Linear Algebra for Communications: A gentle introduction

Shivkumar Kalyanaraman

Linear Algebra has become as basic and as applicable as calculus, and fortunately it is easier.

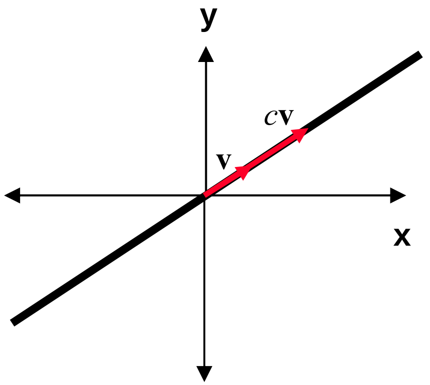
--Gilbert Strang, MIT

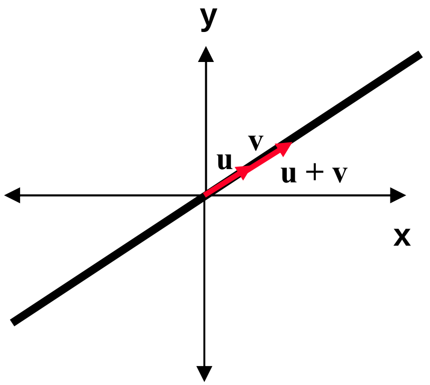
Outline

- What is linear algebra, really? Vector? Matrix? Why care?
- Basis, projections, orthonormal basis
- Algebra operations: addition, scaling, multiplication, inverse
- Matrices: translation, rotation, reflection, shear, projection etc
  - Symmetric/Hermitian, positive definite matrices
- Decompositions:
  - Eigen-decomposition: eigenvector/value, invariants
  - Singular Value Decomposition (SVD).
- **Sneak peeks:** how do these concepts relate to communications ideas: fourier transform, least squares, transfer functions, matched filter, solving differential equations etc

What is "Linear" & "Algebra"?

- Properties satisfied by a *line through the origin* ("one-dimensional case").
  - A directed arrow from the origin (**v**) on the line, when scaled by a constant (*c*) remains on the line
  - Two directed arrows (**u** and **v**) on the line can be "added" to create a longer directed arrow (**u** + **v**) in the same line.





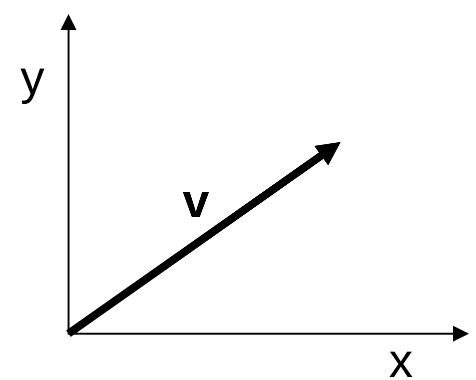
- Wait a minute! This is nothing but *arithmetic with symbols*!
  - "Algebra": generalization and extension of arithmetic.
  - "Linear" operations: addition and scaling.
- Abstract and Generalize !
  - "Line" ↔ **vector space** having N dimensions
  - "Point" ↔ **vector** with N components in each of the N dimensions (***basis*** vectors).
    - Vectors have: "Length" and "Direction".
    - Basis vectors: "span" or define the space & its dimensionality.
  - Linear function transforming vectors ↔ **matrix**.
    - The function acts on each vector component and scales it
    - Add up the resulting scaled components to get a new vector!
    - In general: $f(c\mathbf{u} + d\mathbf{v}) = cf(\mathbf{u}) + df(\mathbf{v})$

What is a Vector ?

- Think of a vector as a *directed line segment in N-dimensions*! (has “length” and “direction”)

$$\vec{v} = \begin{bmatrix} a \\ b \\ c \end{bmatrix}$$

- Basic idea: convert geometry in higher dimensions into algebra!
  - Once you define a “nice” *basis* along each dimension: x-, y-, z-axis …
  - Vector becomes a 1 x N matrix!
  - $\mathbf{v} = [a \;\; b \;\; c]^T$
  - Geometry starts to become linear algebra on vectors like **v**!



Examples of Geometry becoming Algebra

- Lines are vectors through the origin, scaled and translated: m$\mathbf{x}$ + $\mathbf{c}$
  - Intersection of lines can be modeled as addition of vectors: solution of linear equations.
- Linear transformations of vectors can be associated with a matrix **A**, whose columns represent how each basis vector is transformed.
- Ellipses and conic sections:
  - $ax^2 + 2bxy + cy^2 = d$
  - Let $\mathbf{x} = [x\ y]^T$ and A is a symmetric matrix with rows $[a\ b]^T$ and $[b\ c]^T$
  - $\mathbf{x}^T\mathbf{A}\mathbf{x} = c$ {quadratic form equation for ellipse!}
    - This becomes convenient at higher dimensions
  - Note how a *symmetric* matrix A naturally arises from such a homogenous multivariate equation…

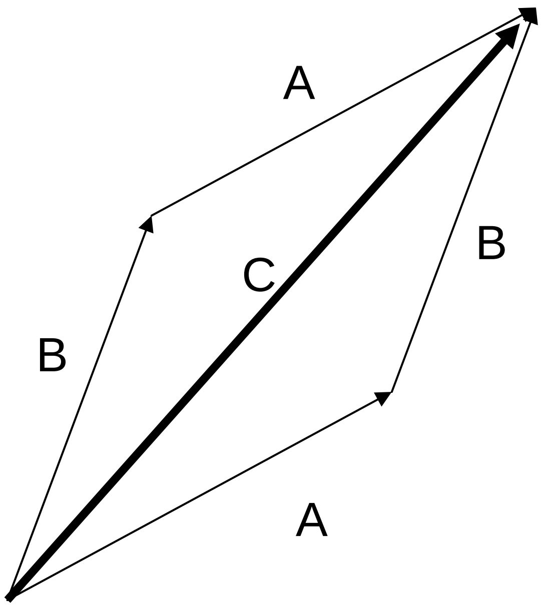
Scalar vs Matrix Equations

- Line equation: $y = mx + c$
- Matrix equation: $\mathbf{y} = M\mathbf{x} + c$

- Second order equations:
  - $\mathbf{x^T M x} = c$
  - $\mathbf{y = (x^T M x) u + M x}$
  - … involves *quadratic forms* like $\mathbf{x^T M x}$

Vector Addition: A+B

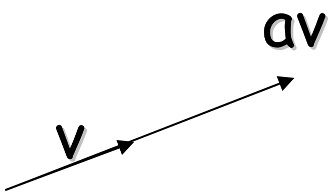
$$\mathbf{A+B} = (x_1, x_2) + (y_1, y_2) = (x_1 + y_1, x_2 + y_2)$$



A+B = C
(use the head-to-tail method to combine vectors)

Scalar Product: $a\mathbf{v}$

$$a\mathbf{v} = a(x_1, x_2) = (ax_1, ax_2)$$



Change only the length ("scaling"), but keep *direction fixed*.

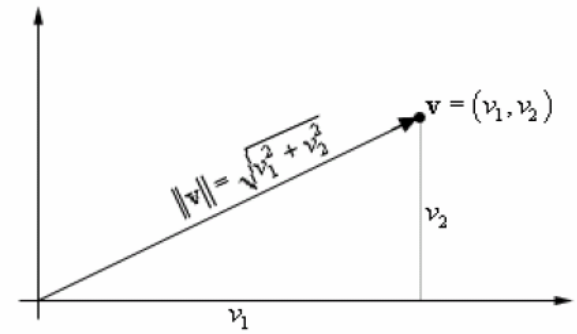
Sneak peek: matrix operation (**Av**) can change *length, direction and also dimensionality*!

Vectors: Magnitude (Length) and Phase (direction)

$$v = (x_1, x_2, \ldots, x_n)^T$$

$$\|v\| = \sqrt{\sum_{i=1}^{n} x_i^2}$$ (**<u>Magnitude or "2-norm"</u>**)

If $\|v\| = 1$, v is a unit vecto r



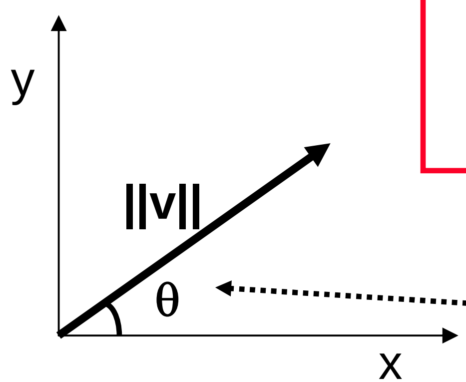
$\mathbf{u} = \frac{1}{\|\mathbf{v}\|}\mathbf{v}$, then $\mathbf{u}$ is a unit vector.

(unit vector => **<u>pure direction</u>**)

Alternate representations:

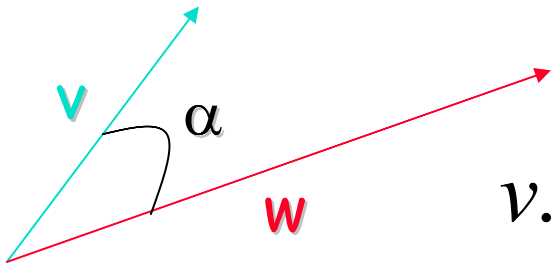
Polar coords: (**||v||**, θ)

Complex numbers: **||v||**$e^{j\theta}$



"phase"

Inner (dot) Product: v.w or $w^T v$



$$v.w = (x_1, x_2).(y_1, y_2) = x_1 y_1 + x_2 . y_2$$

The inner product is a **SCALAR!**

$$v.w = (x_1, x_2).(y_1, y_2) = \|v\| \cdot \|w\| \cos\alpha$$

$$v.w = 0 \Leftrightarrow v \perp w$$

If vectors **v, w** are "columns", then dot product is $\mathbf{w}^T\mathbf{v}$

Inner Products, Norms: *Signal space*

- Signals modeled as vectors in a vector space: "signal space"
- To form a signal space, first we need to know the inner product between two signals (functions):
  - Inner (scalar) product: (generalized for functions)

$$< x(t), y(t) >= \int_{-\infty}^{\infty} x(t) y^*(t) dt$$

= cross-correlation between x(t) and y(t)

  - Properties of inner product:

$$< ax(t), y(t) >= a < x(t), y(t) >$$

$$< x(t), ay(t) >= a^* < x(t), y(t) >$$

$$< x(t)+y(t), z(t) >=< x(t), z(t) > + < y(t), z(t) >$$

Signal space …

- The distance in signal space is measure by calculating the norm.
- What is norm?
  - Norm of a signal (generalization of "length"):

$$\|x(t)\| = \sqrt{<x(t), x(t)>} = \sqrt{\int_{-\infty}^{\infty} |x(t)|^2 dt} = \sqrt{E_x}$$

= "length" of x(t)

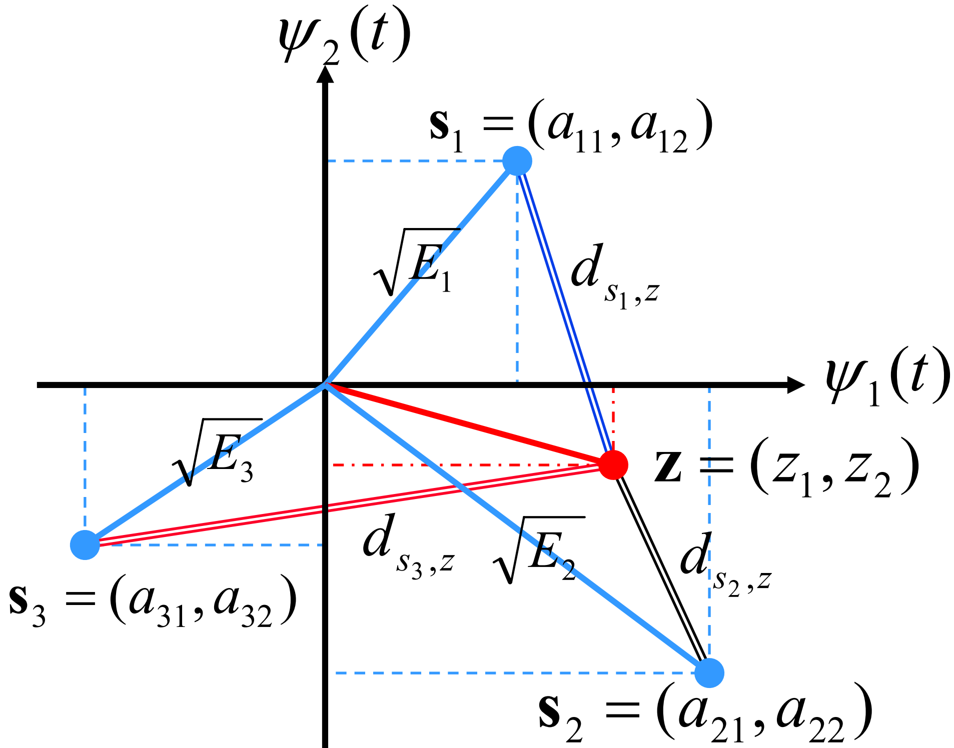
$$\|ax(t)\| = |a|\|x(t)\|$$

  - Norm between two signals:

$$d_{x,y} = \|x(t) - y(t)\|$$

- We refer to the norm between two signals as the Euclidean distance between two signals.

