Advanced DSP

Lectures by
Udayan Kanade

Aditya Thakur	Amit Hadke
Kanika Nema	Prateek Saxena
Ravi Ramaseshan	Sameer Dhavale
Soam Vasani	Umaima Khilawala
Tanmay Dharmadhikari	

Advanced Digital Signal Processing: **Lecture 1**

Lecture date: **9-1-2004**

Lecturer: **Udayan Kanade** . email: udayan@codito.com

ADSP Mailing List: adsp@codito.com

ADSP web site: www.codito.com/udayan/~adsp/

Acknowledgements: **Aditya Thakur**

Convoluted Ants:

We first solved the problem of "communist ants" : given 2 ant hills, each having some amount of sugar, the 'richer' ant hill gives 10% (arbitary communist rule) of the difference between the sugar levels to the other ant hill.

If we assume that the 2 hills can communicate and let each other know of their resp. sugar levels, the the solution is obvious: the hill with more sugar calculates the difference using the 'arithmetic ants' and sending it to the other hills.

Assuming the ant hills can't communicate (because say, the hills are a great distance from each other) or the 'arithmetic ants' have gone on vacation (hence the difference cannot be calculated), brings us to the more interesting solution (suggested by the 'dsp ants') . If each ant hill sends 10% of its sugar to the other hill, then the problem is solved.

Example:	A	B
Original:	20	30
Amount sent:	2	3
Final Amount:	21	29

Yippeeee !!

This can be generalized to 'n' ant hills, each ant hill have 'm' neighbour hills and a corresponding rule to distribute sugar to each of its neighbours.

We immediately notice that this method is "scatter convolution". Wow! , a true natural wonder (recent reports tell us that the 'dsp ants' were handsomely rewarded).

Taking a hint from their communist friends, the capitalist ants also devised a similar solution for their problem: each ant hill is supposed to steal from its neighbour hill. Thus, each hill goes to its neighbour hill and steals a certain amount (governed by the capitalist rule) and collects and adds all the loot at the end of the day.

From this analogy it is easy (and fun) to see that the scatter and gather are equivalent if we 'flip' the kernel (extra brownie points for anyone who finds the origin of the word

'kernel') i.e. suppose the communist ants give 10% to left hill and 20% to right hill and keep the rest, this is equivalent to the capitalist ants stealing 10% from their right hill and 20% from their left hill.

Properties:

Next we reviewed some of the properties of convolution using the ant analogy:

1) Commutative: If we swap the communist rule with the original sugar amounts and perform the same scatter operation, then the result obtained will be the same. This point brings us to the reason why the scatter approach is used to define convolution: the scatter convolution is commutative as opposed to the gather.

   A non-rigorous proof for the latter follows:
   Assuming that scatter operation is commutative and represented by *,
   We prove that the gather operation (represented by #) is not.

   A#B = A * flip(B)
   = flip(B) * A
   = flip(B) # flip(A)
   != B#A
   QED
2) Associative: Suppose we use rule B on the first day and then apply rule C to the result obtained , the final result obtained at the end of the 2^{nd} day is the same as using the dsp ants to apply the rule B to rule C on the 1^{st} day and then applying this combined rule to the original amount on the 2^{nd} day.
   i.e. (A*B)*C = A*(B*C) where A is the original amount.

   Combining the first 2 rules allows us to write the expression A*B*C or B*C*A (or any such permutation) and they all represent the same result. This a prime example of mathematical symbols hiding the beauty of the underling operation.

3) Identity: The identity in this case is each ant hill keeping all the sugar for itself.

4) Linearity: k(A*B) = kA*B and A*(b+c) = A*b + A*c

Strength Matters!

We define the strength operator on a vector as
$S(A) = \Sigma a_i$

Things you need to know about strength:

1) It's a field.
2) It's a scalar function (maps a vector to a scalar).
3) It's linear.
4) It isn't shift invariant. This is because for an operation to be shift invariant both the input and the output should have the same number of components. In the case of the strength operator, since the output is a scalar, it's not shift invariant.
5) Strength Theorem:
   $S(f*x) = S(f)\,S(x)$

   This shows us that if the kernel adds to 1, then the convolution operator conserves strength.
   But if this not true, say the kernel strength is less than 1 (to compensate for loss during the transportation of the sugar due to hungry ants), then after the scatter operation the strength of the resultant sugar will be less than that of the original , and the strength theorem tells us how much was the loss.

When convolution goes bad: see deconvolution (seedy convolution!) :

Deconvolution is when, given the signal and the output, we have find out the response of the system or equivalently given the output and the response of the system, we have to figure out the input.

Real-world applications of deconvolution:

Examples when you are given the output and the response:

1) given a 'blurred' image, you obtain the blurring kernel, and using deconvolution, get the original un-blurred image. Such a technique is commonly used to correct the defects (due to the curvature) in telescope lens.

Examples when you are given the output and the input:

2) Pre and post equalization is used in the telephone system to compensate for the distortion caused by the noise in the channel (found out by using deconvolution) .
3) To compensate for the response in large theaters the sound output is modified.
4) Deconvolution can be used to get better quality pirated VCDs !

Example showing procedure for deconvolution:

Output : 5 10 20 100 50 20 10
Kernel: 0.1 0.8 0.1

Assuming that the system is causal,

$5 = 0*0.1 + 0*0.8 + 0.1* x = 0.1*x$
Thus, $x0 = 50$.

$10 = 0*0.1 + 0.8*x0 + 0.1*x1 = 0.8*50 + 0.1*x1$
$x1 = 10 – 0.1*x1 - 0.8*x0 = -300$

Generalizing,

$x = y\text{-}f\ D(x)$
where y is the current output,
f is the kernel function
D(x) is delay x

Taking Z-transform ,
$X = Y – z^{-1} F X$
$X = Y / (1 + z^{-1} F)$

This equation shows us that deconvolution follows the feedback network.
Further given a FIR convolution system the corr. deconvolution is a simple IIR system (all pole system) with the zeroes getting mapped to the poles.
Thus, though a FIR system is stable, its inverse IIR system can be unstable if the pole lies outside the unit circle.

Types of Filters

We saw three types of filters, viz. MA, AR, and ARMA. These do not include the entire class of filters but cover an important class of filters and are very useful.

For linear systems where output y(n) is related to the input x(n) by the difference equation:

$$y(n)+\sum_{k=1}^{p} g_k\, y(n-k)=\sum_{k=0}^{q} \mathrm{f}_k\, x(n-k)$$

We considered three cases:

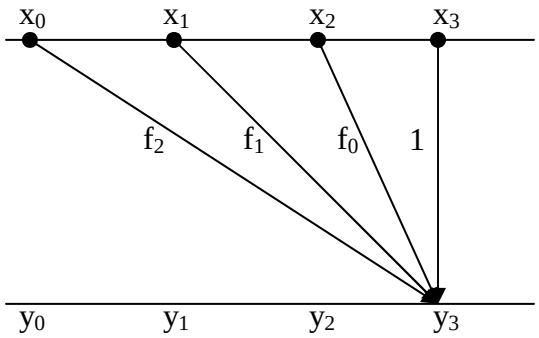
1. **Moving Average Filters (MA)**

   In this case, the linear filter,

   is an all-zero filter and the difference equation for their input-output relationship is

   $$y(k) = x_k + f_0 x_{k-1} + f_1 x_{k-2} + \ldots$$

   Note: The strength of these filters may not be 1

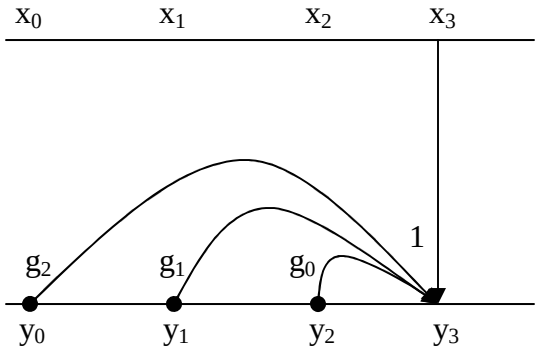


2. **Autoregressive Filters (AR)**

   In this case, the linear filter,

   is an all-zero filter and the difference equation for their input-output relationship is

   $$y(k) = x_k + g_0 y_{k-1} + g_1 y_{k-2} + \ldots$$



AR Filters are linear shift invariant systems

Refer to previous AR filter figure.

Using the figure we can calculate the expressions for y_0, y_1…

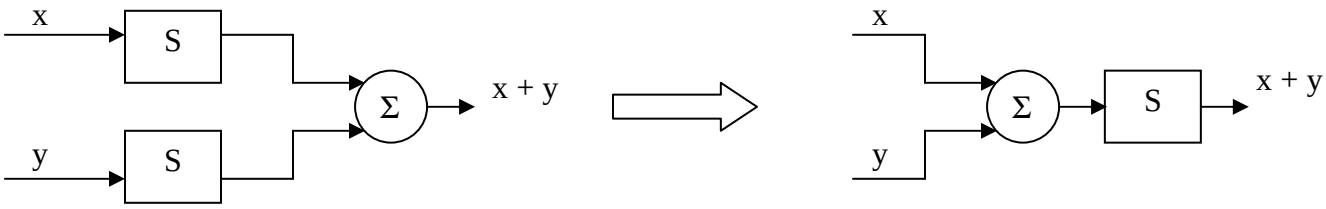
$$y_0 = x_0$$
$$y_1 = x_1 + g_0 x_0$$
$$y_2 = x_2 + g_0 x_1 + (g_0^{\,2} + g_1) x_0$$
$$y_3 = x_3 + g_0 x_2 + (g_0^{\,2} + g_1) x_1 + (g_0^{\,3} + 2 g_0 g_1 + g_2) x_0$$

This can be seen as

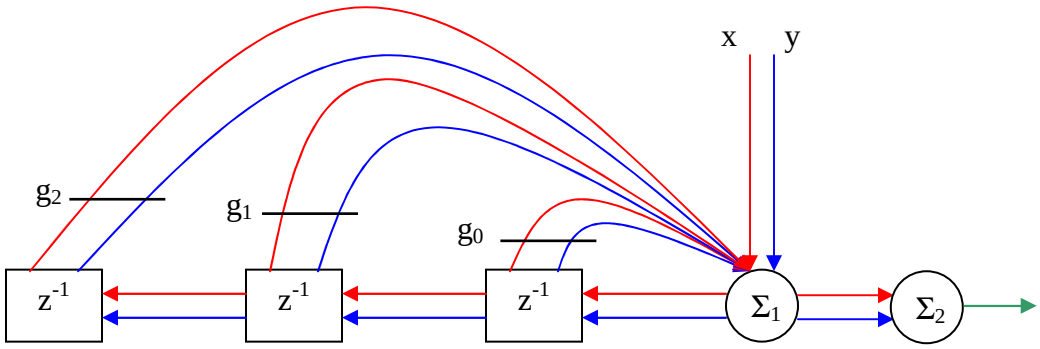
$$Y = X * F$$
$$F = \begin{bmatrix} 1 & g_0 & (g_0^{\,2} + g_1) & (g_0^{\,3} + 2 g_0 g_1 + g_2) \ldots \end{bmatrix}$$

By looking at the difference equation of AR filters we observe that y_k is described in terms of k and does not refer to any particular value of y or x. Hence we can say that AR filters are shift invariant

To show that AR is linear, we have to show that

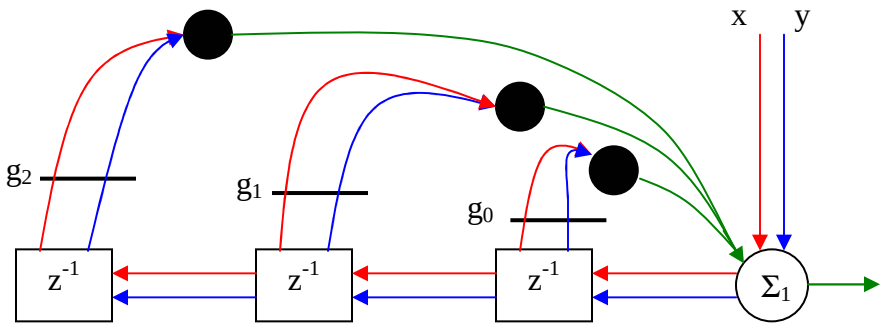


Let us superimpose the blue and red AR systems as in the figure below and finally add up the two results in Σ_2. We can look at the AR system as containing the summation element Σ_1, the delay elements z^{-1} and the multiplying elements g_k.

