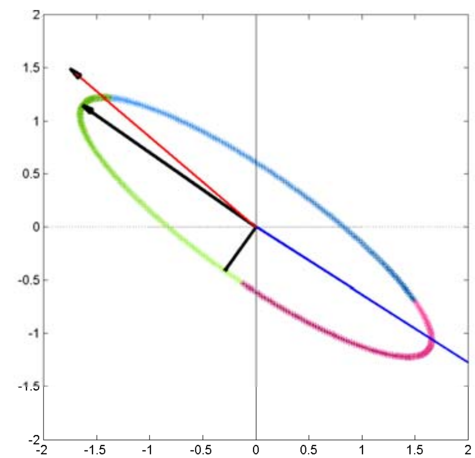


- Matrices that do not change on transposition
  - Row and column vectors are identical
- Symmetric matrix: Eigen vectors and Eigen values are always real
- Eigen vectors are always orthogonal
  - At 90 degrees to one another

Symmetric Matrices



$$\begin{bmatrix} 1.5 & -0.7 \\ -0.7 & 1 \end{bmatrix}$$



- Eigen vectors point in the direction of the major and minor axes of the ellipsoid resulting from the transformation of a spheroid
  - The eigen values are the lengths of the axes

Symmetric matrices

- Eigen vectors V_i are orthonormal
  - $V_i^T V_i = 1$
  - $V_i^T V_j = 0$, i != j

- Listing all eigen vectors in matrix form V
  - $V^T = V^{-1}$
  - $V^T V = I$
  - $V V^T = I$

- $M V_i = \lambda V_i$

- In matrix form : $M V = V L$
  - L is a diagonal matrix with all eigen values

- $M = V L V^T$