Example of distances in signal space



The Euclidean distance between signals *z(t)* and *s(t)*:

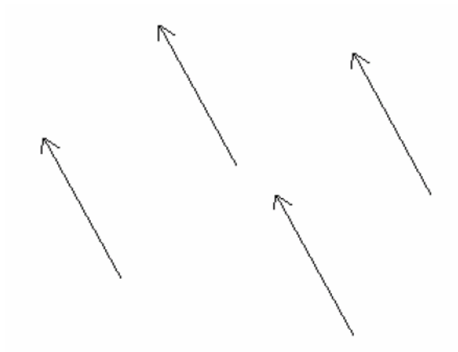
$$d_{s_i,z} = \| s_i(t) - z(t) \| = \sqrt{(a_{i1} - z_1)^2 + (a_{i2} - z_2)^2}$$

$$i = 1,2,3$$

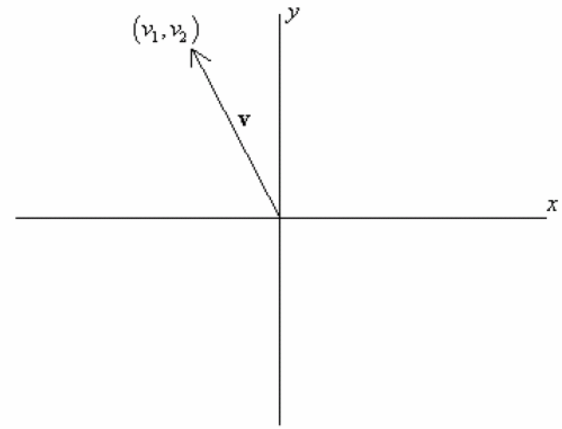
Detection in AWGN noise: Pick the "closest" signal vector

Bases & Orthonormal Bases

- Basis (or axes): frame of reference



vs



Basis: a space is totally defined by a set of vectors – any point is a *linear combination* of the basis

Ortho-Normal: orthogonal + normal

[**Sneak peek**:

Orthogonal: dot product is zero

Normal: magnitude is one]

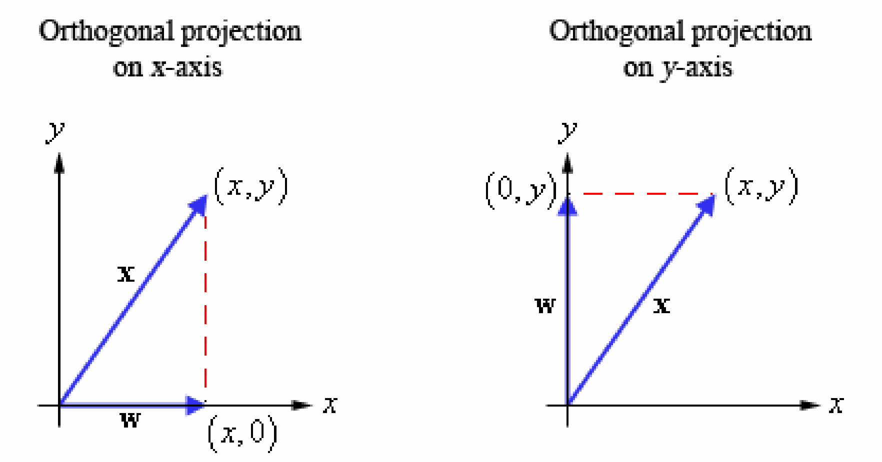
$$x = \begin{bmatrix}1 & 0 & 0\end{bmatrix}^T \qquad x \cdot y = 0$$

$$y = \begin{bmatrix}0 & 1 & 0\end{bmatrix}^T \qquad x \cdot z = 0$$

$$z = \begin{bmatrix}0 & 0 & 1\end{bmatrix}^T \qquad y \cdot z = 0$$

Projections w/ Orthogonal Basis

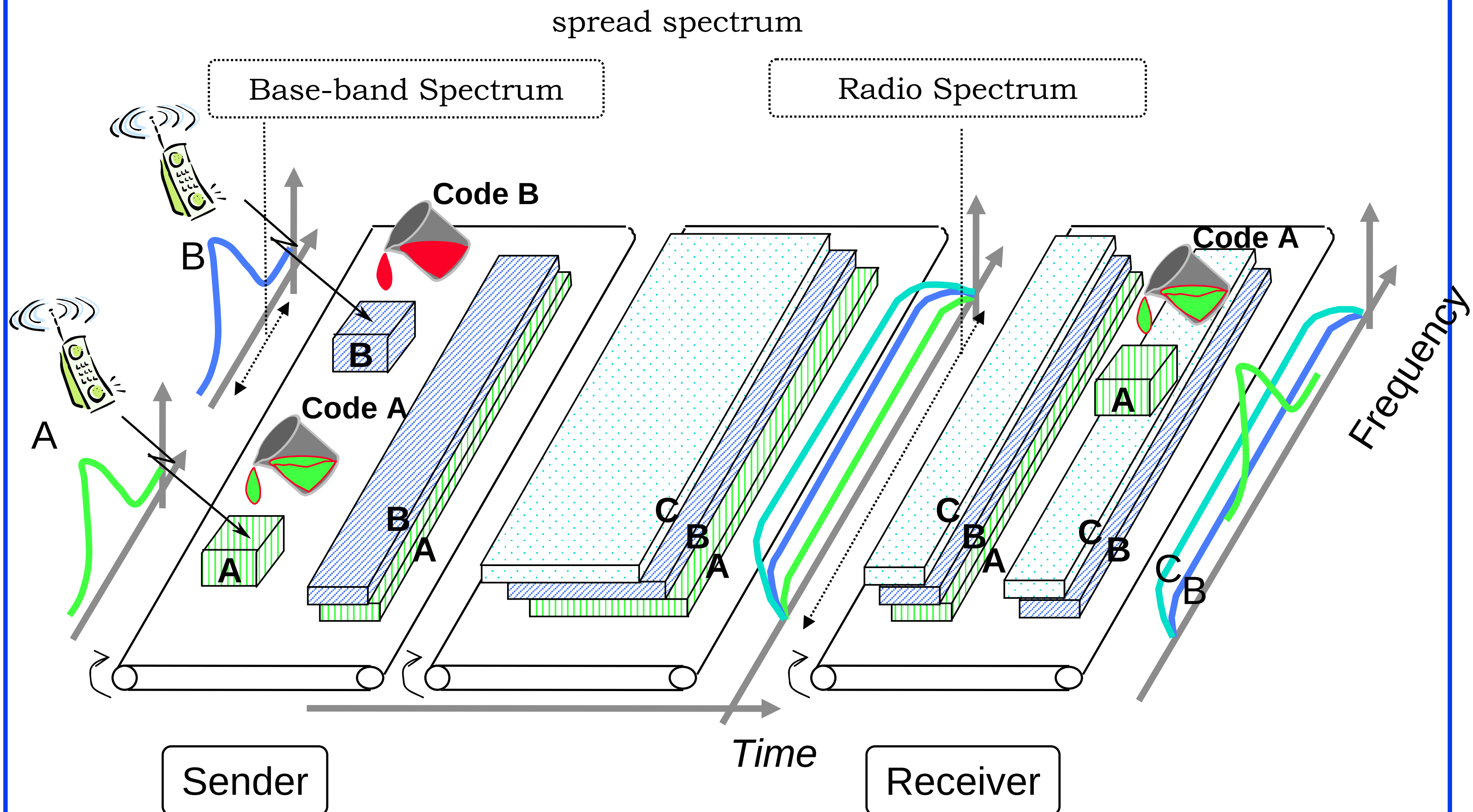
- Get the component of the vector on each axis:
  - dot-product with unit vector on each axis!



Sneak peek: this is what Fourier transform does!
Projects a function onto a **infinite** number of orthonormal basis **functions: ($e^{j\omega}$ or $e^{j2\pi n\theta}$)**, and adds the results up (to get an equivalent "representation" in the "frequency" domain).

CDMA codes are "orthogonal", and projecting the composite received signal on each code helps extract the symbol transmitted on that code.

Orthogonal Projections: CDMA, Spread Spectrum



Each "code" is an orthogonal basis vector => signals sent are orthogonal

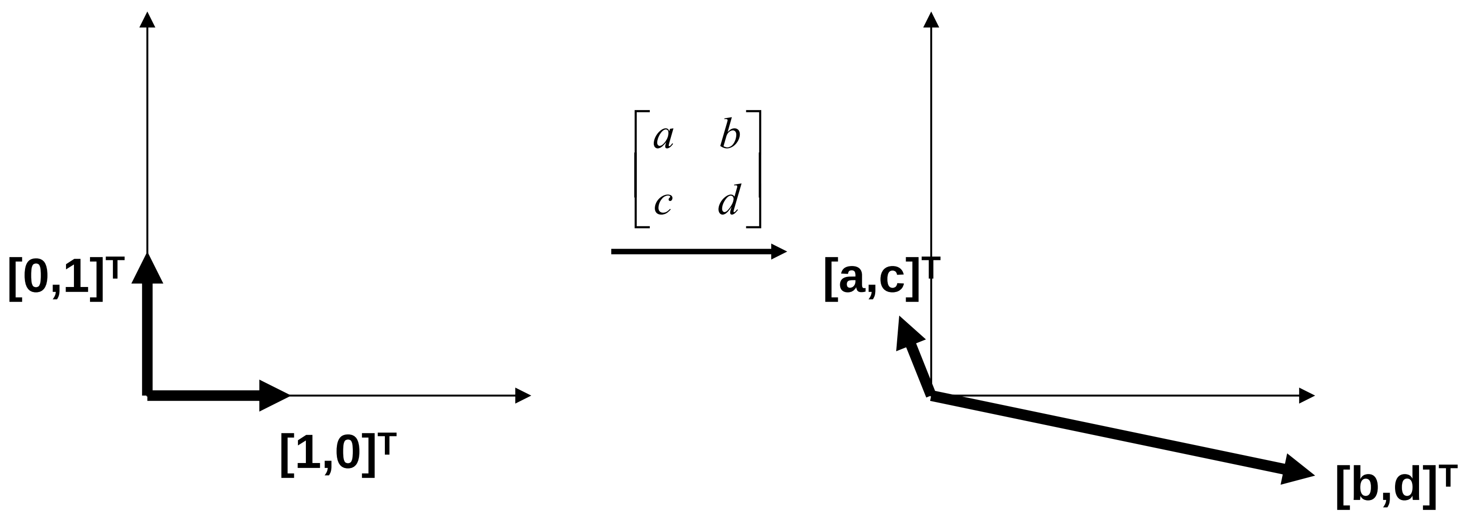
What is a Matrix?

- A matrix is a set of elements, organized into rows and columns

$$\begin{array}{cc} & \text{rows} \longrightarrow \\ \text{columns} \downarrow & \begin{bmatrix} a & b \\ c & d \end{bmatrix} \end{array}$$

What is a Matrix? (Geometrically)

- Matrix represents a *linear* function acting on vectors:
  - Linearity (a.k.a. superposition): $f(au + bv) = af(u) + bf(v)$
- f transforms the unit x-axis basis vector $\boldsymbol{i} = [1\ 0]^T$ **to** $[a\ c]^T$
- f transforms the unit y-axis basis vector $\boldsymbol{j} = [0\ 1]^T$ **to** $[b\ d]^T$
- f can be <u>*represented*</u> by the matrix with $[a\ c]^T$ and $[b\ d]^T$ as columns
  - Why? $f(w = m\boldsymbol{i} + n\boldsymbol{j}) = A[m\ n]^T$
- <u>*Column*</u> viewpoint: focus on the columns of the matrix!



<u>Linear</u> Functions f : <u>***Rotate***</u> and/or <u>***stretch/shrink***</u> the basis vectors

Matrix operating on vectors

- Matrix is like a <u>function</u> that <u>transforms the vectors on a plane</u>
- Matrix operating on a general point => transforms x- and y-components
- *System of linear equations*: matrix is just the bunch of coeffs !