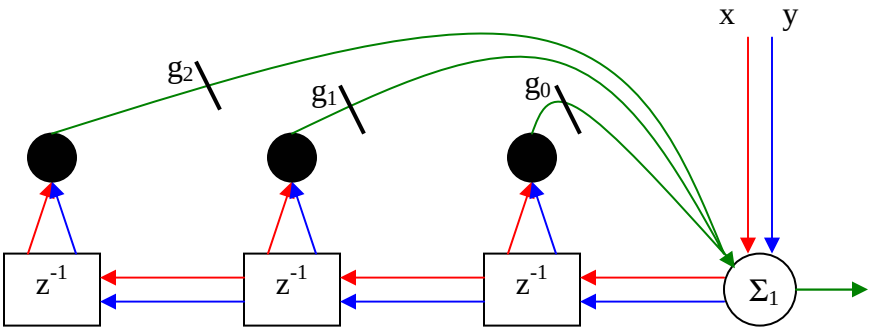


Let us denote, for convenience, Σ_2 as a black filled circle.

Add (Σ_2) before the addition (Σ_1) or after, it doesn't make a difference…

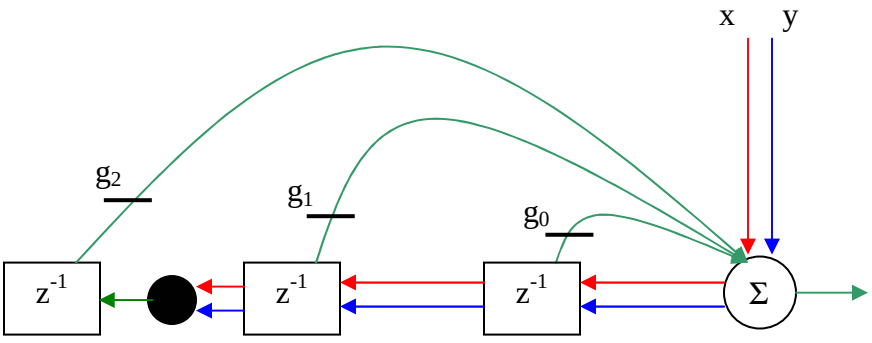


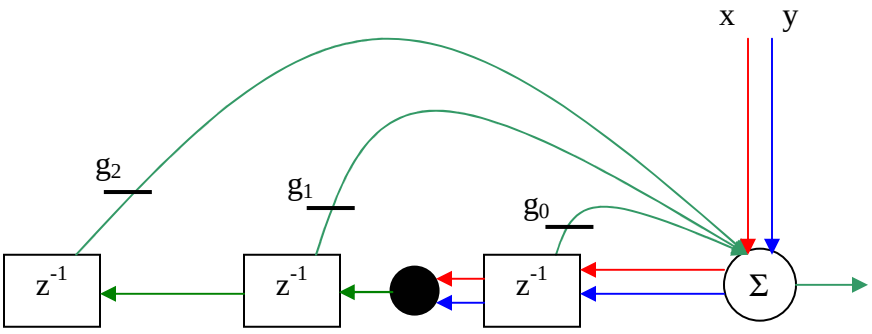
Add (Σ_2) before the multiplication (g_k) or after, it doesn't make a difference…

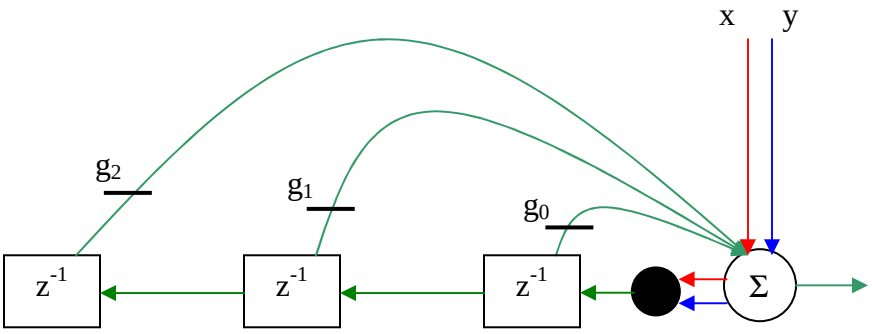


Now, for each of the three Σ_2 we have the following:
Next three figures for the Σ_2 before g_2
Among the next three, the 2nd and 3rd for the Σ_2 before g_1
Among the next three, the 3rd for the Σ_2 before g_0
At each step we combine the Σ_2's between the delay elements

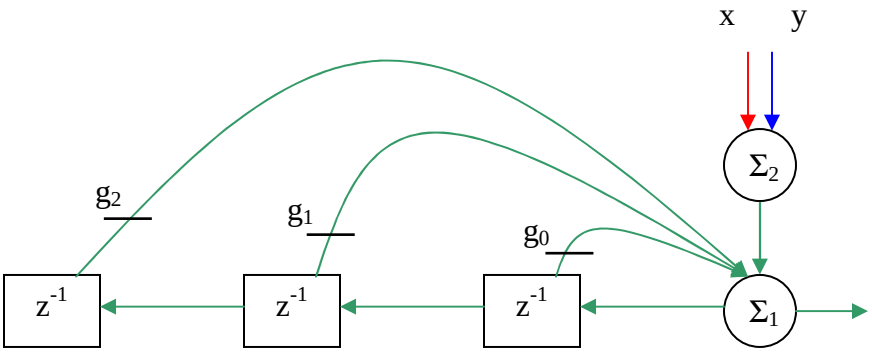
Add (Σ_2) before the delay (z^{-1}) or after, it doesn't make a difference…







Add (Σ_1) before the addition (Σ_2) or after, it doesn't make a difference…



Another way of showing AR filters as linear is by induction.
$y_0 = x_0$ forms the basis of the induction hypothesis
Let us assume y_{k-1}, y_{k-2},… to be linear
y_k is defined in terms of x_k and y_{k-1}, y_{k-2},… in the AR filter difference equation.
Since x_k is just added and the y terms are multiplied by constants, we can say that y_k is also linear.
Hence AR filters are linear systems.

3. Autoregressive, Moving Average Filters (ARMA)

In this case, the linear filter,

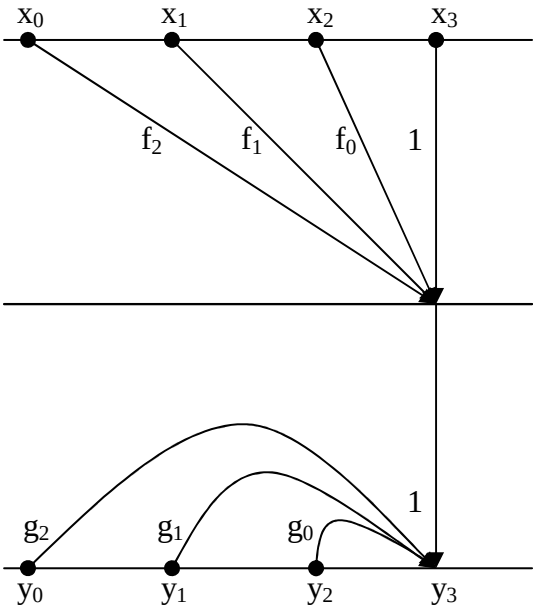
$$\frac{H(z)}{G(z)} = H(z).\frac{1}{G(z)}$$

is an pole-zero filter which has both finite poles and zeroes.

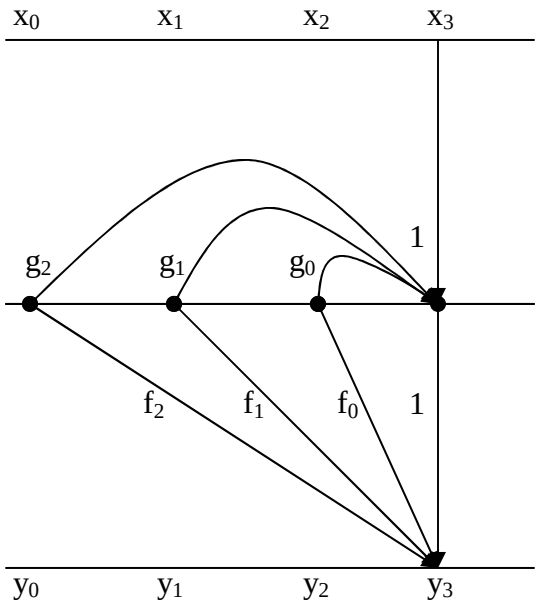
ARMA filters are basically cascaded MA and AR filters.

There are 2 types[i] of ARMA filters:

Type 1 (MA → AR)



Type 2 (AR → MA)



Type 2 filters have an advantage over Type 1 in that only one set of latches is required for their implementation.

When constructing an ARMA filter, the AR filter may be unstable. As long as the poles of the AR filter match the zeros of the MA filter, the resulting ARMA filter is stable. However, sometimes we may not be able to match the poles and the zeros perfectly. Some of the reasons are:

1. On computers, due to precision / truncation errors
2. Incapability of specifying the physical media (plant errors)

Given two vectors y and x, if we wanted to fit them together we would scale one of them by a scalar a. Using mathematical symbols we would write it as:

$$f(a) = \|y - ax\|_2^2$$
$$f(a) = (y - ax)^T (y - ax)$$
$$f(a) = y^T y - 2ay^T x + a^2 x^T x$$

This is a minimization problem where a has to be varied to minimize *f*

$$\frac{df(a)}{da} = 2ax^T x - 2y^T x = 0$$
$$\therefore a = \frac{y^T x}{x^T x}$$

f(a) is a parabola which can have a minimum only if the coefficient of the second degree term is greater than zero viz. $x^T x$. This is true for an upward facing parabola as this term dominates every other terms.

For a parabola ax^2+bx+c the minima is at $-b/2a$. This is obvious because the intersections with the x-axis are at $-b/a$ and b/a and the parabola is symmetric[ii].

Deconvolution Revisited

As we have seen before an LTI system can be represented as

$$Y = AX$$

Further, deconvolution is the process of finding the kernel/input given the output and the input/kernel. Hence we can view deconvolution as matrix inversion where in we need to find *X* given *Y* and *A* by finding A^{-1} and pre-multiplying the above equation.

A generalization of the above problem can be stated as

In most practical situations, *X* does not span the space of *Y* and hence there is no exact solution to the above equation. Thus we formulate the minimization problem as:

Given *Y* and *A*, we have to get *X* such that

$$f(X) = \|Y - AX\|_2 \text{ is minimum.}$$

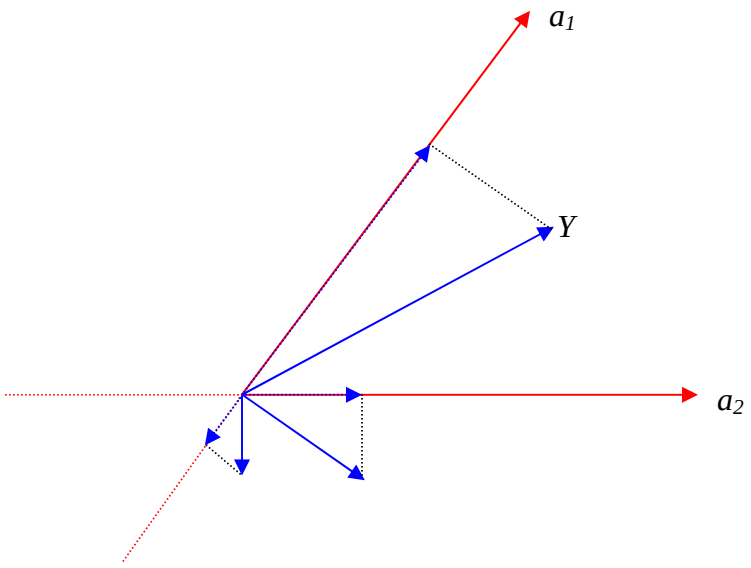
Where $\|U\|_2^2 = \sum U_i^2 = U^T U = \langle U, U \rangle$ (L_2 Norm)

This problem is very similar to the problem we encountered earlier. Y, A and X in the above equation correspond to y, x, and a in the earlier problem. There a was a scalar here X is a vector. We can see this as multiplying each multiplying each column of A (a vector) with each row of X (a scalar).

Thus it may seem that we have reduced this problem into multiple instances of the previous problem. However, this is not so because the columns of A do not form an orthogonal basis.