- x' = ax + by
- y' = cx + dy

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} x' \\ y' \end{bmatrix}$$

- Vector (column) viewpoint:
  - New **basis vector [a c]^T** is scaled by x, and added to:
  - New **basis vector [b d]^T** scaled by y
  - i.e. a <u>*linear combination of columns of A*</u> to get [x' y']^T

- For larger dimensions this "column" or vector-addition viewpoint is better than the "row" viewpoint involving hyper-planes (that intersect to give a solution of a set of linear equations)

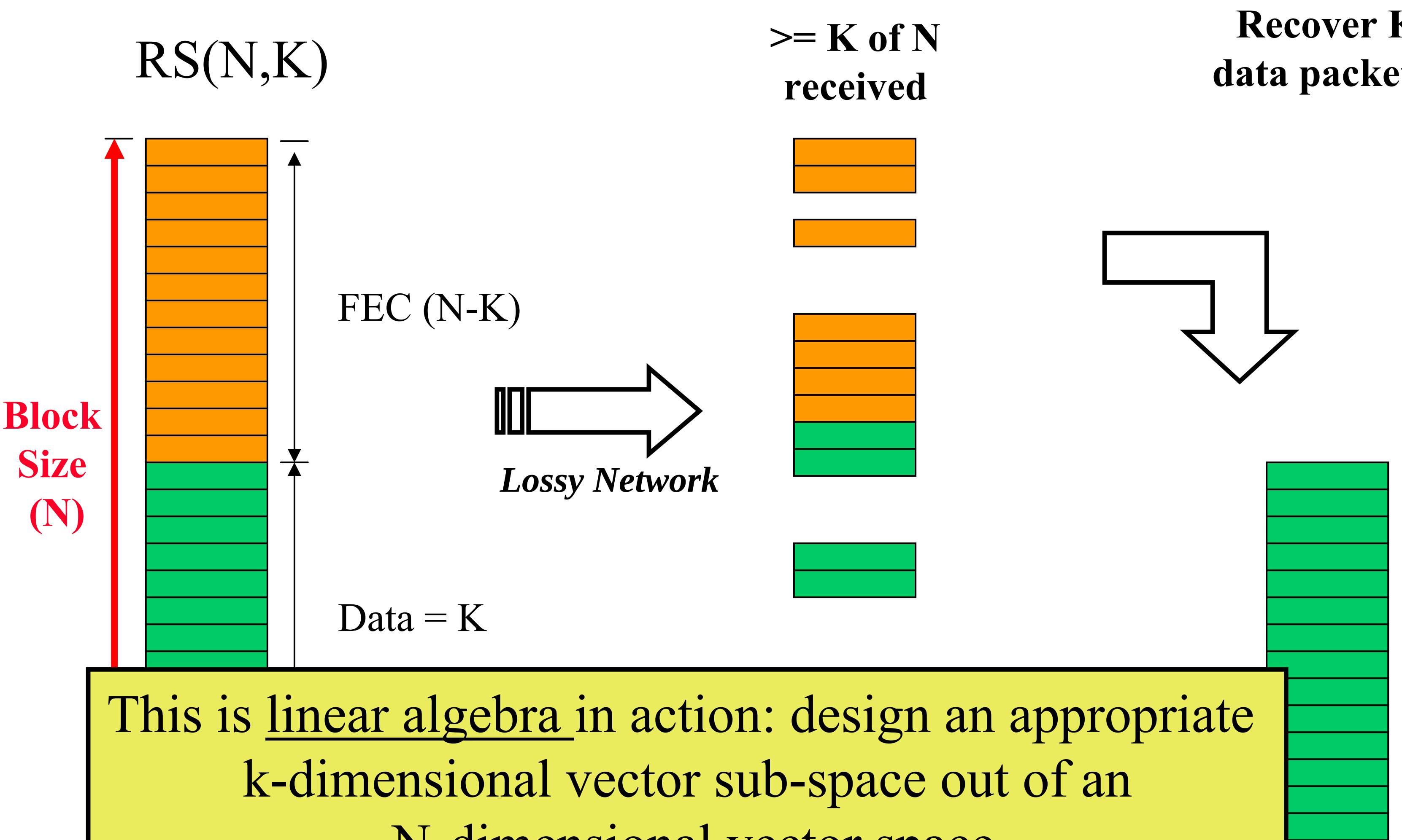
Vector Spaces, Dimension, Span

- Another way to view Ax = b, is that a solution exists for all vectors b that lie in the "*column space*" of A,
  - i.e. b is a linear combination of the basis vectors represented by the columns of A
  - The columns of A "*span*" the "column" space
  - The dimension of the column space is the *column rank (or rank)* of matrix A.

- In general, given a bunch of vectors, they span a *vector space*.
  - There are some "algebraic" considerations such as closure, zero etc
  - The *dimension* of the space is maximal only when the vectors are *linearly independent* of the others.
  - Subspaces are vector spaces with lower dimension that are a subset of the original space

- **Sneak Peek:** linear channel codes (eg: Hamming, Reed-solomon, BCH) can be viewed as k-dimensional vector sub-spaces of a larger N-dimensional space.
  - k-data bits can therefore be protected with N-k parity bits

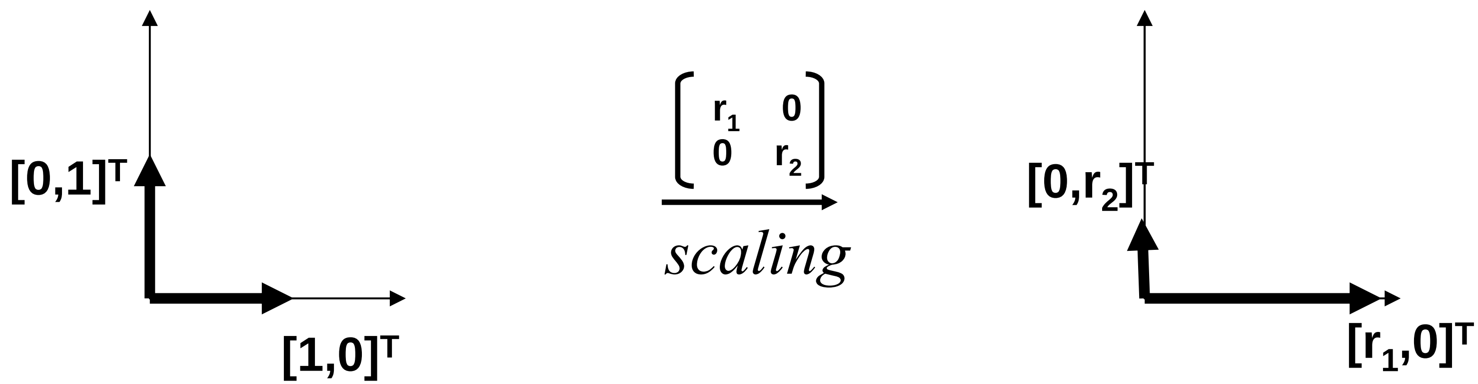
Forward Error Correction (FEC): Eg: Reed-Solomon RS(N,K)

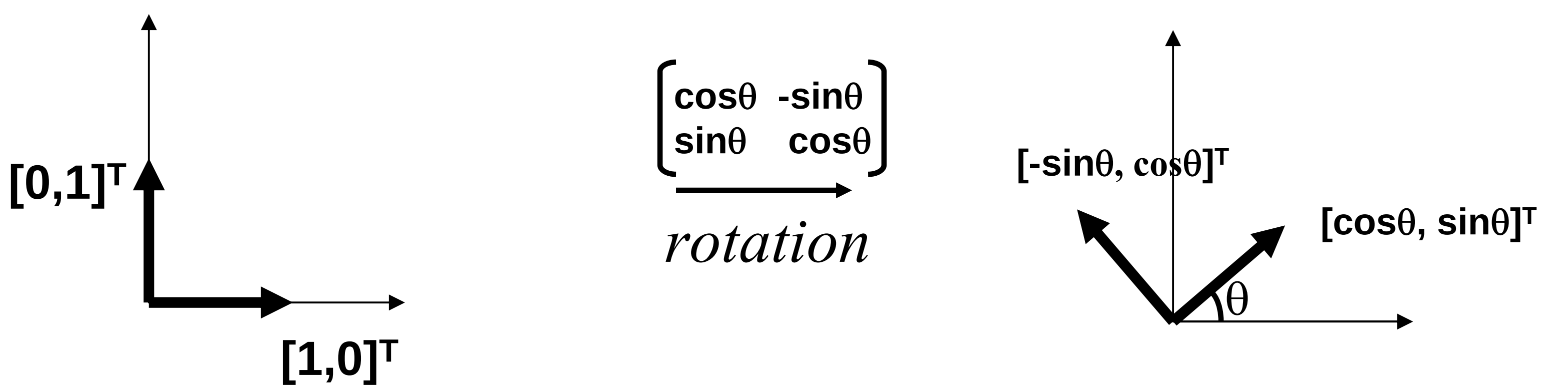


This is linear algebra in action: design an appropriate k-dimensional vector sub-space out of an N-dimensional vector space

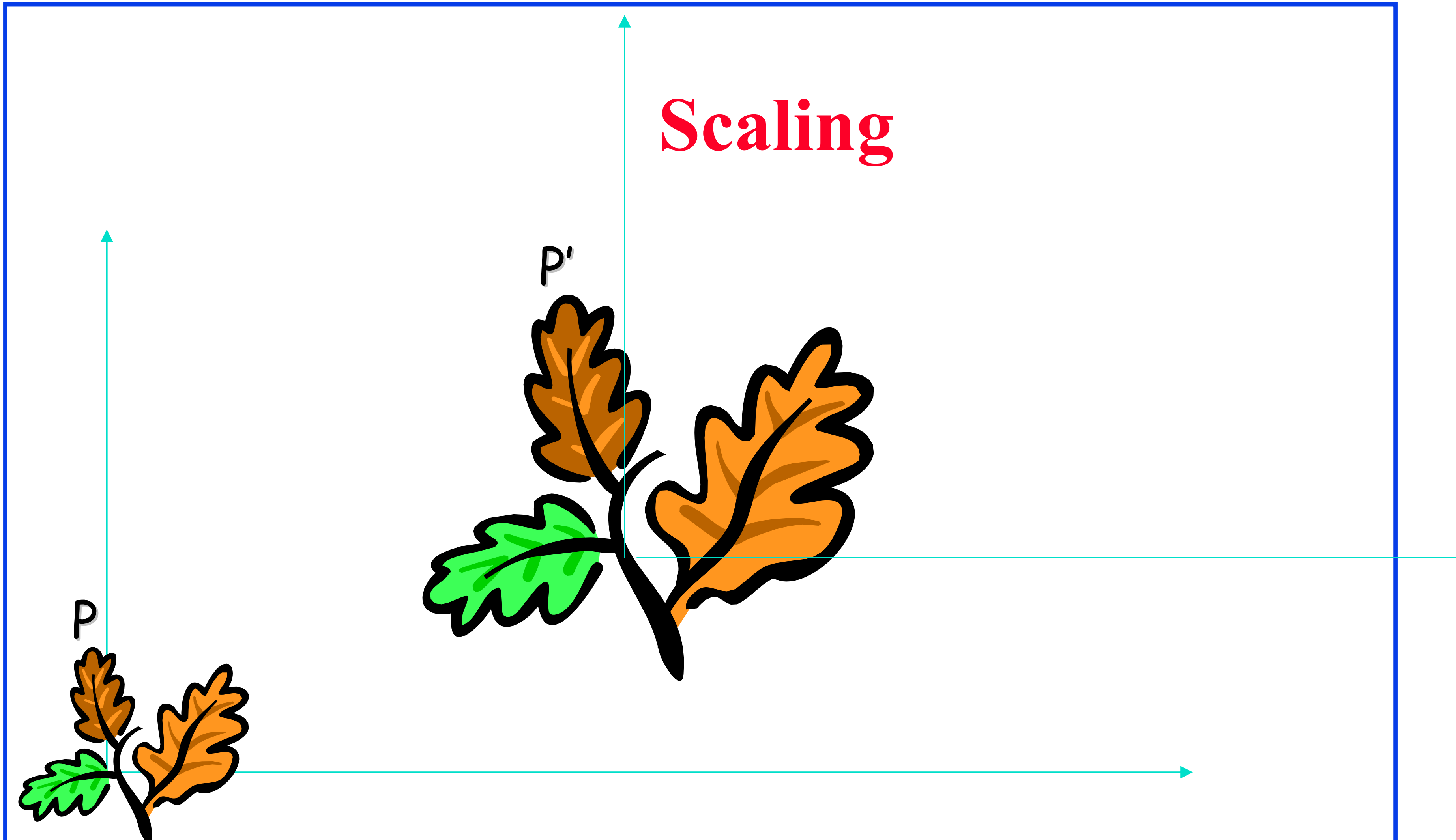
Matrices: Scaling, Rotation, Identity

- Pure scaling, no rotation => "**diagonal** matrix" (note: x-, y-axes could be scaled differently!)
- Pure rotation, no stretching => "**orthogonal** matrix" **O**
- **Identity** ("do nothing") matrix = unit scaling, no rotation!