The problem can be seen when we consider the following figure in which Y is a vector in the plane of a_1 and a_2 (A spans Y). Let us try and adopt the method of the previous problem here. We project Y onto a_1 and project whatever is remaining onto a_2. We see that we are still left with some amount of Y and we have to repeat the same procedure again (and again…). Although this procedure converges, it takes a lot of time.

The figure below shows the first few steps in the projection and re-projection of Y along a_1 and a_2



Least Square Matrix Inversion

$$f(X) = \|Y - AX\|_2^2$$

$$f(X) = Y - \begin{bmatrix} a_1 & a_2 & \cdots & a_n \\ \vdots & \vdots & \cdots & \vdots \\ \vdots & \vdots & \cdots & \vdots \\ \vdots & \vdots & \cdots & \vdots \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ \vdots \\ \vdots \\ x_n \end{bmatrix}$$

$$f(X) = (Y - AX)^T (Y - AX)$$

$$f(X) = (Y^T - X^T A^T)(Y - AX)$$

$$f(X) = X^T (A^T A) X - (Y^T AX) - (Y^T AX)^T + Y^T Y$$

$$f(x) = X^T \underbrace{(A^T A)}_{P} X - \underbrace{(2Y^T A)}_{Q^T} X + \underbrace{Y^T Y}_{R}$$

We write the above equation as

$f(X) = X^T PX + Q^T X + R$

where $P = A^T A$
$Q = -2A^T Y$
$R = Y^T Y$

$f(X)$ is a field representing an n-dimensional paraboloid[iii].

The above equation will have a minima only if $\forall x \; X^T PX > 0$
P is the positive definite, written as $P \succ 0$

The above equation using summations:

$$f(X) = \sum_i \sum_j P_{ij} X_i X_j + \sum_i Q_i$$

We take the partial derivative with respect to x_i in order to minimize $f(X)$

$$\frac{\partial f}{\partial x_i} = \sum_{j \neq i} (P_{ij} + P_{ji}) X_j + 2P_{ii} X_i + Q_i$$

$$\frac{\partial f}{\partial x_i} = \sum_{j} (P_{ij} + P_{ji}) X_j + Q_i$$

$$\nabla f = \begin{bmatrix} \partial f / \partial x_1 \\ \partial f / \partial x_2 \\ \vdots \\ \partial f / \partial x_n \end{bmatrix} = \begin{bmatrix} \sum_j (P_{1j} + P_{j1}) X_j + Q_1 \\ \sum_j (P_{1j} + P_{j1}) X_j + Q_1 \\ \vdots \\ \sum_j (P_{nj} + P_{jn}) X_j + Q_n \end{bmatrix}$$

$$\nabla f = \begin{bmatrix} \sum_j P_{1j} X_j \\ \sum_j P_{2j} X_j \\ \vdots \\ \sum_j P_{nj} X_j \end{bmatrix} + \begin{bmatrix} \sum_j P_{j1} X_j \\ \sum_j P_{j1} X_j \\ \vdots \\ \sum_j P_{jn} X_j \end{bmatrix} + \begin{bmatrix} Q_1 \\ Q_2 \\ \vdots \\ Q_n \end{bmatrix}$$

$$\nabla f = \left(\begin{bmatrix} P_{11} & P_{12} & \cdots & P_{1n} \\ P_{21} & P_{22} & \cdots & P_{2n} \\ \vdots & \vdots & & \vdots \\ P_{n1} & P_{n2} & \cdots & P_{nn} \end{bmatrix} + \begin{bmatrix} P_{11} & P_{21} & \cdots & P_{n1} \\ P_1 & P_{22} & \cdots & P_{2n} \\ \vdots & \vdots & & \vdots \\ P_{n1} & P_{n2} & \cdots & P_{nn} \end{bmatrix} \right) \cdot \begin{bmatrix} X_1 \\ X_2 \\ \vdots \\ X_n \end{bmatrix} + \begin{bmatrix} Q_1 \\ Q_2 \\ \vdots \\ Q_n \end{bmatrix}$$

$$\nabla f = (P + P^T) X + Q$$

The minima of *f(X)* can be found by solving $\nabla f = \vec{0}$.

$$(P + P^T) X + Q = \vec{0}$$

$$\therefore (P + P^T) X = -Q$$

$$X = -(P + P^T)^{-1} Q$$

This is very similar to the solution *–b/2a* if we put Q as *b* and *P* as *a* (Assuming P is symmetric which it is in most practical cases).

Substituting for *P* and *Q*

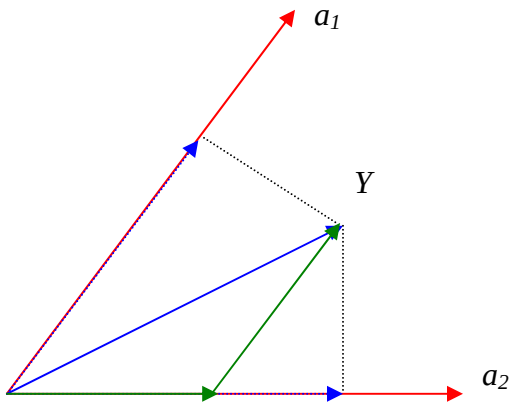
$$X = -2(A^T A + A^T A)^{-1} A^T Y$$

$$X = (A^T A)^{-1} A^T Y$$

$X = A^{\dagger} Y$

$A^{\dagger} = (A^T A)^{-1}$ is called the pseudo inverse[iv].

In the above equation A^T is the projection matrix. It projects Y onto the basis vectors of A. Finally $(A^TA)^{-1}$ converts the projections into linear combinands[v].



In the above figure, the red vectors are the basis, the blue vectors are the projections of Y and the green vectors are the linear combinands.

Let us review what we have done. Given the vectors *Y* and transformation we had to find *X* such that *Y* - *AX* was minimum. Usually Y has far more components than X. We have to tweak only a few parameters of the input vector X. For example,

$$\underbrace{\begin{bmatrix}\vdots\\ \vdots\\ \vdots\\ \vdots\end{bmatrix}_{(4x1)}}_{Y} = \underbrace{\begin{bmatrix}\vdots & \vdots\\ \vdots & \vdots\\ \vdots & \vdots\\ \vdots & \vdots\end{bmatrix}_{(4x2)}}_{A} \underbrace{\begin{bmatrix}\vdots\\ \vdots\end{bmatrix}_{(2x1)}}_{X}$$

Here *X* is a 2-dimensional vector and *Y* is a 4-dimensional vector. *A* converts *X* from its coordinate system into the coordinate system of *Y*, thus making *Y* and *AX* comparable. AX is the linear combination of the bases of *A* weighted by the components of *X*. *AX* still lies in its plane of *X* while *Y* is outside that plane. Hence we cannot directly equate them. Instead we drop perpendiculars to the space of *A* and equate them. Dropping perpendiculars is done by pre-multiplying them with A^T

Hence, $A^T AX = A^T Y$

And $X = (A^TA)^{-1} A^T Y$

Note that this looks very similar to $a = \frac{y^T x}{x^T x}$ which we had found earlier.

Udayan's comments and corrections –

[i] Because of commutativity, type I and type II systems are equivalent. Thus, these are not two types of ARMA filters. Rather they are two ways of implementing them.

[ii] The parabola *we* are talking about is completely positive, and has no roots. But the $-b/2a$ formula holds for complex-root parabolas just like it does for real root parabolas.

[iii] "Quadratic surface"

[iv] $A^{\dagger} = (A^{T}A)^{-1}A^{T}$

[v] "Combinants"!!

Applying the Least-Squares Method

28th Jan 2004

Summary of the lecture

1. We proved that the inverse of a linear system, when it exists, is also linear

2. Overview of system identification

3. We saw how to apply the least-squares method to system identification when we're modelling the system as MA.

Inverse of a linear system

Any linear system can be represented by a matrix (or by infinite matrix-shaped things). That's because a linear system is completely specified by its impulse responses at each time; and the columns of the matrix are these impulse responses.

So, a linear system P is the matrix multiplication

$$y = Px$$

where x is the input and y the output.

When does the function P^{-1} exist ? For any function, the inverse exists when it is one-one and onto. In the language of matrices and spaces and stuff, what does this mean ?

One-one-ness :

A necessary condition is that the output should have atleast as many dimensions as the input. That's because it is impossible to **linearly** map (i.e. "project") a higher dimensional space into a smaller dimensional one without becoming a many-to-one function.

It turns out that this is also a sufficient condition. There's a theorem that if a linear system is one-one then it will have same output dimension as input dimension. You can't become many-to-one and stay linear without losing a dimension.

Onto-ness :

A function is onto when its range equals its codomain.

The dimension of the output space is the rank of the matrix P^T.

To see why, recall how we find the rank of a matrix: by doing row transformations and trying to make some row zero. The number of non-zero rows remaining is the rank.

This means that we are trying to express some row as the linear combination of other rows. When this matrix is P^T, the rows are the bases of P. So we're trying to express *one basis vector as a linear combination of others.* Such a vector doesn't need to be lying around in the basis, because the basis would span the same space without it. So we remove this extra vector. Removing all such extra vectors is exactly what we do while finding the rank. That's why the "true" dimension of the output vector space is the rank of P^T.

For the matrix to be an onto function, it must be a *full rank* matrix, meaning that none of the colums should be extra vectors. This corresponds to the range=codomain condition.

Now we'll prove that when the inverse does exist, it is linear. That is, we'll show that P^{-1} is a linear transformation.

Applying the function P^{-1} to a vector gets back the linear combinants[1]

First we show that $P^{-1}(x+y) = P^{-1}x + P^{-1}y$:

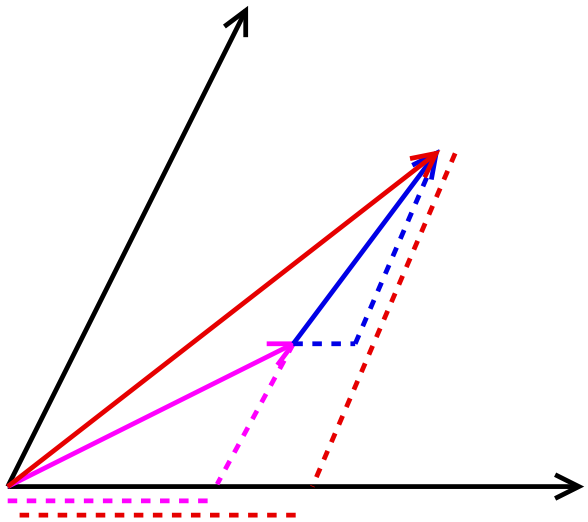


Figure 1: Linearity : Superposition

The blue solid line is x; the pink solid line is y; The red solid line is $(x+y)$. The dotted lines are what P^{-1} gives us. See how the blue and pink dotted lines add up to the red dotted line.

Then, we prove $P^{-1}(ax) = aP^{-1}x$:

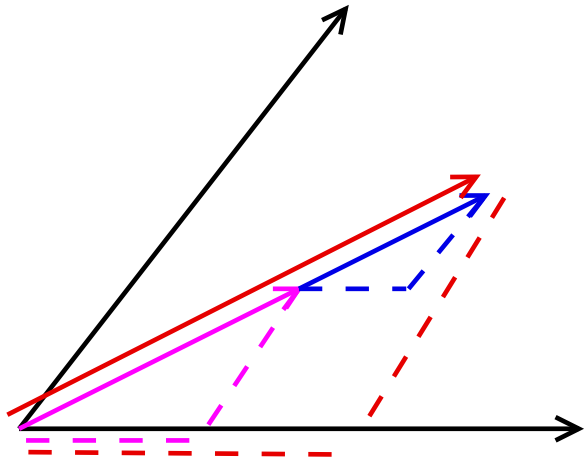


Figure 2: Linearity : Scaling

The pink solid line is x; the blue solid line is ax for some a; The red solid line is $(a+1)x$. Again, the blue and pink dotted lines add up to the red dotted line, i.e. the components of $(a+1)x$ are $(a+1)$ times the components of x.

[1] In 12th we called these the *scalar components* of the vector. The scalar components multiplied by the corresponding basis form the *vector components*. We'll call both these things "linear combinants".

Algebraically, if $Px = y$:

Superposition:

$$P(x_1 + x_2) = Px_1 + Px_2 = y_1 + y_2$$

Therefore,

$$P^{-1}(y_1 + y_2) = P^{-1}(P(x_1 + x_2)) = x_1 + x_2 = P^{-1}y_1 + P^{-1}y_2$$

Scaling:

$$P(ax) = aPx = ay$$

Therefore

$$P^{-1}ay = P(P^{-1}(ax)) = ax = aP^{-1}y$$

Overview of System Identification

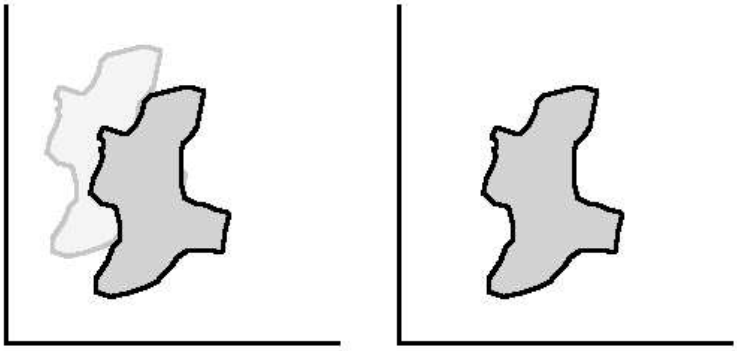


Figure 3: Output and Input

If you have a telescope which gives you an image like the one the left, and you know that the object it's looking at looks like the one on the right.

Then you have to find the convolution kernel which turns the image on the right into the one on the left. This kernel will be something like:

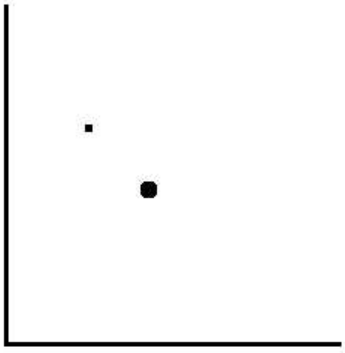


Figure 4: The filter kernel

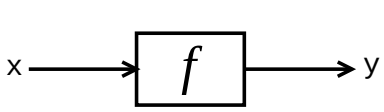


Figure 5: A system

The system identification problem is: to find the impulse response of the LSI system f given an input vector x and the resulting output vector y.

We classify this problem into two kinds:

1. Systems where we cannot control x, only measure what is already there.

2. Systems where we can give any x we want.

In the second type, what's a good x to give ? Well, the best x is an impulse because that immediately gives you the impulse response. But you can't always give an impulse, because it may be impossible in that system. For example, if x is a sound wave, you can't physically make an exact sound impulse.

Basically, any input x which has a wide range of frequencies in it (i.e. a broadband signal) is good. A square wave, a ramp, a frequency sweep, are all good inputs to give. A sine wave is not, because it has just one frequency. If you give a sine wave to the system, you'll get the response to just one frequency. It's better if you give many sine waves with different frquencies. This is called a frequency sweep.

When you cannot control x, you can measure the input being given and try to work with that. But if it is narrow band, then there's nothing you can do about it.

Any causal non-zero signal is broadband.

Applying the Least Squares Method to MA system identification

Suppose we model the system as a moving average system with q coefficients. Then, this is how we can apply the least squares method to find those coefficients given x and y:

The least squares method gives us f when we give it the equation

$$Af = y$$

In this equation, we'll make A out of the input vector x; f is the vector of dimension q that we want to find; and y is the output vector y. For example, if $q = 4$:

$$\begin{pmatrix} x_1 & 0 & 0 & 0 \\ x_2 & x_1 & 0 & 0 \\ x_3 & x_2 & x_1 & 0 \\ x_4 & x_3 & x_2 & x_1 \\ x_5 & x_4 & x_3 & x_2 \\ x_6 & x_5 & x_4 & x_3 \\ \vdots & \vdots & \vdots & \vdots \end{pmatrix} \begin{pmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ \vdots \end{pmatrix}$$

We've shown x causal in this equation, but it need not be. If it is non-causal then those zeroes in A will be replaced by x_{-1},x_{-2} etc.

When x is causal, no column in A_x can be expressed as a linear combination of the others. This means that the basis has no extra vectors.

What should the order (q) of the filter be ?

Sometimes you may be able to decide the order from other characteristics of the system. For example if you're modelling a system which disperses its input a little, and you know the time for which the dispersion can occur, then you can decide the order.

Another way to get the order is to iterate over it, each time using the least-squares method to find f, and plot the error $\|y - A_x f\|$ against q. The graph will look something like the blue line in :

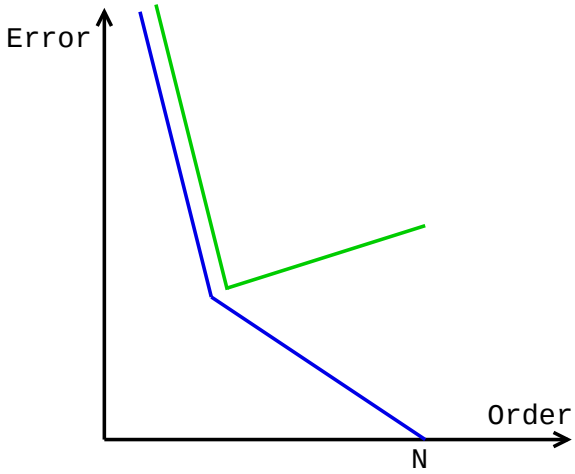


Figure 6: Deciding q

The error becomes zero when there are as many coefficients as the dimension of x, in which case least-squares is actually Gauss Elimination.

The corner in the middle of the graph is the order you should use. Increasing the order beyond this point is just putting in extra coefficients to make the filter better for this particular input.

Instead of plotting the error for the same input and output as was used in the least-squares, suppose we use a different pair of x and y to find $\|y - A_x f\|$. Then if you increase the order too much, the error increases because you're tailoring the filter to the input that was given to least-squares, not the input which was given to $\|y - A_x f\|$. So in this method the order we should use is the point where the error is minimum.

Using MA modelling for modelling an AR system:

You can use MA modelling to model an AR system, using a neat little trick:

The inverse of an AR system is an MA system. So, to model a system as AR, interchange the roles of y and x, give them to least-squares. An MA system will come out. Invert it. That's the required AR system.

Least-Squares Method Continued

-30th January 2004

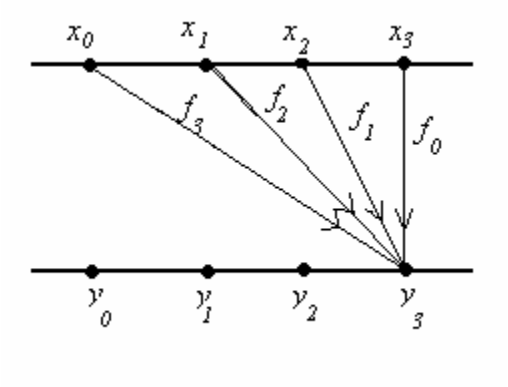
Why do we need Auto Regressive Systems?

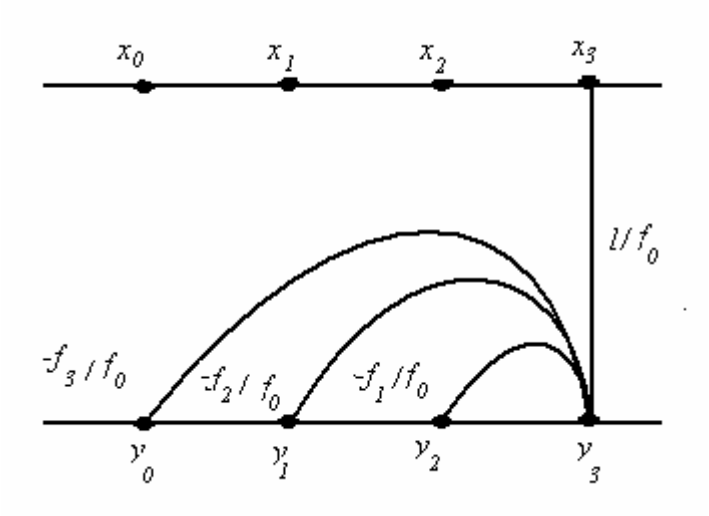
If the system plant is recursive e.g. standing wave echo.
Moving Average requires more latches
You need many coefficients for FIR
You can use it in integrator, differentiator and shock absorber

Modeling AR systems:

1. Model the inverse of the AR system as MA

We know that if the coefficients of the Moving Average system are 1, f_1,f_2 and f_3 , the coefficients of the (inverse) AR system are 1, -f_1,-f_2 and -f_3 respectively.





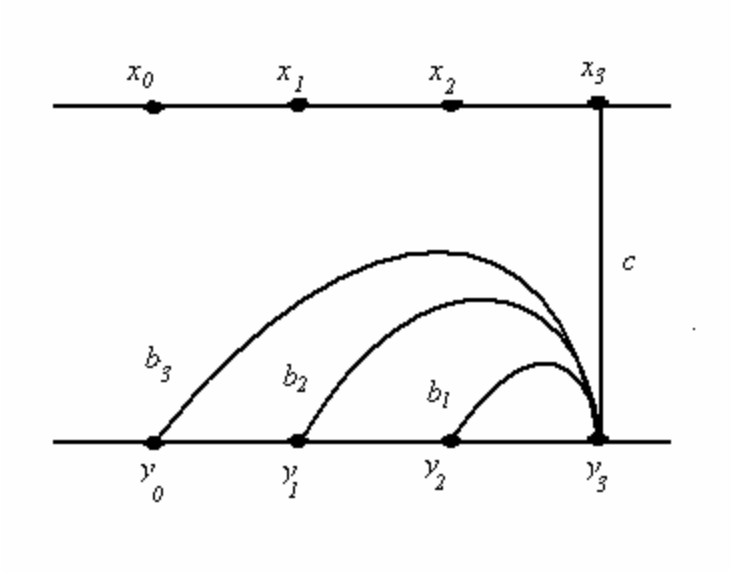
2. Direct AR modeling

We know the equation of the AR filter.

$y_0 = cx_0 + b_1y_{-1} + b_2y_{-2} + b_3y_{-3} \ldots\ldots$

Now since we start with zeroes, we can write the AR filter in matrix form as follows:

$$\begin{bmatrix} 0 & 0 & 0 & x_0 \\ y_0 & 0 & 0 & x_1 \\ y_0 & y_1 & 0 & x_2 \\ y_0 & y_1 & y_2 & x_3 \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ c \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \end{bmatrix}$$

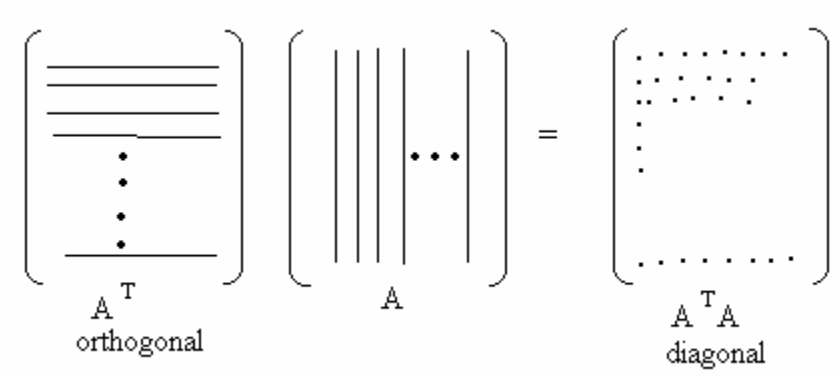


This method gives the same answer as the inverse. But the inverse method is easier.
Direct AR is good if we have c=0. (c=0 indicates that the system does not depend on any input)

Note: The matrix that we observe in the direct AR modeling is not the Toeplitz matrix, as it is not the shift in shift out x thing.

A^TA:
Some stuff about A^TA.
It is the dot product of the bases and is the skinny matrix whose size is the order of the system.

A^T has the bases on the rows and A on the columns.



If A^T is orthogonal then A^TA is diagonal. Impulse is an example of orthogonal bases.

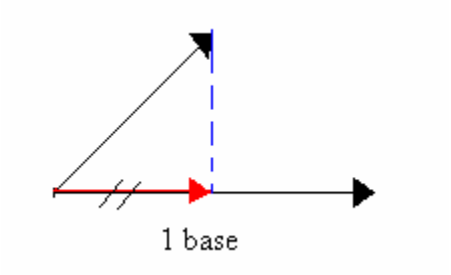
A^TA is the auto correlation of the original matrix and it is a square, symmetric matrix and the dot product of the bases.