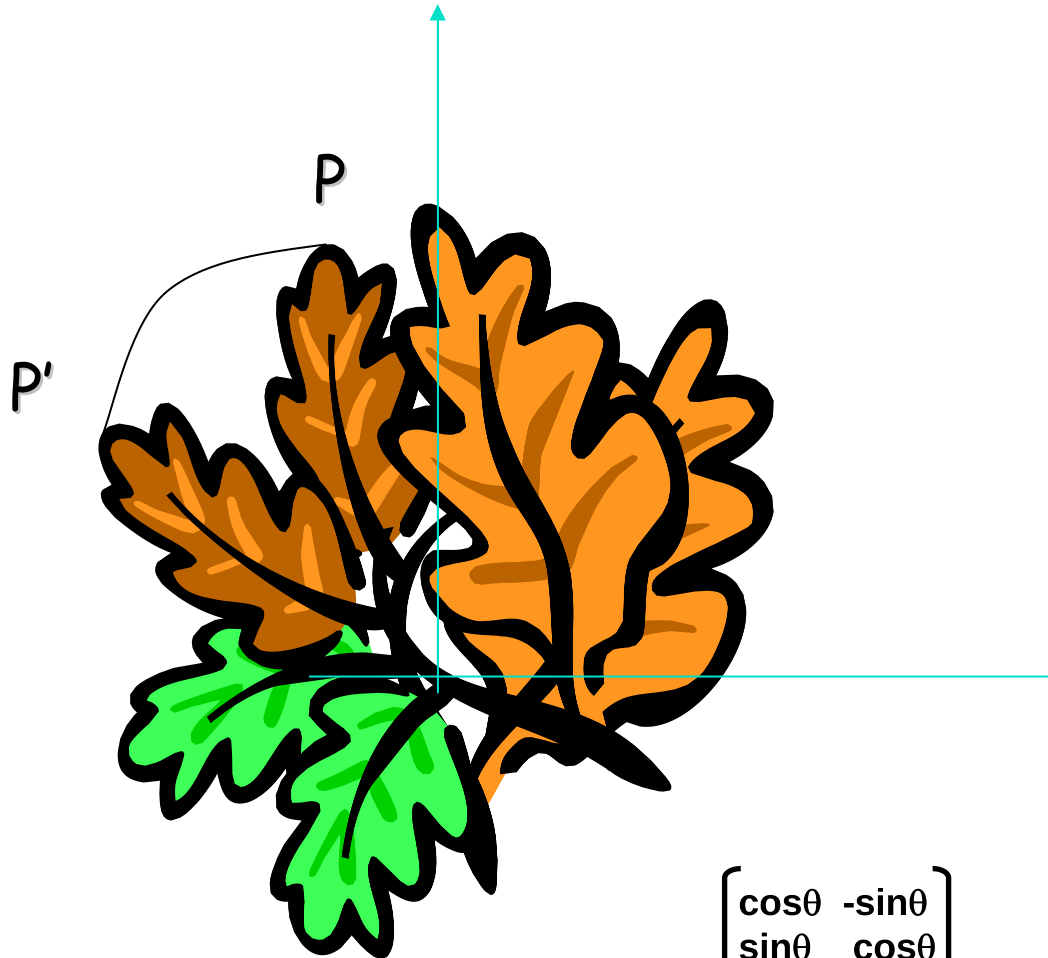
Scaling



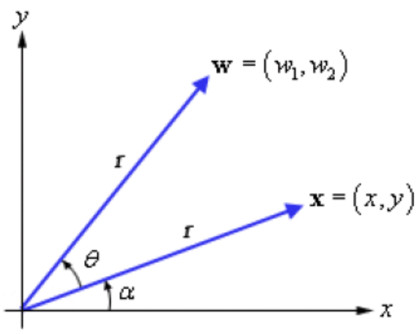
$$\begin{bmatrix} r & 0 \\ 0 & r \end{bmatrix}$$

a.k.a: dilation (r >1), contraction (r <1)

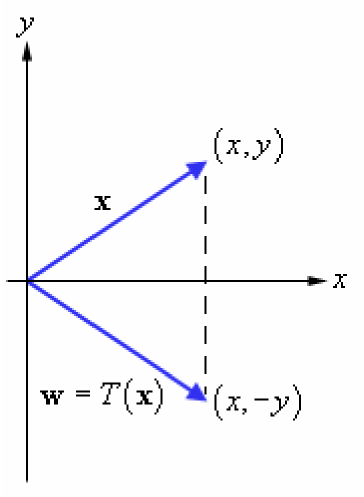
Rotation



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

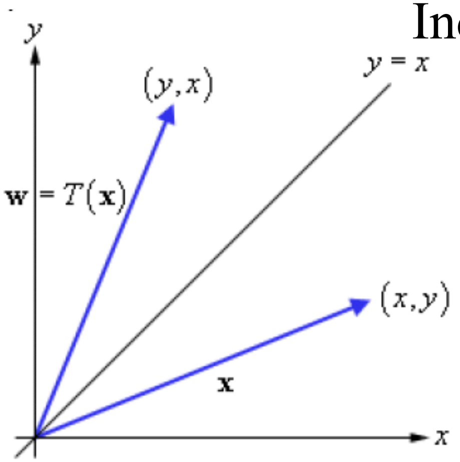


Reflections



- Reflection can be about any line or point.
- <u>Complex Conjugate:</u> reflection about x-axis (i.e. flip the phase θ to -θ)
- Reflection => two times the projection distance from the line.
- Reflection does not affect magnitude

$$\begin{bmatrix} w_1 \\ w_2 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$



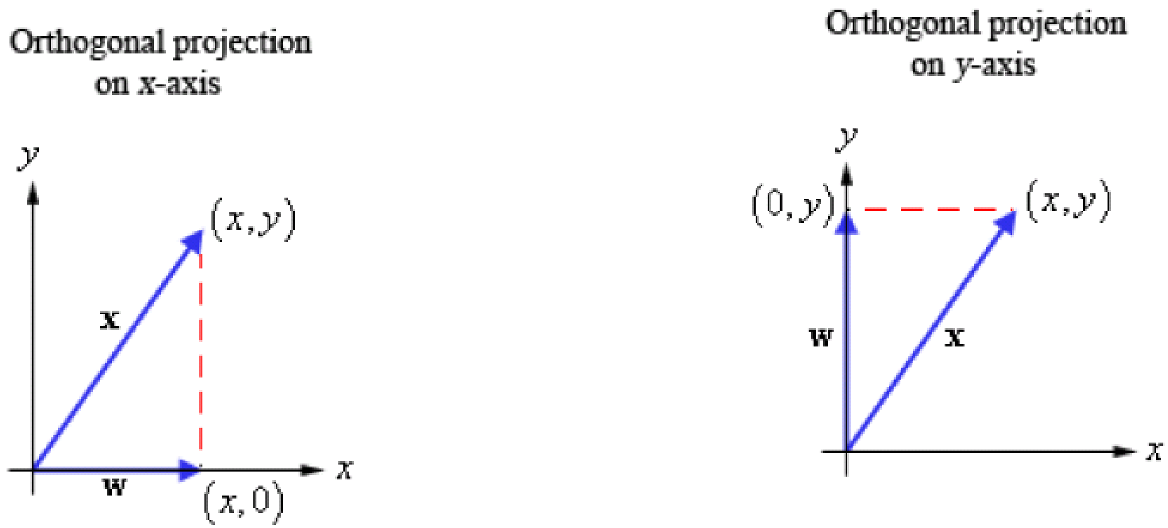
$w_1 = y$
$w_2 = x$

Induced Matrix

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

Reflection about x-axis in $\mathbb{R}^2$	$w_1 = x$ $w_2 = -y$	$\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$
Reflection about y-axis in $\mathbb{R}^2$	$w_1 = -x$ $w_2 = y$	$\begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}$
Reflection about line $x = y$ in $\mathbb{R}^2$	$w_1 = y$ $w_2 = x$	$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$
Reflection about origin in $\mathbb{R}^2$	$w_1 = -x$ $w_2 = -y$	$\begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix}$

Orthogonal Projections: Matrices

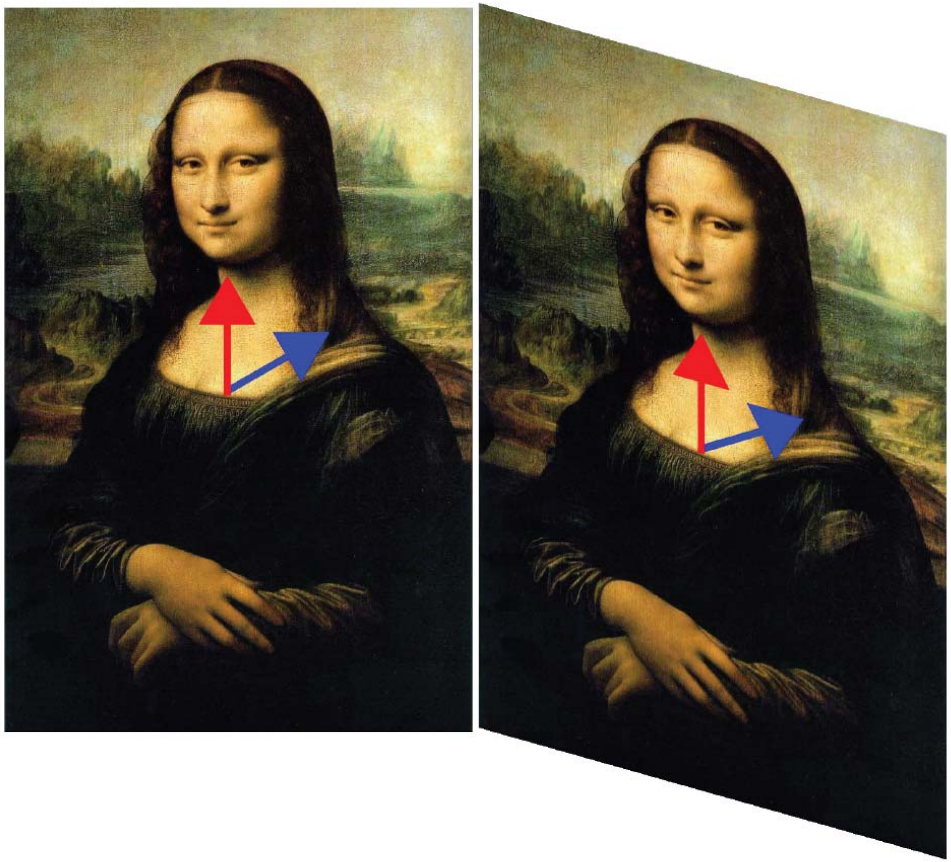


Orthogonal Projection	Equations	Induced Matrix
Projection on x-axis in $\mathbb{R}^2$	$w_1 = x$ $w_2 = 0$	$\begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}$
Projection on y-axis in $\mathbb{R}^2$	$w_1 = 0$ $w_2 = y$	$\begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}$

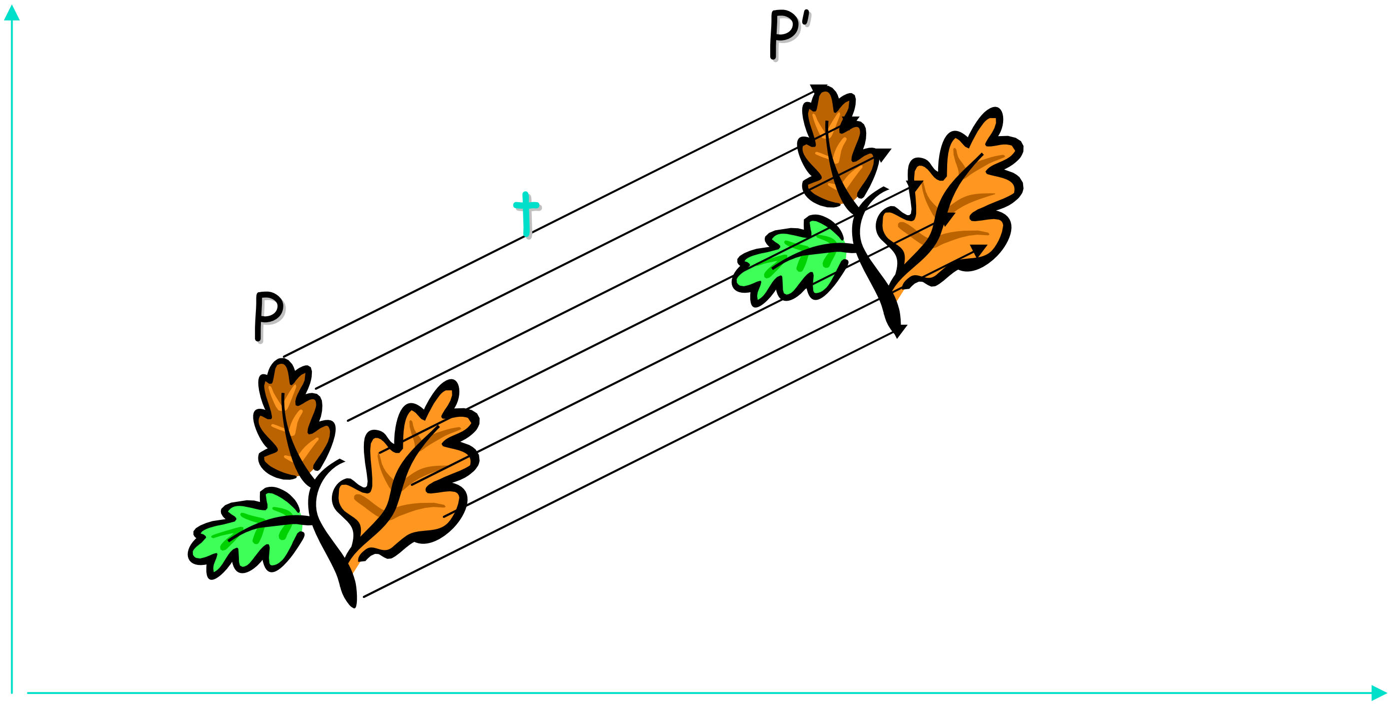
Shear Transformations

- Hold one direction constant and transform ("pull") the other direction

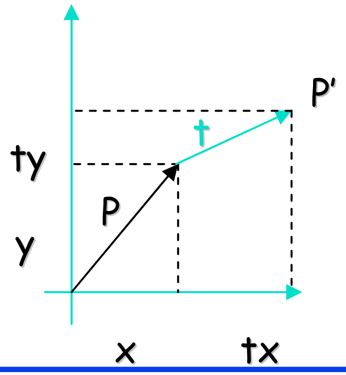
$$\begin{bmatrix} 1 & 0 \\ -0.5 & 1 \end{bmatrix}$$



2D Translation



$$\mathbf{P}' = (x + t_x, y + t_y) = \mathbf{P} + \mathbf{t}$$



Basic Matrix Operations

- Addition, Subtraction, Multiplication: creating new matrices (or functions)

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

Just add elements

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

Just subtract elements

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} e & f \\ g & h \end{bmatrix} = \begin{bmatrix} ae+bg & af+bh \\ ce+dg & cf+dh \end{bmatrix}$$

Multiply each row by each column

Multiplication

- Is AB = BA? Maybe, but maybe not!