Its dimension = no. of coefficients of the system.

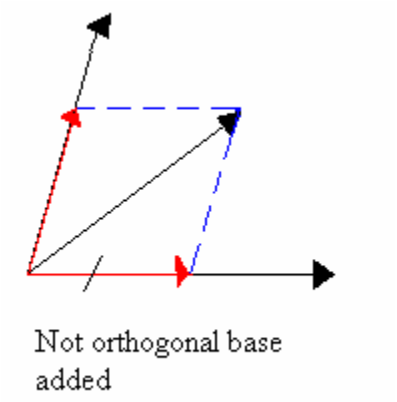
Advantage of orthogonal bases.

If the bases are not orthogonal, then adding a base changes how the previous filter is used.

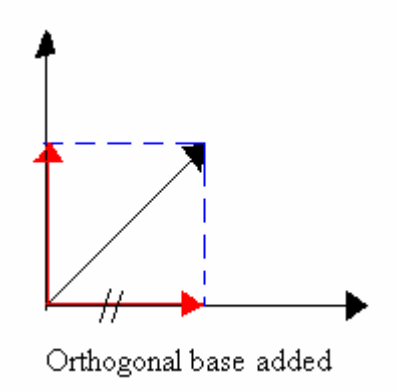
If there is only one base, so the red line shows how the filter is used



Now the new base is not orthogonal, we wont know how to add the previous filter, its not equal to the one base case.



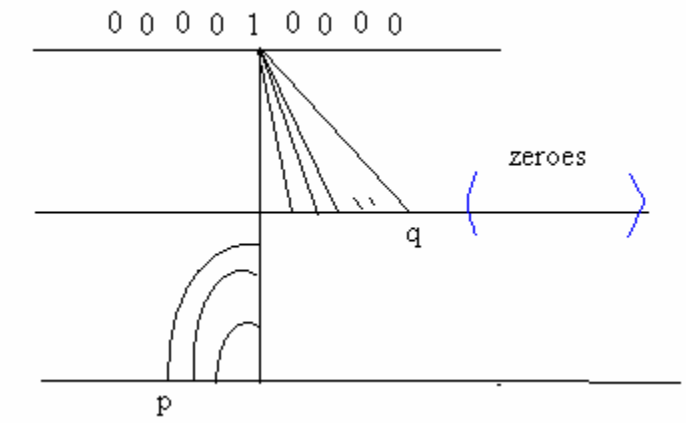
But in case of orthogonal bases there is no change in how the previous filter is used.



ARMA system identification:

There are two cases in this depending on the input.
If you can control the system input then we have three approximation techniques as follows:

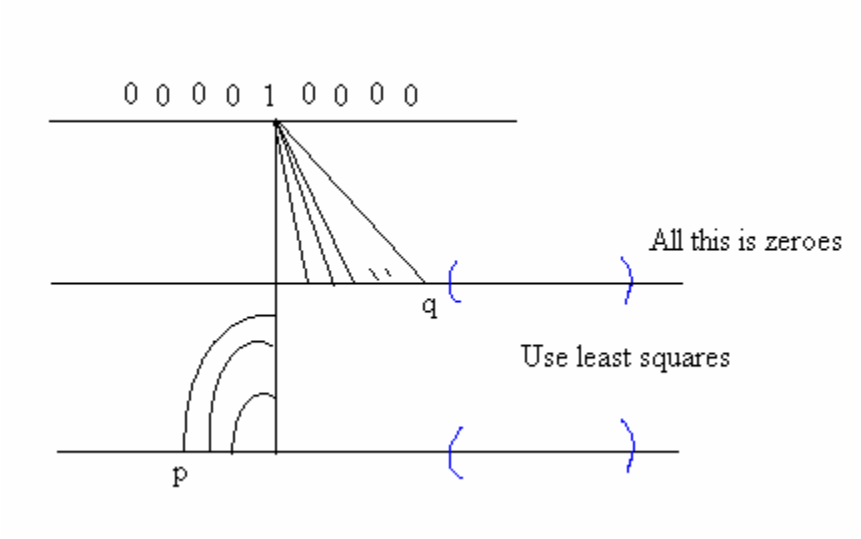
1. Pade Approximation Technique



In this technique we give an impulse as input. Assume that it has p poles and q zeroes.

1. Since the input is finite, we know that the middle line will have all zeroes after the 'q'.
2. And you know the p+q output after this middle input is given to the AR (the impulse response)
3. With this information (p+q output and q followed by 0 input) you can find out the coefficients of the AR filter.
4. Get the inverse of it, and then get the MA coefficients.

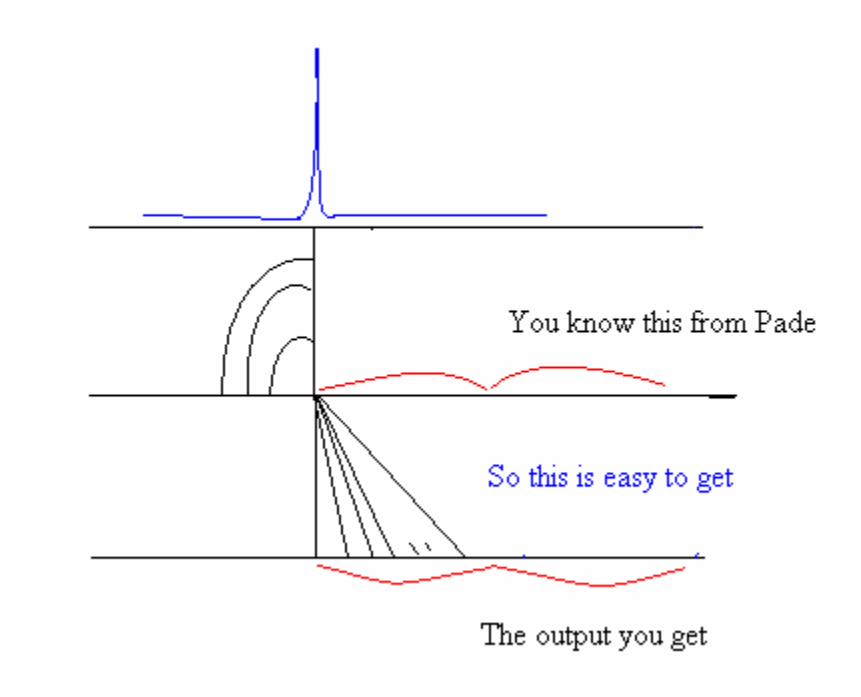
2. Prony's Approximation Technique:



In this you use the funda of all zeroes after q and Least Squares.
So consider lots of zeroes in the input and you get some huge output(for the AR filter). Apply Least Squares and you can get the AR coefficients. Continue as above for MA.

3. Shanks' Technique:

Use the first part of Prony or Pade and get the AR coefficients. Now we are going to consider the equivalent MAAR



You have the impulse response of the AR system i.e the MA input. And given and impulse you know the system output. Use these to get the MA coefficients.

ARMA as a large MA

If you <u>can't control</u> the system input i.e cannot give an impulse then there is another technique

Model the ARMA as a large MA system. Give input in the order of millions, which will get you huge output. Now Least Squares using information will give you a nice approximation of the Impulse Response of the system.

In all the Prony, Pade and Shanks we first found the impulse response, which we now have. So you start with any of these techniques to get the remaining stuff.

Optimal ARMA Modeling

You can use a very good linear method.

We know our ARMA system looks like this

And the equation for a particular y_k will look like this

$$y_k = b_0x_k + b_1x_{k-1} + b_2x_{k-2}\ldots + a_1y_{k-1} + a_2y_{k-2} + \ldots$$

So basically you can have lots of such equations (for different values of k) which can be expressed as

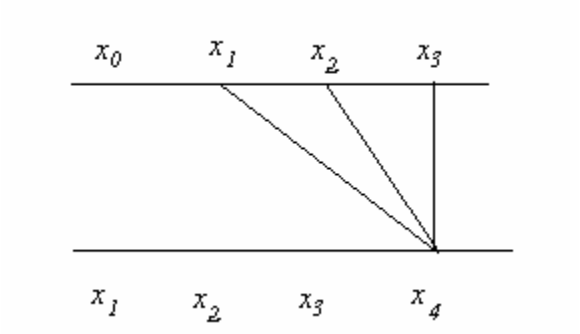
$$\begin{bmatrix} x_0 & 0 & 0 & 0 & 0 & \ldots & 0 & 0 & 0 & 0 & \ldots \\ x_1 & x_0 & 0 & 0 & 0 & \ldots & y_0 & 0 & 0 & 0 & \ldots \\ x_2 & x_1 & x_0 & 0 & 0 & \ldots & y_1 & y_0 & 0 & 0 & \ldots \\ x_3 & x_2 & x_1 & x_0 & 0 & \ldots & y_2 & y_1 & y_0 & 0 & \ldots \\ x_4 & x_3 & x_2 & x_1 & x_0 & \ldots & y_3 & y_2 & y_1 & y_0 & \ldots \\ . & . & . & . & . & \ldots & . & . & . & . & \ldots \\ . & . & . & . & . & \ldots & . & . & . & . & \ldots \\ . & . & . & . & . & \ldots & . & . & . & . & \ldots \\ . & . & . & . & . & \ldots & . & . & . & . & \ldots \\ . & . & . & . & . & \ldots & . & . & . & . & \ldots \\ . & . & . & . & . & \ldots & . & . & . & . & \ldots \\ . & . & . & . & . & \ldots & . & . & . & . & \ldots \\ . & . & . & . & . & \ldots & . & . & . & . & \ldots \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ .. \\ .. \\ a_1 \\ a_2 \\ a_3 \\ .. \\ .. \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ .. \\ .. \\ .. \\ .. \\ .. \\ .. \\ .. \\ .. \end{bmatrix}$$

This again may look like the Toeplitz matrix but isn't, its more of double Toeplitz (notice the x's and y's). Solve this using Least Squares and you will get the linear, optimal ARMA system identification.

Applications of Least Squares:

1. Stock Market Prediction

If the next day's stock market value is dependent on the rise or fall in the pass few days then you can model it as the following system, where all the x's are the stock market sensex everyday.



2. Innovations Process
3. Used for Compression

Fourier Transform Basics

What is the Fourier Transform?

The Fourier Transform is the measurement of a signal along a set of complete , unit , orthogonal set of basis vectors. Completeness means that all real world signals can be expressed without loss of information in terms of the Fourier basis. Orthogonality implies that any component along one basis gives a zero projection on all other basis,i.e changes in component along one basis doesnot affect the measurement along the other basis.It also turns out that this set of basis is symmetric.

So what is these wonderful set of basis , and why we do we like them so much?

This set is the set of complex spirals, each rotating at multiples of a certain base frequency.This is very useful because complex sinusoids e^{jw} are eigen vectors of all linear systems.

And, since we can represent all real world signals as weighted sum of complex sinusoids, it is possible to represent any real world signal in terms of the Fourier basis.

Complex sinusoids (or spiarls) have certain useful prpoerties.

Most important property is that when a complex sinusoid is passed through any linear time invariant system, it only gets scaled at the output. It remains a complex sinusoid of the same frequency.This is so because all that an LTI system can do is scale and shift the input. A shift of a complex spiral by $'x'$ is same as its multiplication by e^{jx}.('Cork Screw effect') .So , at the outputs this only amounts to a scale of the input.

So , all LTI systems can be expressed in terms of how they scale the complex spirals. This is best known to us as $H(w)$ or the frequency response of the system.

Note :: It is important to realize that a real sinusoid is not an eigen vector of LTI systems, as it gets scaled as well as undergo phase change.

Least Squares approximation in the complex domain

We have already seen, in the 2nd lecture, how we can obtain a real number by which if we scale a real vector U it fits another vector X best in the L2 sense.Now, we will extend it complex numbers, i.e. we will find a complex number A , such that if we scale U by A it best fits X in the L2 sense.

L2 norm best fit of X to Y , means that the root sum of squares(R.S.S) of the difference of Y and X is minimum.

If we look at the vector U as a measurement basis that measures a component of X along its direction, then the complex scale A that we find such that U fits X best in the L2 sense, then A is the Fourier component or the eigen value for eigen vector U in X.

Why is the L2 norm used usually ?

Refer to fig-1.

The y axis is the minimaztion function(remanant vector) and x-axis is the independent variable(scale).

Now we notice that in case of the L1 norm,the difference signal is not differentiable at its minimum point. Such is not the case for the L2 norm, which is parabolic and differentiable at the minima.

Problem ::

Given complex vectors X and U we need to find a complex scalar A such that

$||X - aU||_2^2$

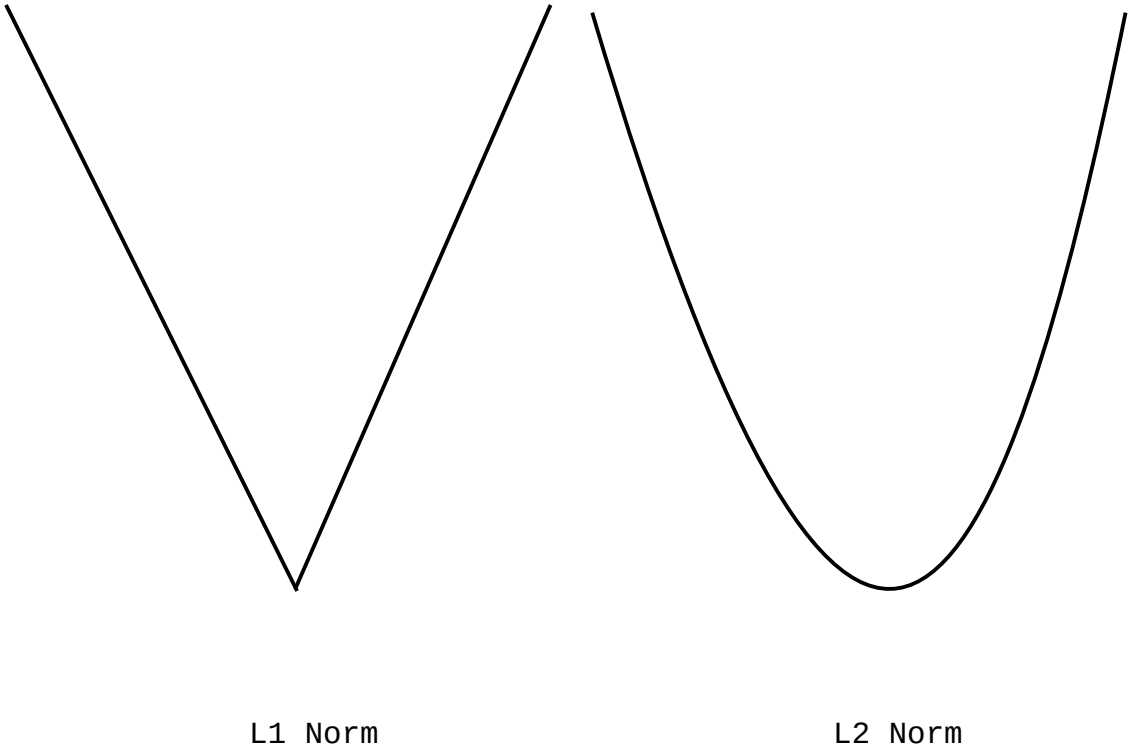


Figure 1: Comparisons of the L1 and L2 norms

is minimum.

Solution ::

$E = \int (X - AU)(\overline{X - AU})$

$E = \int (X - AU)(\overline{X} - \overline{AU})$

$E = \int (X - AU)(\overline{X} - \overline{A}.\overline{U})$

This is because

1. $\overline{(p+q)} = \overline{p} + \overline{q}$

2. $\overline{(p.q)} = \overline{p}.\overline{q}$

$E = \int (X\overline{X} - AU\overline{X} - X\overline{AX} - AU\overline{AU})$

$E = \int X\overline{X} - \int AU\overline{X} - \int A\overline{A}.\overline{U} + \int U\overline{U}$

Rewriting the above equation in a simpler form,

$E = C + BA + \overline{B}.\overline{A} + Aa\overline{A}$

$$E = \begin{pmatrix} A_{real} & A_{img} \end{pmatrix} \begin{pmatrix} 2A & 0 \\ 0 & 2A \end{pmatrix} \begin{pmatrix} A_{real} \\ A_{img} \end{pmatrix} + \begin{pmatrix} 2B & -2B \end{pmatrix} \begin{pmatrix} a_{real} \\ a_{img} \end{pmatrix} + \begin{pmatrix} C & 0 \\ 0 & C \end{pmatrix}$$

This a quadratic equation in matrices, just like the one we obtained in 2nd lecture. Solving this ,

$Min = (A + \overline{A})^{-1}(2B_{real} - 2B_{img})^T$

$$Min = \begin{pmatrix} 2A_{real} & 0 \\ 0 & A_{img} \end{pmatrix}^{-1} \begin{pmatrix} 2B_{real} \\ 2B_{img} \end{pmatrix}$$

$MinScale = \overline{B}/A$

$MinScale = \int U\overline{X}dt / \int U\overline{U}dt$

$MinScale =< u, x > / < u, u >$

Observation ::

DOT(X , U) is not commutative. In fact it is conjugate of DOT(U , X)

Domain of the 4 types of the F.T

There 4 types of Fourier transforms depending on their domains. Here is the classification.

TRANSFORM :: DOMAIN

Fourier Series :: Continous Finite

Fourier Transform :: Continous Infinite

Discrete F.T :: Discrete Infinite

Discrete Time F.T :: Discrete Finite

Discrete Fourier Transform

Time domain is Discrete and Finite Frequency domain is Discrete and Finite

The Fourier basis here is a set of N discrete time complex spirals rotating at frequency that is integral multiples of 2pi/N.

Equation of the k'th spiral : $u_k[n] = e^{jk2pi/N}$

NOTE :: Different normalization factors are used for these basis vectors, but they only affect the scaling of the components in the other domain. The normalization factor of

$$1/\sqrt{N}$$

, gives same energy in both domains.

For measurement over N such spirals rotating at speeds k = 0 , 1 ,2 ... N-1,the above result the Fourier coefficient matrix X is obatained from x as

$$X = \begin{pmatrix} \overline{u_0} \\ \overline{u_1} \\ \overline{u_2} \\ \overline{u_3} \\ \overline{u_4} \\ \overline{u_5} \\ \vdots \end{pmatrix} x$$

The reverse transformation is ,

$$x = (\, u_0 \quad u_1 \quad . \quad . \quad .\,)\, X$$

We can show that the set of basis are mutually orthogonal.

Lets take two unit complex spirals one that completes w1 turns in N samples, other that takes w2 turns. If we dot product of the two, the resultant is a complex spiral of w1 + w2 frequency. So, over N samples, the spiral completes w1 + w2 rotations. The sum over w1 + w2 cycles is always zero.

To see why the sum of a complex spiral, like the one above, over w1 + w2 cycles is zero ,see the figure below. Assume that N = 8

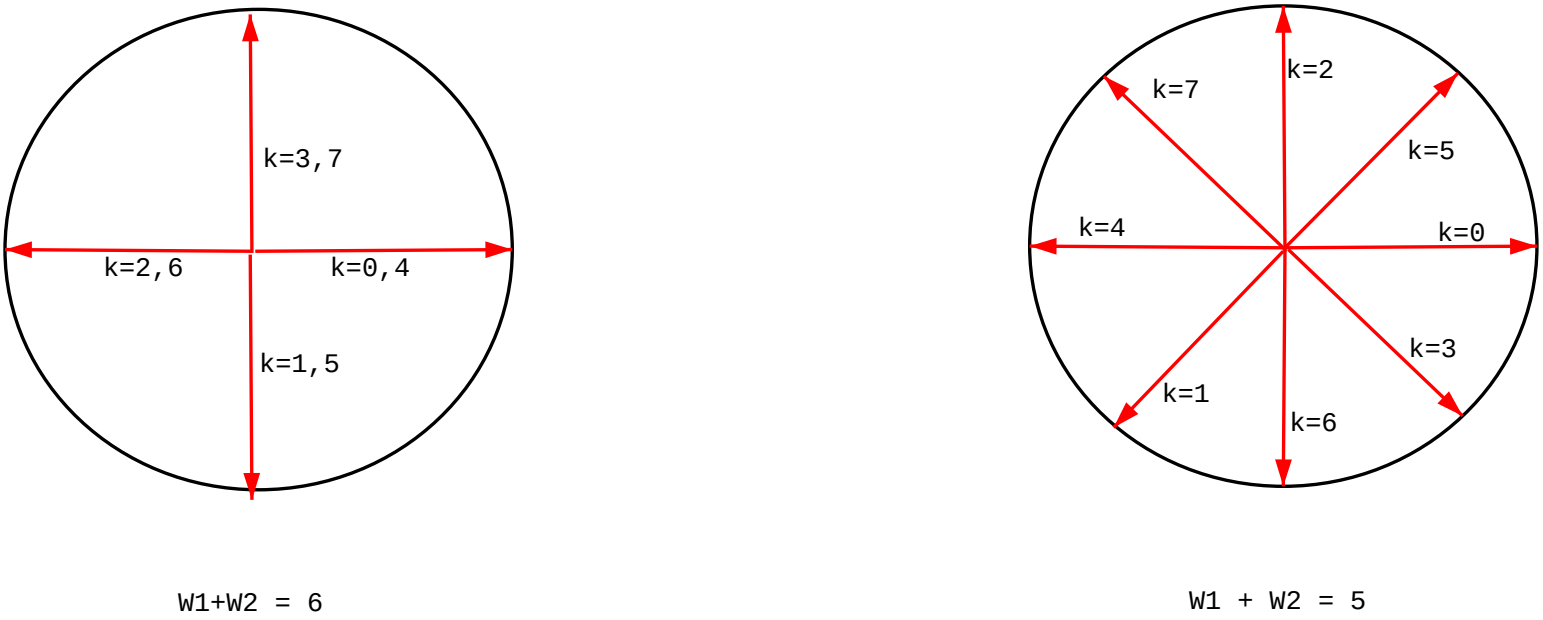


Figure 2: Sum of N=8 spiral over w1+w2 cycles is 0

What we notice as that the vectors are all having their tips along the unit circle,i.e. their magnitudes are the same, and there are pairs of such vectors pointing in excatly opposie directions, their total sum due to cancellation will be 0. Remember that when we add to vectors the addition is by triangle law.So two opposite vectors yeild a zero. The thing to notice is that only such opposite pair of vectors occur.

Of course, the more rigorous algebraic proof is very simple to do. It is just a summation with sqiggly math charcatars, and not as colorful as the one above.

Hence, what is seen is an overview of what Fourier Transform is. Its application and properties will be seen in the next lecture.

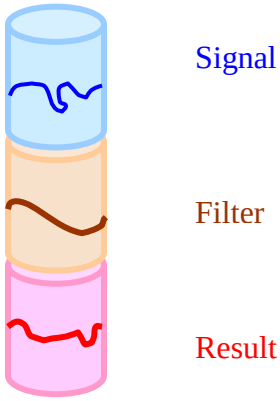
ADSP : Lecture 6

8th Feb 2004
Sunday [@ Codito, Pune]
Scribed by: Tanmay Dharmadhikari

The lecture started with circular convolution calculation viewable with cylinders. This technique is also useful when the filter kernel is larger (in sample size) than one period of the signal itself.

Circular Convolution: Using Cylinders

Consider the signal and filter to be marked on two different cylinders as shown in the figure below. For circular convolution:



a.) The filter cylinder and result cylinder both, are rotated 1 unit about their height axes.

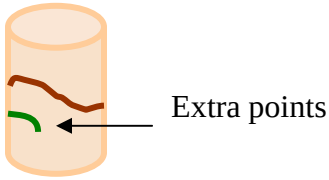
b.) The corresponding points on the signal cylinder and filter cylinders are multiplied with each other.

c.) These products are added to give the output at that point of the result cylinder.

d.) On performing these steps for all the points we unfurl the result cylinder to get the convolution result.

What if the filter size is larger than one period of the signal? The most intuitive thing is to change the size of the filter cylinder as well. Another solution that we can see is to mark the filter kernel on the cylinder spirally i.e. in rudimentary terms let the extra kernel points get marked below the kernel.

The extra points are also used during the convolution along with the corresponding point from top.



Thus during one cycle of our convolution process, for some points on the signal, we perform two multiplications.

This idea can be simplified by adding the extra points into the corresponding kernel points that align with them. Thus, we get a new kernel of the same size as the signal period.

With this clear we moved on to show that convolution of two signals is easier to calculate by

(i) Decomposing the signals into their spiral components

(ii) Convolving only those corresponding spirals which have the same frequency

(iii) And finally synthesizing all these individual results to get the final result i.e. the convolution of the signals.