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix}\begin{bmatrix} e & f \\ g & h \end{bmatrix} = \begin{bmatrix} ae+bg & \dots \\ \dots & \dots \end{bmatrix} \qquad \begin{bmatrix} e & f \\ g & h \end{bmatrix}\begin{bmatrix} a & b \\ c & d \end{bmatrix} = \begin{bmatrix} ea+fc & \dots \\ \dots & \dots \end{bmatrix}$$

- Matrix multiplication AB: apply transformation B first, and then again transform using A!
- Heads up: multiplication is NOT commutative!

- **Note**: If A and B both represent either pure "*rotation*" or "*scaling*" they can be interchanged (i.e. AB = BA)

Multiplication as Composition…

$$\underbrace{\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}}_{\text{Project on } y\text{-axis}} \underbrace{\begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 2 \end{bmatrix}}_{\text{Dilate by 2}} = \underbrace{\begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix}}_{\text{Composition}}$$

$$\underbrace{\begin{bmatrix} \cos 45 & -\sin 45 \\ \sin 45 & \cos 45 \end{bmatrix}}_{\text{Rotate by } 45^\circ} \underbrace{\begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}}_{\substack{\text{Project on} \\ x\text{-axis}}} = \begin{bmatrix} \frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} \frac{\sqrt{2}}{2} & 0 \\ \frac{\sqrt{2}}{2} & 0 \end{bmatrix}$$

Different!

$$\underbrace{\begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}}_{\substack{\text{Project on} \\ x\text{-axis}}} \underbrace{\begin{bmatrix} \cos 45 & -\sin 45 \\ \sin 45 & \cos 45 \end{bmatrix}}_{\text{Rotate by } 45^\circ} = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \end{bmatrix} = \begin{bmatrix} \frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} \\ 0 & 0 \end{bmatrix}$$

Inverse of a Matrix

- Identity matrix:
  $\mathbf{AI = A}$
- Inverse exists only for square matrices that are non-singular
  - Maps N-d space to another N-d space bijectively
- Some matrices have an inverse, such that:
  $\mathbf{AA^{-1} = I}$
- Inversion is tricky:
  $\mathbf{(ABC)^{-1} = C^{-1}B^{-1}A^{-1}}$

  Derived from non-commutativity property

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

Determinant of a Matrix

- Used for inversion
- If det(A) = 0, then A has no inverse
- Can be found using factorials, pivots, and cofactors!
- "Volume" interpretation

$$A = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

$$\det(A) = ad - bc$$

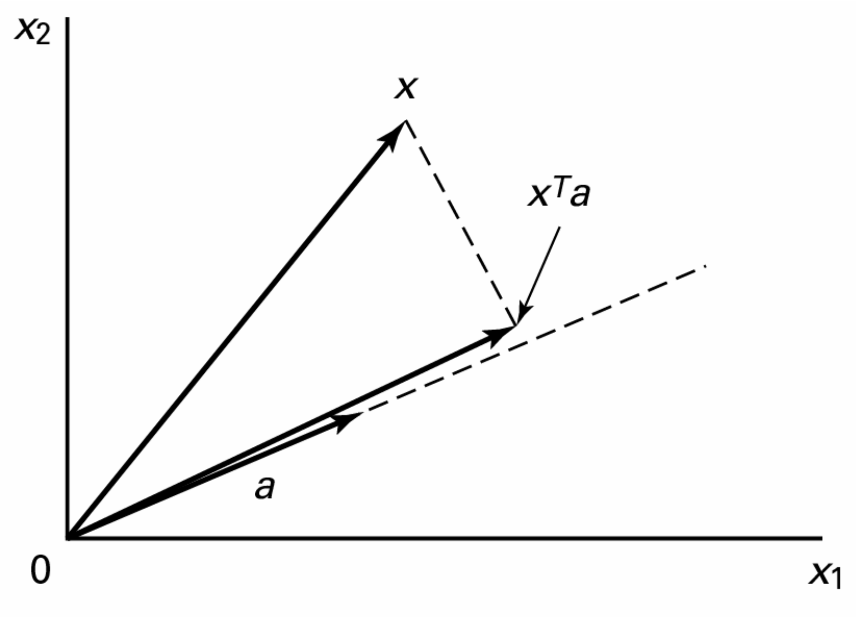
<u>Sneak Peek:</u> Determinant-criterion for space-time code design.

- Good code exploiting time diversity should maximize the minimum product distance between codewords.
- Coding gain determined by min of determinant over code words.

$$\det[(\mathbf{X}_A - \mathbf{X}_B)(\mathbf{X}_A - \mathbf{X}_B)^*]$$

$$A^{-1} = \frac{1}{ad - bc}\begin{bmatrix} d & -b \\ -c & a \end{bmatrix}$$

Projection: Using Inner Products (I)

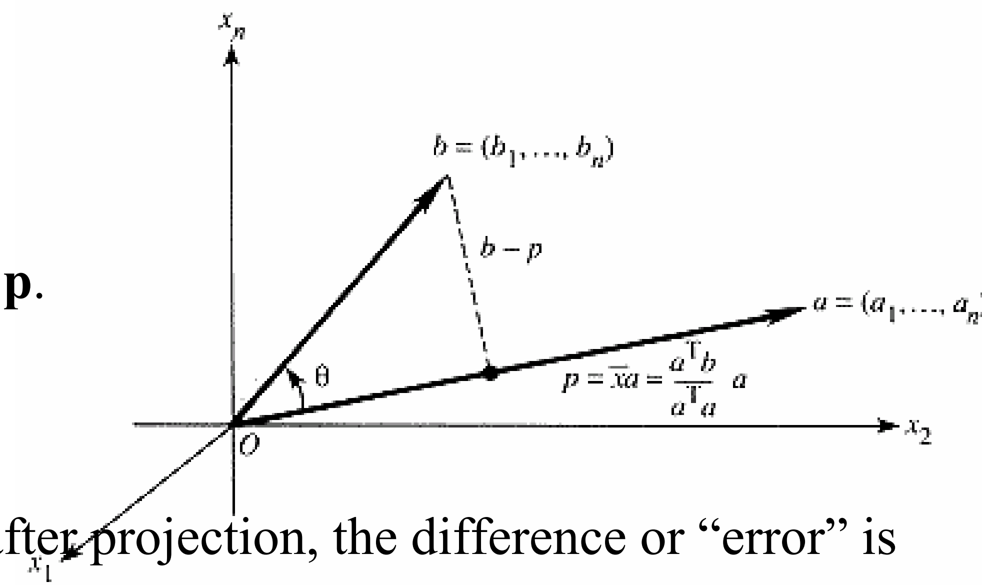


Projection of x along the direction **a** ($\|\mathbf{a}\| = 1$).

$$\mathbf{p} = \mathbf{a}\,(a^T x)$$
$$\|a\| = a^T a = 1$$

Projection: Using Inner Products (II)

$\boxed{\mathbf{p} = \mathbf{a}\,(a^T b)/\,(a^T a)}$



Note: the "error vector" **e** = **b-p** is orthogonal (perpendicular) to **p**.
i.e. Inner product: $\mathbf{(b-p)^T p = 0}$

<u>"Orthogonalization" principle:</u> after projection, the difference or "error" is orthogonal to the projection

<u>**Sneak peek :**</u> we use this idea to find a *"least-squares"* line that minimizes the sum of squared errors (i.e. min $\mathbf{\Sigma e^T e}$).

This is also used in detection under AWGN noise to get the "test statistic":
<u>Idea</u>: *project* the noisy received vector **y** onto (complex) transmit vector **h:**
"matched" filter/max-ratio-combining (MRC)

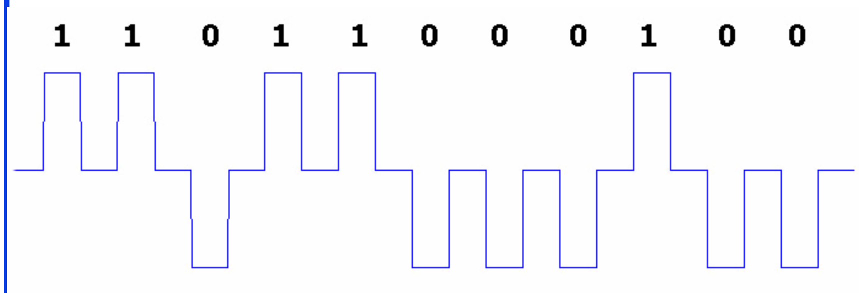
Schwartz Inequality & Matched Filter

- Inner Product ($\mathbf{a}^T\mathbf{x}$) <= Product of Norms (i.e. |**a**||**x**|)
  - Projection length <= Product of Individual Lengths
- This is the <u>*Schwartz Inequality*</u>!
  - Equality happens when **a** and **x** are in the same direction (i.e. $\cos\theta = 1$, when $\theta = 0$)

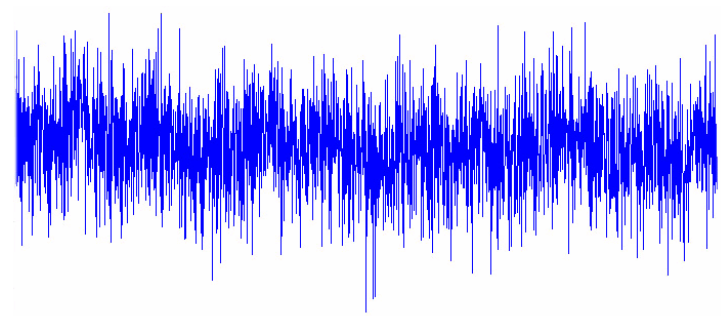
- Application: "matched" filter
  - Received vector **y** = **x** + **w** (zero-mean AWGN)
  - Note: **w** is infinite dimensional
  - Project **y** to the subspace formed by the finite set of transmitted symbols **x**: y'
  - y' is said to be a "<u>**sufficient statistic**</u>" for detection, i.e. reject the noise dimensions outside the signal space.
  - This operation is called "*matching*" to the signal space (projecting)
  - Now, pick the x which is closest to y' in distance (ML detection = nearest neighbor)

Matched Filter Receiver: Pictorially…

Transmitted Signal



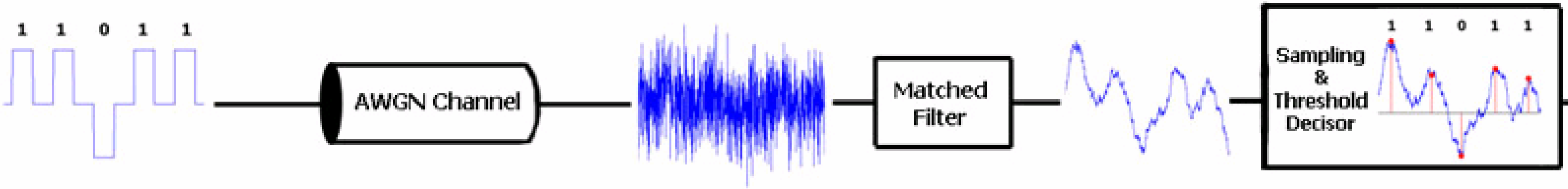
Received Signal (w/ Noise)



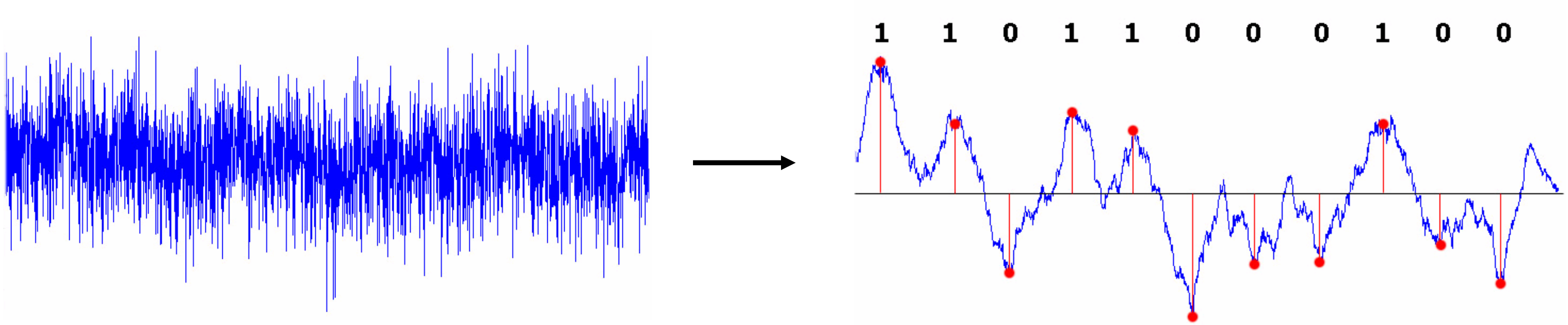
Signal + AWGN noise will not reveal the original transmitted sequence.
There is a high power of noise relative to the power of the desired signal (i.e., low SNR).

If the receiver were to sample this signal at the correct times, the resulting binary message would have a lot of bit errors.