Representing this algebraically:

Let x and f be the signal and filter we want to convolve. Lets decompose x and f into the spirals given by:

$$x = x_1 + x_2 + \ldots + x_n$$

$$f = f_1 + f_2 + \ldots + f_n$$

x_i, f_i are the corresponding spiral components of x and f with same frequencies

$$\begin{aligned}\therefore (x * f) &= (x_1 + x_2 + \ldots + x_n) * (f_1 + f_2 + \ldots + f_n) \\ &= (x_1 * f_1) + (x_1 * f_2) + \ldots (x_1 * f_n) \\ &\quad + (x_2 * f_1) + (x_2 * f_2) + \ldots (x_2 * f_n) \\ &\quad \ldots \\ &\quad + (x_n * f_1) + (x_n * f_2) + \ldots (x_n * f_n)\end{aligned}$$

Now, we can prove that if the two convolving spirals do not have the same frequency then their resultant is zero and that only those with same frequency contribute to the value of the final convolution result. Thus the above equation reduces to:

$$(x * f) = (x_1 * f_1) + (x_2 * f_2) + \ldots (x_n * f_n)$$

The convolution of x and f is thus reduced to the ‘n’ convolutions of the ‘n’ corresponding spiral components and adding their individual resultants.

This procedure is feasible because we know or rather have the data about each spiral component viz. :

(i) Starting phase

(ii) Individual frequency

(iii) Individual amplitude

The convolution result of each corresponding component pair is given by the spiral with:

(i) Starting phase = $x_{i\theta} + f_{i\theta}$

(ii) Frequency = $x_{i\omega} - f_{i\omega}$

(iii) Amplitude = $N \left| x_{i\omega} \right| \left| f_{i\omega} \right|$

[N is no. of sample points.]

We noted when to use which form of the Fourier Transform depending on the signal domain:

Signal Domain	**Transform**
Discrete & Finite	DFT
Continuous & Finite	Fourier Series
Discrete & Infinite	DTFT
Continuous & Infinite	FT

Shift Property:

Shifting a sequence in time results in the multiplication of the DTFT by a complex exponential (linear phase term):

$$x(n - n_0) \overset{\text{DTFT}}{\longleftrightarrow} e^{-jn_0 w} X(e^{jw})$$

We also saw the application of using rules governing linear systems to simplify convolution of continuous signals:

Modify one of the signals in some linear way, perform the convolution, and then undo the original modification. As an example we used the *derivative* as the modification and it is undone by taking the *integral*.

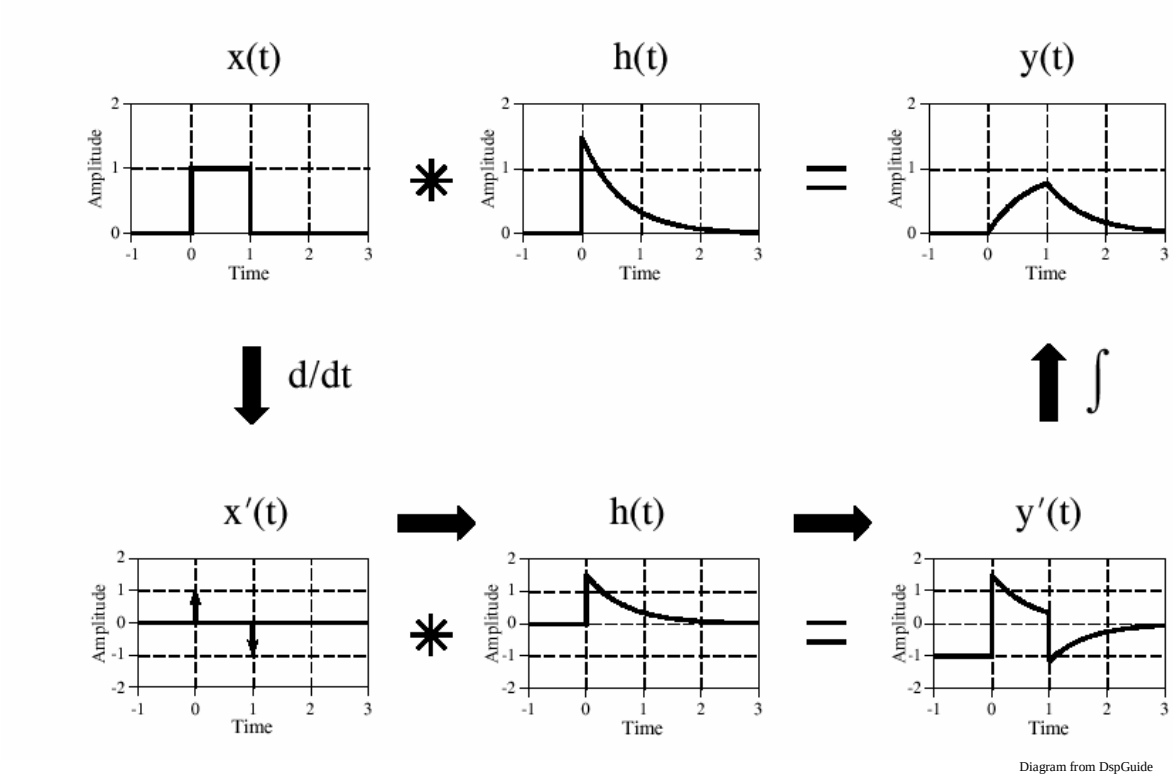


Diagram from DspGuide

Advanced Digital Signal Processing
Lecture 7
Date: 10-2-2004
Scribed by Aditya Thakur

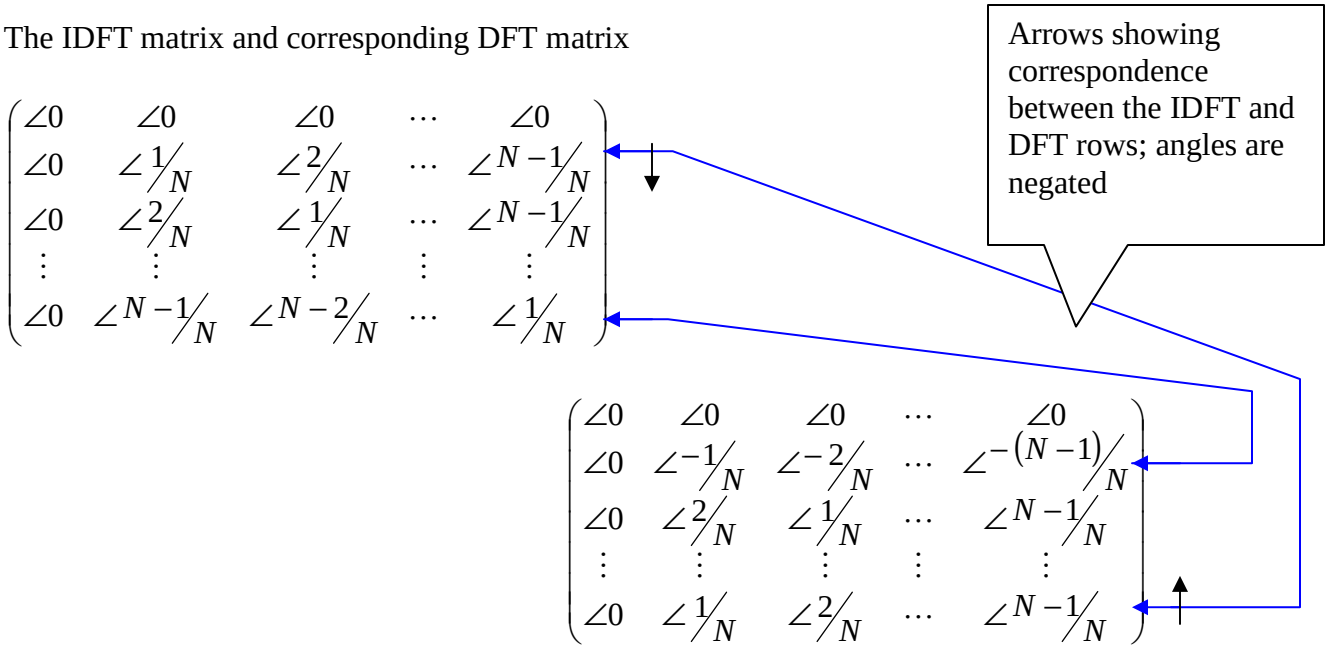
Properties of FT (*contd.*):

Duality property:

Similar to the duality theorem in Boolean algebra wherein $0 \leftrightarrow 1$, AND $\leftrightarrow$ OR.

Here,
Frequency $\leftrightarrow$ time

The IDFT matrix and corresponding DFT matrix



$$\begin{pmatrix} \angle 0 & \angle 0 & \angle 0 & \cdots & \angle 0 \\ \angle 0 & \angle 1/N & \angle 2/N & \cdots & \angle (N-1)/N \\ \angle 0 & \angle 2/N & \angle 1/N & \cdots & \angle (N-1)/N \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \angle 0 & \angle (N-1)/N & \angle (N-2)/N & \cdots & \angle 1/N \end{pmatrix}$$

$$\begin{pmatrix} \angle 0 & \angle 0 & \angle 0 & \cdots & \angle 0 \\ \angle 0 & \angle -1/N & \angle -2/N & \cdots & \angle -(N-1)/N \\ \angle 0 & \angle 2/N & \angle 1/N & \cdots & \angle (N-1)/N \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \angle 0 & \angle 1/N & \angle 2/N & \cdots & \angle (N-1)/N \end{pmatrix}$$

So if we do $DFT(DFT(x))$, we get a flipped or inverted x at the output. This is because the DFT matrix is orthogonal and symmetric (not conjugate symmetric).

Windowing Theorem: Duality applied to the Convolution theorem:

Point wise multiplication in time $\leftrightarrow$ Convolution in frequency domain

Applications of windowing theorem:

1) Analysis of infinite signals:

Given an infinite signal, we can use a 'suitable' size window to clip it in the time domain, so that it is easier to analyze.

The size of the window depends on the nature of the signal being analyzed:

Slow changing signal $\leftrightarrow$ larger window size

Fast changing signal $\leftrightarrow$ a small window

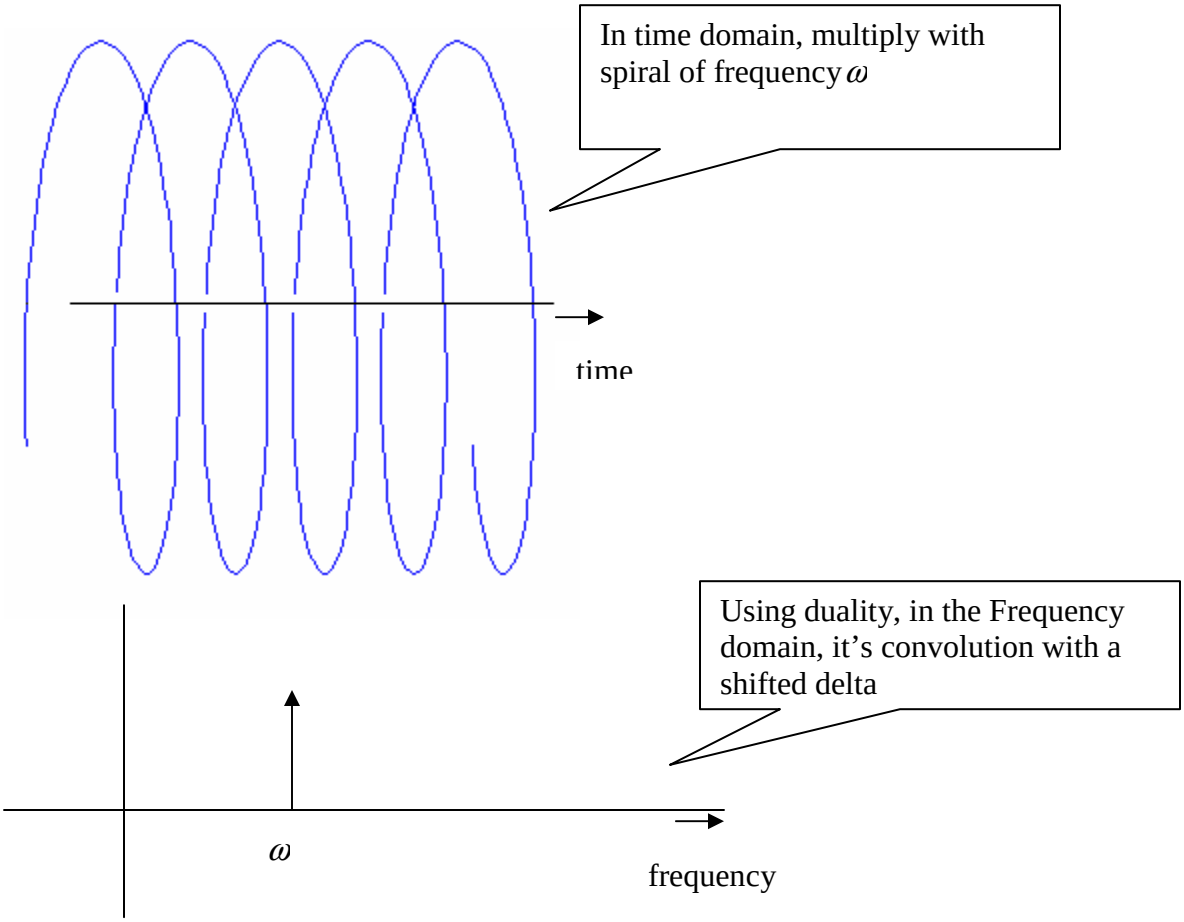
A large window size implies more precise frequency information, while a higher time resolution (small time window) implies worse frequency resolution. This is similar to Heisenberg's Uncertainty Principle, wherein momentum and position, time and energy etc were canonically conjugate to each other.