Matched Filter (Contd)



- Consider the received signal as a vector **r,** and the transmitted signal vector as **s**
- Matched filter "projects" the **r** onto signal space spanned by **s** ("matches" it)



Filtered signal can now be safely sampled by the receiver at the correct sampling instants, resulting in a correct interpretation of the binary message

Matched filter is the filter that <u>maximizes the signal-to-noise ratio</u> it can be shown that it also <u>minimizes the BER: *it is a simple projection operation*</u>

Matched Filter w/ Repetition Coding

3.2.1 Repetition Coding

The simplest code is a *repetition code*, in which $x_\ell = x_1$ for $\ell = 1, \ldots, L$. In vector form, the overall channel becomes

$$\mathbf{y} = \mathbf{h}x_1 + \mathbf{w}, \tag{3.32}$$

where $\mathbf{y} = [y_1, \ldots, y_L]^t$, $\mathbf{h} = [h_1, \ldots, h_L]^t$ and $\mathbf{w} = [w_1, \ldots, w_L]^t$.

hx$_1$ only spans a

Consider now coherent detection of x_1, i.e., the channel gains are known to the receiver. This is the canonical vector Gaussian detection problem in Summary 2 of Appendix A. The scalar

||**h**||

$$\frac{\mathbf{h}^*}{\|\mathbf{h}\|}\mathbf{y} = \|\mathbf{h}\|x_1 + \frac{\mathbf{h}^*}{\|\mathbf{h}\|}\mathbf{w} \tag{3.33}$$

is a sufficient statistic. Thus, we have an equivalent scalar detection problem with noise $(\mathbf{h}^*/\|\mathbf{h}\|)\mathbf{w} \sim \mathcal{CN}(0, N_0\mathbf{I}_L)$. The receiver structure is a *matched filter* and is also called a *maximal ratio combiner*: it weighs the received signal in each branch in proportion to the signal strength and also aligns the phases of the signals in the summstion to maximize the output SNR. This receiver structure is also called *coherent combining*.

Multiply by conjugate => cancel phase!

Symmetric, Hermitian, Positive Definite

- **Symmetric**: $\mathbf{A} = \mathbf{A}^T$
  - Symmetric => square matrix
- **Complex vectors/matrices:**
  - Transpose of a vector or a matrix with complex elements must involve a "conjugate transpose", i.e. flip the phase as well.
  - For example: $||\mathbf{x}||^2 = \mathbf{x}^H\mathbf{x}$, where $\mathbf{x}^H$ refers to the conjugate transpose of $\mathbf{x}$
- **Hermitian** (for complex elements): $\mathbf{A} = \mathbf{A}^H$
  - Like symmetric matrix, but must also do a conjugation of each element (i.e. flip its phase).
  - i.e. symmetric, except for flipped phase
  - Note we will use $\mathbf{A}^*$ instead of $\mathbf{A}^H$ for convenience
- **Positive definite:** symmetric, and its quadratic forms are strictly positive, for non-zero $\mathbf{x}$:
  - $\mathbf{x}^T\mathbf{A}\mathbf{x} > 0$
  - Geometry: bowl-shaped minima at $x = 0$

Orthogonal, Unitary Matrices: Rotations

- Rotations and Reflections: **Orthogonal matrices Q**
  - Pure rotation => Changes vector direction, but not magnitude (no scaling effect)
    - Retains dimensionality, and is invertible
  - Inverse rotation is simply Q^T

- **Unitary matrix (U):** complex elements, rotation in complex plane
  - Inverse: $\mathbf{U}^{\mathbf{H}}$ (note: conjugate transpose).

- **Sneak peek:**
  - Gaussian noise exhibits "*isotropy*", i.e. invariance to direction. So any rotation Q of a gaussian vector (w) yields another gaussian vector Qw.
  - *Circular symmetric* (c-s) complex gaussian vector **w** => complex rotation w/ U yields another c-s gaussian vector **Uw**

- **Sneak peek:** The Discrete Fourier Transform (DFT) matrix is both unitary and symmetric.
  - DFT is nothing but a "complex rotation," i.e. viewed in a basis that is a rotated version of the original basis.
  - FFT is just a fast implementation of DFT. It is fundamental in OFDM.

Quadratic forms: $x^T A x$

- Linear:
  - y = mx + c … generalizes to vector equation
  - **y = Mx + c (… y, x, c** are vectors, **M** = matrix**)**
- Quadratic expressions in 1 variable: x^2
  - Vector expression: $\mathbf{x}^T\mathbf{x}$ (… projection!)
  - Quadratic forms generalize this, by allowing a linear transformation A as well
- Multivariable quadratic expression: $x^2 + 2xy + y^2$
- Captured by a symmetric matrix A, and quadratic form:
  - $\mathbf{x}^T\mathbf{A}\mathbf{x}$
- **<u>Sneak Peek:</u>** Gaussian *vector* formula has a quadratic form term in its exponent: $\exp[-0.5\,(\mathbf{x} - \boldsymbol{\mu})^T\, \mathbf{K}^{-1}\, (\mathbf{x} - \boldsymbol{\mu})]$
  - Similar to 1-variable gaussian: $\exp(-0.5\,(x - \mu)^2/\sigma^2)$
  - K^{-1} (inverse covariance matrix) instead of $1/\sigma^2$
  - Quadratic form involving $(\mathbf{x} - \boldsymbol{\mu})$ instead of $(x - \mu)^2$

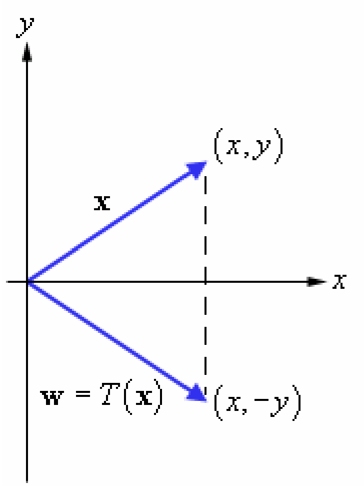
Rectangular Matrices

- Linear system of equations:
  - $\mathbf{Ax = b}$
- More or less equations than necessary.
  - Not full rank
- If full column rank, we can modify equation as:
  - $\mathbf{A^T A x} = \mathbf{A^T b}$
  - Now $(\mathbf{A^T A})$ is square, symmetric and invertible.
  - $x = (\mathbf{A^T A})^{-1} \mathbf{A^T b}$ … now solves the system of equations!
  - This solution is called the *least-squares* solution. Project **b** onto column space and then solve.
- $(\mathbf{A^T A})^{-1} \mathbf{A^T}$ is sometimes called the "*pseudo inverse*"

- **Sneak Peek:** $(\mathbf{A^T A})$ or $(\mathbf{A^* A})$ will appear often in communications math (MIMO). They will also appear in SVD (singular value decomposition)
- The pseudo inverse $(\mathbf{A^T A})^{-1} \mathbf{A^T}$ will appear in decorrelator receivers for MIMO
- More: http://tutorial.math.lamar.edu/AllBrowsers/2318/LeastSquares.asp
  - (or Prof. Gilbert Strang's (MIT) videos on least squares, pseudo inverse):

Invariants of Matrices: Eigenvectors

- Consider a NxN matrix (or linear transformation) T
- An *invariant input* x of a function T(x) is nice because it does not change when the function T is applied to it.
  - i.e. solve this eqn for $\mathbf{x}$: $T(\mathbf{x}) = \mathbf{x}$
- We allow (positive or negative) scaling, but want invariance w.r.t direction:
  - $T(\mathbf{x}) = \lambda\mathbf{x}$
- There are multiple solutions to this equation, equal to the rank of the matrix T. If T is "full" rank, then we have a full set of solutions.
- These invariant solution vectors $\mathbf{x}$ are *eigenvectors*, and the "characteristic" scaling factors associated w/ each $\mathbf{x}$ are *eigenvalues.*



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

E-vectors:

- Points on the x-axis unaffected $[1\ 0]^T$

- Points on y-axis are flipped $[0\ 1]^T$

(but this is equivalent to scaling by -1!)

E-values: 1, -1 (also on diagonal of matrix)

Eigenvectors (contd)

- Eigenvectors are even more interesting because any vector in the domain of T can now be …
  - … viewed in a *new* coordinate system formed with the invariant "eigen" directions as a basis.
  - The operation of T(**x**) is now decomposable into simpler operations on **x**,
  - … which involve projecting **x** onto the "eigen" directions and applying the characteristic (eigenvalue) scaling along those directions

- **Sneak Peek:**
- In fourier transforms (associated w/ linear systems):
  - The unit length phasors $e^{j\omega}$ are the eigenvectors! And the frequency response are the eigenvalues!
  - Why? Linear systems are described by differential equations (i.e. $d/d\omega$ and higher orders)
    - Recall $d\,(e^{j\omega})/d\omega = je^{j\omega}$
    - j is the eigenvalue and $e^{j\omega}$ the eigenvector (actually, an "*eigenfunction*")

Eigenvalues & Eigenvectors

- **Eigenvectors** (for a square $m \times m$ matrix **S**)

$$\mathbf{S}\mathbf{v} = \lambda \mathbf{v}$$

(right) eigenvector $\quad$ eigenvalue

$\mathbf{v} \in \mathbb{R}^m \neq \mathbf{0} \qquad \lambda \in \mathbb{R}$

Example

$$\begin{pmatrix} 6 & -2 \\ 4 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ 2 \end{pmatrix} = \begin{pmatrix} 2 \\ 4 \end{pmatrix} = 2 \begin{pmatrix} 1 \\ 2 \end{pmatrix}$$

- How many eigenvalues are there at most?

$$\mathbf{S}\mathbf{v} = \lambda \mathbf{v} \iff (\mathbf{S} - \lambda \mathbf{I})\, \mathbf{v} = \mathbf{0}$$

only has a non-zero solution if $|\mathbf{S} - \lambda \mathbf{I}| = 0$

this is a m-th order equation in λ which can have **at most m distinct solutions** (roots of the characteristic polynomial) – <u>can be complex even though **S** is real.</u>

Diagonal (Eigen) decomposition – (homework)

Let $$S = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}; \lambda_1 = 1, \lambda_2 = 3.$$

The eigenvectors $\begin{pmatrix} 1 \\ -1 \end{pmatrix}$ *and* $\begin{pmatrix} 1 \\ 1 \end{pmatrix}$ *form* $U = \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix}$

Inverting, we have $U^{-1} = \begin{bmatrix} 1/2 & -1/2 \\ 1/2 & 1/2 \end{bmatrix}$

Recall $UU^{-1} = 1.$

Then, $\mathbf{S=U\Lambda U^{-1}} = \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 3 \end{bmatrix} \begin{bmatrix} 1/2 & -1/2 \\ 1/2 & 1/2 \end{bmatrix}$

Example (homework)

Let's divide ***U*** (and multiply ***U***$^{-1}$) *by* $\sqrt{2}$

Then, **S=** $\begin{bmatrix} 1/\sqrt{2} & 1/\sqrt{2} \\ -1/\sqrt{2} & 1/\sqrt{2} \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 3 \end{bmatrix} \begin{bmatrix} 1/\sqrt{2} & -1/\sqrt{2} \\ 1/\sqrt{2} & 1/\sqrt{2} \end{bmatrix}$

Q $\quad$ Λ $\quad$ ***(Q***$^{-1}$ **=** ***Q***^{T} ***)***

Geometric View: EigenVectors

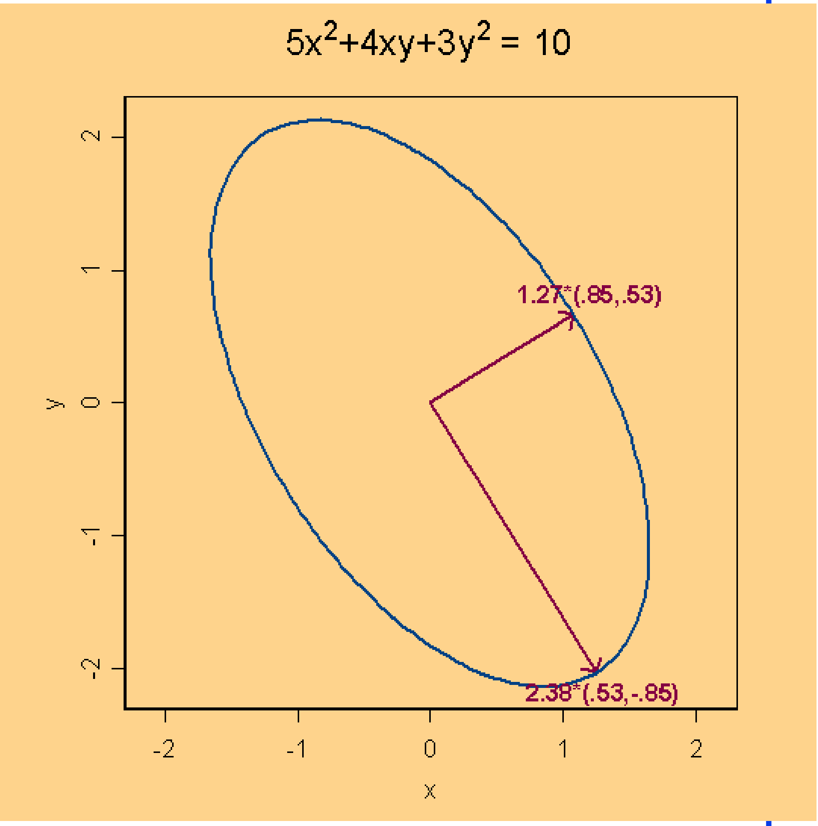
- Homogeneous (2nd order) multivariable equations: $ax^2 + 2kxy + by^2 = c$
- Represented in matrix (quadratic) form w/ symmetric matrix A:

$$\boxed{\mathbf{x}^T \mathbf{A}\mathbf{x} = c,} \quad \text{where} \quad \mathbf{x} = \begin{pmatrix} x \\ y \end{pmatrix}, \ \mathbf{A} = \begin{pmatrix} a & k \\ k & b \end{pmatrix}$$

- **Eigenvector decomposition**:

$$5x^2 + 4xy + 3y^2 = 10 \qquad \mathbf{A} = \begin{pmatrix} 5 & 2 \\ 2 & 3 \end{pmatrix}$$

$$\lambda_1 = 6.24, \mathbf{s}_1 = \begin{pmatrix} 0.85 \\ 0.53 \end{pmatrix} \qquad \lambda_2 = 1.76, s_2 = \begin{pmatrix} 0.53 \\ -0.85 \end{pmatrix}$$



- Geometry: Principal Axes of Ellipse
- Symmetric A => *orthogonal e-vectors*!
- Same idea in fourier transforms
  - E-vectors are "frequencies"
- Positive Definite A => +ve real e-values!

Why do Eigenvalues/vectors matter?

- Eigenvectors are invariants of A
  - Don't change direction when operated A
- Recall $d(e^{\lambda t})/dt = \lambda e^{\lambda t}$.
  - $e^{\lambda t}$ is an invariant function for the linear operator d/dt, with eigenvalue λ
- Pair of differential eqns:
  - $dv/dt = 4v - 5u$
  - $du/dt = 2u - 3w$
- Can be written as: $d\mathbf{y}/dt = \mathbf{A}\mathbf{y}$, where $\mathbf{y} = [v\ u]^T$
  - $\mathbf{y} = [v\ u]^T$ at time $0 = [8\ 5]^T$
- Substitute $\mathbf{y} = e^{\lambda t}\mathbf{x}$ into the equation $d\mathbf{y}/dt = \mathbf{A}\mathbf{y}$
  - $\lambda e^{\lambda t}\mathbf{x} = \mathbf{A}e^{\lambda t}\mathbf{x}$
  - This simplifies to the eigenvalue vector equation: $\mathbf{A}\mathbf{x} = \lambda\mathbf{x}$

- Solutions of multivariable differential equations (the bread-and-butter in linear systems) correspond to solutions of linear algebraic eigenvalue equations!

Eigen Decomposition

- Every *square matrix* A, with *distinct* eigenvalues has an eigen decomposition:
  - $\mathbf{A = S\Lambda S^{-1}}$
  - … S is a matrix of eigenvectors and
  - … Λ is a diagonal matrix of <u>distinct</u> eigenvalues $\Lambda = \text{diag}(\lambda_1, \ldots \lambda_N)$
- Follows from definition of eigenvector/eigenvalue:
  - $A\mathbf{x} = \lambda\mathbf{x}$
  - Collect all these N eigenvectors into a matrix (S):
    - $AS = S\Lambda$.
    - or, if S is invertible (if e-values are distinct)…
  - => $\mathbf{A = S\Lambda S^{-1}}$

Eigen decomposition: Symmetric A

- Every **square, symmetric matrix A** can be decomposed into a *product of a* *rotation (Q), scaling (Λ) and an inverse rotation (Q^T)*
  - $\mathbf{A = Q\Lambda Q^T}$
  - Idea is similar … $\mathbf{A = S\Lambda S^{-1}}$
  - But the eigenvectors of a symmetric matrix A are orthogonal and form an orthogonal basis transformation Q.
  - For an orthogonal matrix Q, inverse is just the transpose Q^T

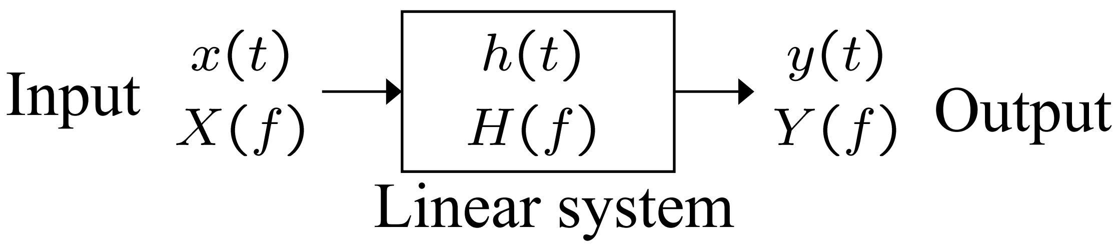
- This is why we love *symmetric (or hermitian) matrices*: they admit nice decomposition
  - We love positive definite matrices even more: they are symmetric and all have all eigenvalues strictly positive.
  - Many linear systems are equivalent to symmetric/hermitian or positive definite transformations.

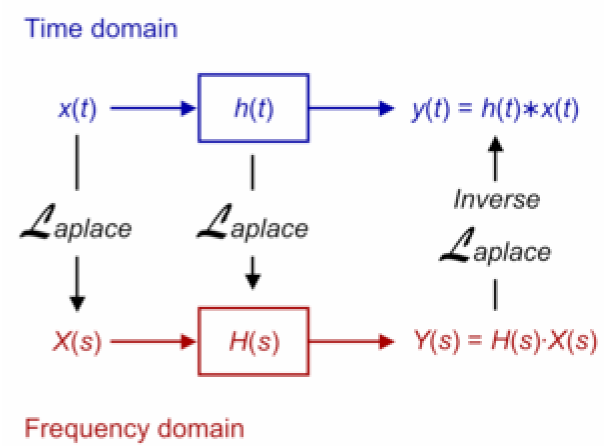
Fourier Methods ≡ Eigen Decomposition!

- Applying transform techniques is just eigen decomposition!
- Discrete/Finite case (DFT/FFT):
  - Circulant matrix **C** is like *convolution*. Rows are circularly shifted versions of the first row
  - $\mathbf{C} = \mathbf{F \Lambda F^*}$ where F is the (complex) fourier matrix, which happens to be both unitary and symmetric, and multiplication w/ F is rapid using the FFT.
  - Applying F = DFT, i.e. transform to frequency domain, i.e. "rotate" the basis to view C in the frequency basis.
  - Applying $\mathbf{\Lambda}$ is like applying the complex gains/phase changes to each frequency component (basis vector)
  - Applying $\mathbf{F^*}$ inverts back to the time-domain. (IDFT or IFFT)

Fourier /Eigen Decomposition (Continued)

- Continuous case:
  - Any function f(t) can be viewed as a integral (sum) of scaled, time-shifted impulses $\int c(\tau)\delta(t+\tau)\, d\tau$
  - h(t) is the response the system gives to an impulse ("impulse response").
  - Function's response is the convolution of the function f(t) w/ impulse response h(t): for linear time-invariant systems (LTI): f(t)*h(t)
  - Convolution is messy in the time-domain, but becomes a multiplication in the frequency domain: F(s)H(s)





Fourier /Eigen Decomposition (Continued)

- Transforming an impulse response h(t) to frequency domain gives H(s), the characteristic frequency response. This is a generalization of multiplying by a fourier matrix F
  - H(s) captures the eigen values (i.e scaling) corresponding to each frequency component s.
  - Doing convolution now becomes a matter of multiplying eigenvalues for each frequency component;
    - and then transform back (i.e. like multiplying w/ IDFT matrix F*)

- The eigenvectors are the orthogonal harmonics, i.e. phasors e^{ikx}
  - Every harmonic e^{ikx} is an eigen function of every derivative and every finite difference, which are linear operators.
  - Since dynamic systems can be written as differential/difference equations, eigen transform methods convert them into simple polynomial equations!

Applications in Random Vectors/Processes

- Covariance matrix K for random vectors X:
  - Generalization of variance, K_{ij} is the "co-variance" between components x_i and x_j
  - $K = E[(\mathbf{X} - \mu)(\mathbf{X} - \mu)^T]$
  - $K_{ij} = K_{ji:}$ => K is a real, symmetric matrix, with orthogonal eigenvectors!
  - K is positive semi-definite. When K is full-rank, it is positive definite.
- "<u>White</u>" => no off-diagonal correlations
  - K is diagonal, and has the same variance in each element of the diagonal
  - Eg: "Additive White Gaussian Noise" (AWGN)
  - Whitening filter: eigen decomposition of K + normalization of each eigenvalue to 1!
- (Auto)Correlation matrix $R = E[\mathbf{XX}^T]$
- R.vectors X, Y "*<u>uncorrelated</u>*" => $E[\mathbf{XY}^T] = 0$. "*<u>orthogonal</u>*"

Gaussian Random Vectors

- Linear transformations of the standard gaussian vector: $\mathbf{x} = \mathbf{A}\mathbf{w} + \boldsymbol{\mu}.$

$$\mathbf{c}^t\mathbf{x} \sim \mathcal{N}\left(\mathbf{c}^t\boldsymbol{\mu}, \mathbf{c}^t\mathbf{A}\mathbf{A}^t\mathbf{c}\right);$$

Singular Value Decomposition (SVD)

Like the eigen-decomposition, but for **ANY matrix**!
(*even rectangular, and even if not full rank*)!

$$\underset{m \times n}{\begin{pmatrix} & & \\ & A & \\ & & \end{pmatrix}} = \underset{m \times \rho}{\begin{pmatrix} & & \\ & U & \\ & & \end{pmatrix}} \cdot \underset{\rho \times \rho}{\begin{pmatrix} & 0 \\ \Sigma & \\ 0 & \end{pmatrix}} \cdot \underset{\rho \times n}{\begin{pmatrix} & V & \end{pmatrix}}^T$$

ρ: rank of A

U (V): orthogonal matrix containing the left (right) singular vectors of A.

S: diagonal matrix containing the singular values of A.

$\sigma_1, \sigma_2, \ldots, \sigma_\rho$: the entries of Σ.

Singular values of **A** (i.e. σ_i) are related (see next slide) to the eigenvalues of the square/symmetric matrices $\mathbf{A^T A}$ and $\mathbf{A A^T}$

Singular Value Decomposition

For an $m \times n$ matrix $\mathbf{A}$ of rank r there exists a factorization (Singular Value Decomposition = **SVD**) as follows:

$$A = \underbrace{U}_{m\times m}\underbrace{\Sigma}_{m\times n}\underbrace{V^T}_{V \text{ is } n\times n}$$

The columns of ***U*** are orthogonal eigenvectors of $\boldsymbol{AA^T}$.

The columns of ***V*** are orthogonal eigenvectors of $\boldsymbol{A^TA}$.

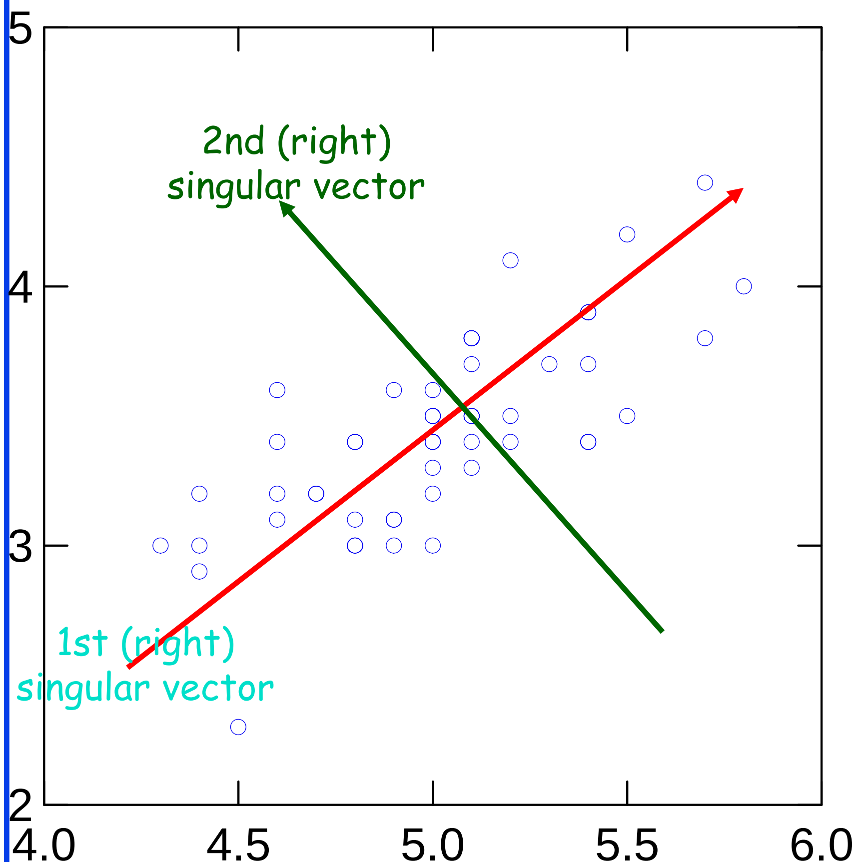
Eigenvalues $\lambda_1 \ldots \lambda_r$ of $\boldsymbol{AA^T}$ are the eigenvalues of $\boldsymbol{A^TA}$.

$$\sigma_i = \sqrt{\lambda_i}$$

$$\Sigma = diag(\sigma_1 \ldots \sigma_r)$$

Singular values.

SVD, intuition



Let the blue circles represent m data points in a 2-D Euclidean space.

Then, the SVD of the m-by-2 matrix of the data will return …

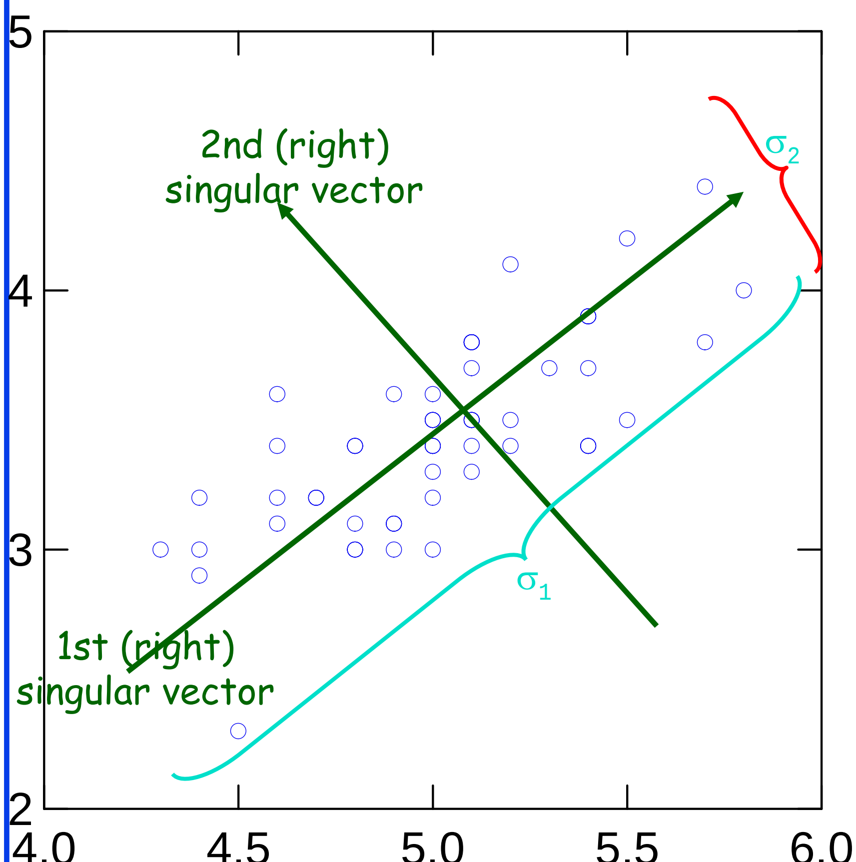
<u>1st (right) singular vector:</u>

direction of maximal variance,

<u>2nd (right) singular vector:</u>

direction of maximal variance, after removing the projection of the data along the first singular vector.

Singular Values



σ_1: measures how much of the data variance is explained by the first singular vector.

σ_2: measures how much of the data variance is explained by the second singular vector.

SVD for MIMO Channels

MIMO (vector) channel: $\boxed{\mathbf{y} = \mathbf{H}\mathbf{x} + \mathbf{w},}$

<u>SVD:</u> $\mathbf{H} = \mathbf{U}\Lambda\mathbf{V}^*$

$$\mathbf{H} = \sum_{i=1}^{n_{\min}} \lambda_i \mathbf{u}_i \mathbf{v}_i^*,$$

i.e., the sum of rank-one matrices $\lambda_i \mathbf{u}_i \mathbf{v}_i^*$'s.

Rank of H is the number of non-zero singular values

$$\mathbf{H}\mathbf{H}^* = \mathbf{U}\Lambda\Lambda^t\mathbf{U}^*$$

$$\mathbf{H}^*\mathbf{H} = \mathbf{V}\Lambda^t\Lambda\mathbf{V}^*$$

<u>Change of variables:</u>

$$\begin{aligned} \tilde{\mathbf{x}} &:= \mathbf{V}^*\mathbf{x}, \\ \tilde{\mathbf{y}} &= \mathbf{U}^*\mathbf{y}, \\ \tilde{\mathbf{w}} &= \mathbf{U}^*\mathbf{w}, \end{aligned}$$

=>

Transformed MIMO channel: Diagonalized!

$$\boxed{\tilde{\mathbf{y}} = \Lambda\tilde{\mathbf{x}} + \tilde{\mathbf{w}},}$$

SVD & MIMO continued

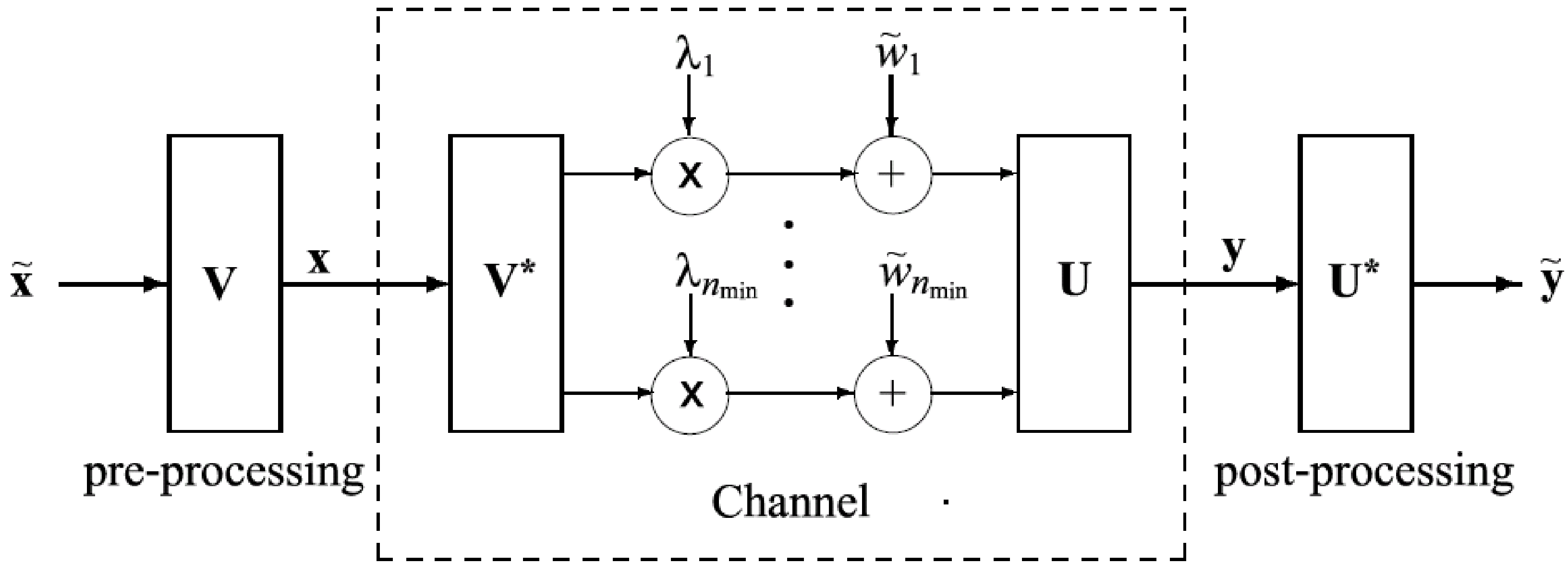


Figure 7.1: Converting the MIMO channel into a parallel channel through the SVD.

- Represent input in terms of a coordinate system defined by the columns of V (**V*x**)
- Represent output in terms of a coordinate system defined by the columns of U (**U*y**)
- Then the input-output relationship is very simple (diagonal, i.e. scaling by singular values)
- Once you have "*parallel channels*" you gain additional degrees of freedom: aka "*spatial multiplexing*"

SVD example (homework)

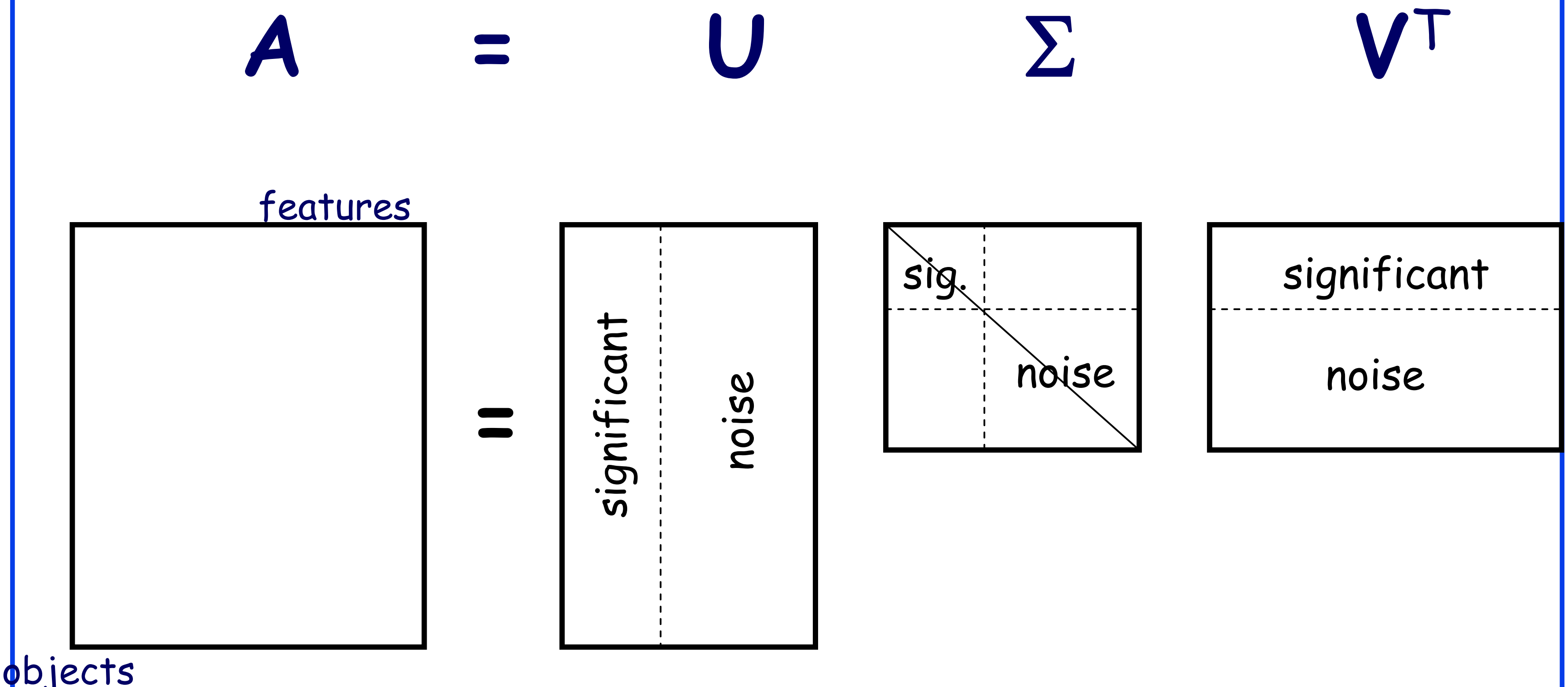
Let $A = \begin{bmatrix} 1 & -1 \\ 0 & 1 \\ 1 & 0 \end{bmatrix}$

Thus m=3, n=2. Its SVD is

$$\begin{bmatrix} 0 & 2/\sqrt{6} & 1/\sqrt{3} \\ 1/\sqrt{2} & -1/\sqrt{6} & 1/\sqrt{3} \\ 1/\sqrt{2} & 1/\sqrt{6} & -1/\sqrt{3} \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & \sqrt{3} \\ 0 & 0 \end{bmatrix} \begin{bmatrix} 1/\sqrt{2} & 1/\sqrt{2} \\ 1/\sqrt{2} & -1/\sqrt{2} \end{bmatrix}$$

Note: the singular values arranged in decreasing order.

Aside: Singular Value Decomposition, cont'd

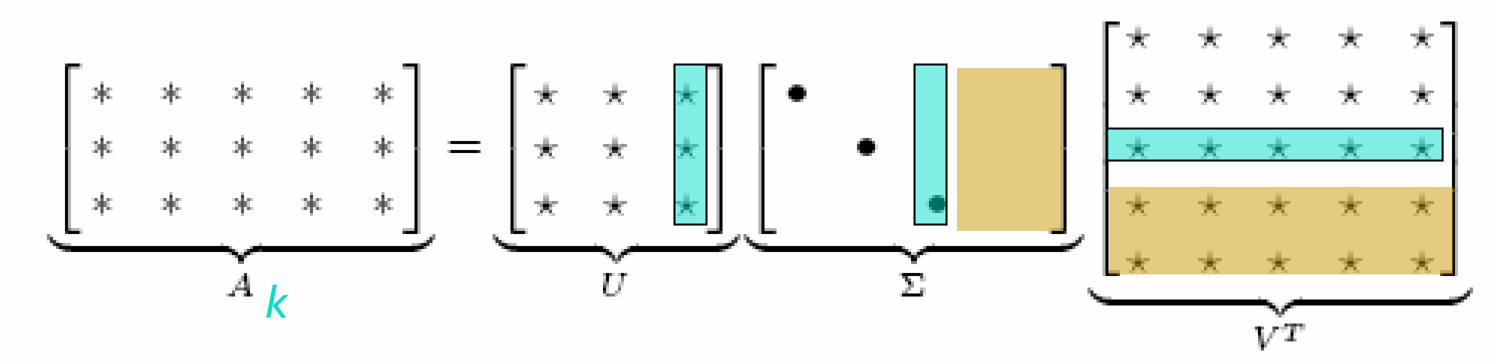


Can be used for noise rejection (compression):
aka low-rank approximation

Aside: Low-rank Approximation w/ SVD

$$A_k = U\,\mathrm{diag}(\sigma_1, \ldots, \sigma_k, \underbrace{0, \ldots, 0}_{}) V^T$$

set smallest r-k
singular values to zero



$$A_k = \sum_{i=1}^{k} \sigma_i u_i v_i^T$$

column notation: sum of rank 1 matrices

For more details

- Prof. Gilbert Strang's course videos:
- http://ocw.mit.edu/OcwWeb/Mathematics/18-06Spring-2005/VideoLectures/index.htm

- Esp. the lectures on eigenvalues/eigenvectors, singular value decomposition & applications of both. (second half of course)

- Online Linear Algebra Tutorials:
- http://tutorial.math.lamar.edu/AllBrowsers/2318/2318.asp