Another view to explain this phenomenon is by using information theory. Short signals have inherently smaller frequency resolution, and we cannot 'create information'. Of course, using non-linear heuristics it is possible to get better results.

2) Amplitude modulation:

The other application where we point wise multiply in the time domain is in amplitude modulation.

Consider a modulation of a given signal with a single complex spiral having frequency ω. This results in the convolution of the signal with a delta shifted by ω in the frequency domain, which of course results in the frequency shift of the signal from base band to higher band.



The above reasoning used the Duality theorem.

One can also think about it directly by decomposing the signal and the modulator into spirals and using the fact that the multiplication of to spirals of frequencies $\omega 1$ and $\omega 2$, results in a spiral of frequency $\omega 1 + \omega 2$ (a frequency shift).

So, the modulation theorem is the dual of the Shift theorem!

Dual of the Derivative theorem:

Multiplication in time domain by a ramp $\leftrightarrow$ differentiation in frequency domain

$$tf(t)dt \qquad \leftrightarrow \qquad \frac{dF}{d\omega}$$

$$\int_{-\infty}^{\infty} tf(t)dt \qquad \leftrightarrow \qquad \frac{dF}{d\omega} \quad \text{, at } \omega = 0 \text{ (the dc value)}$$

The LHS represents the expected value of the signal $\mathrm{E}[X]$.
This property is also known as the *First moment property* of the FT.

Similarly,

$$\mathrm{E}[X^2] = \int x^2 f_X(x)dx = \frac{d^2F}{d\omega^2} \text{ at } \omega = 0$$

is called the *Second moment* of the signal.

Together these 2 properties are called the Moment generating properties of the FT, and are used to find the Variance of the signal.

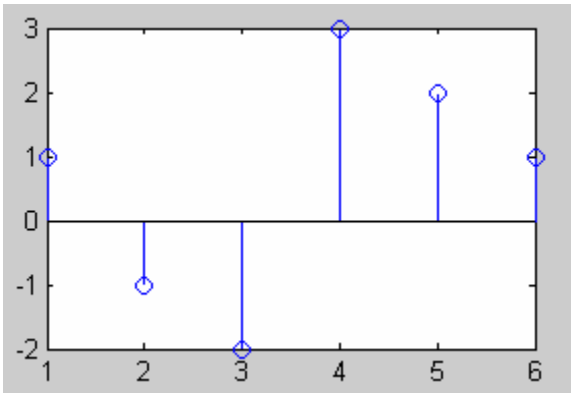
Another interesting point we covered was that when you *add* 2 random variables, their probability mass functions get *convolved*!

Fast Fourier Transform:

2 types:
1) Decimation in time
2) Decimation in frequency

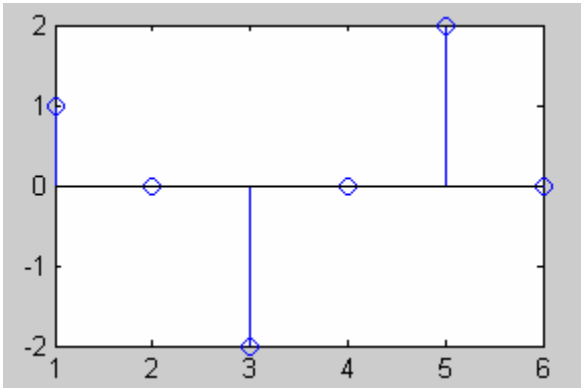
Steps for decimation in time FFT:
(For the following figures, imagine that the x-axis index starts from 0, not 1)

Given a signal 0,

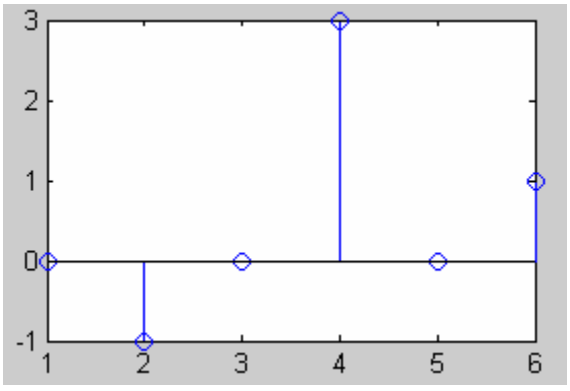


Separate it into even and odd bands 1 & 2 (using linearity),

1: Even band

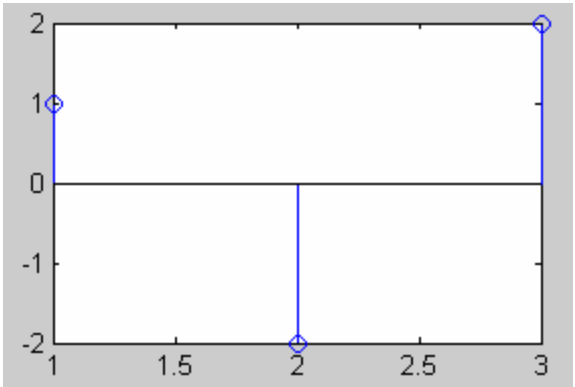


2: Odd band

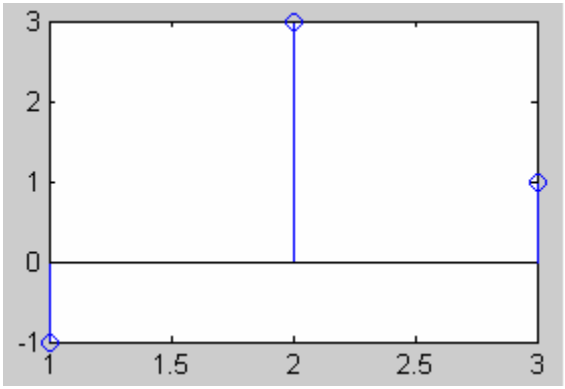


2. Get 1′ & 2′ by down sampling.

1′ *:*



$2'$:



3. Recursively get the DFT of these 2 signals
4. Use stretch theorem (and rotation by w for shifted odd band) to get DFT of 1 & 2
5. Then again use linearity to get DFT of 0

The following recursion represents this process,

$T(n) = T(n/2) + n$
Which has the solution,
$T(n) = \Theta(n \lg n)$

This procedure represents an order improvement over the naive N^2 algorithm. A similar analysis can be carried for decimation in frequency.

Other improvements are possible:

- Sparse matrices are used to get from $1' + 2'$ to 0, since they can be multiplied in linear time.
- Other bases reduce the constant in the $n \lg n$ and are more suitable for use on computers.
- Taking a 4-point instead of 2-point, gives rise to some common sub-expression savings.
- Algorithms such Prime Number decomposition deals with finding DFTs of non-dyadic sequences.

These will be discussed in detail in later lectures.

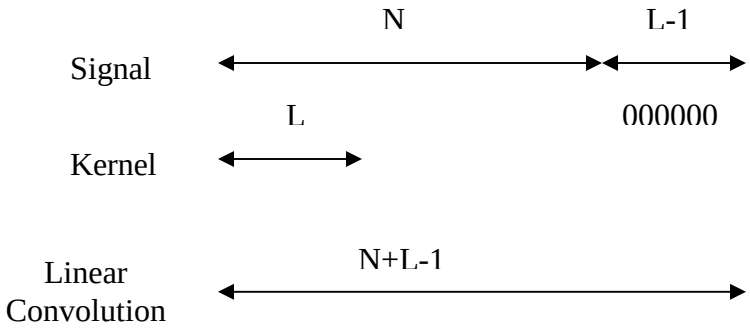
Another interesting point about the FFT is that apart from it being faster it is *more precise*: each coefficient is found after $\lg n$ multiply-adds; as opposed to the n for naïve DFT. This leads to less round-off error.

Interestingly, the best way to minimize round-off error when adding a sequence of numbers is to add the smallest two each time.

So with all our knowledge, we can now do *circular* convolution in $n \lg n$.

What if you wanted to get linear convolution?

Add enough padding to the signal to avoid wraparound distortion.



Can't use such a large FFT because:

- Real-time applications: larger N, more time taken to do the convolution.
- Precision problems: more N, more additions, less precision.

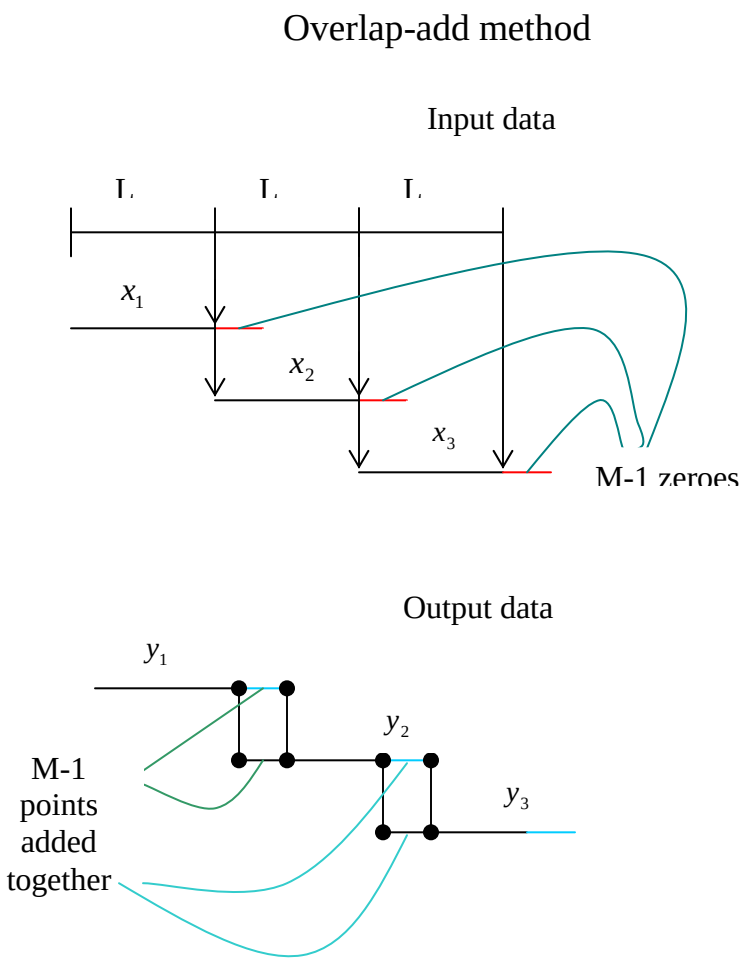
So you would have a case where your kernel is of length 40, your signal is of length 1 million, and you are only allowed (due to the two reasons mentioned above) to take a FFT of 1000!

In which case, you use techniques such as *Overlap-add* and *Overlap-save* to do convolution, but don't get the transform.

Overlap-add method:

M : size of FIR filter kernel
L : size of the input data block
N=L+M-1: size of DFT and IDFT

To each data block we append M-1 zeroes and compute N-point DFT. Since each data block is terminated with M-1 zeroes, the last M-1 points from each output block must be overlapped and added to the first M-1 points of the succeeding block.

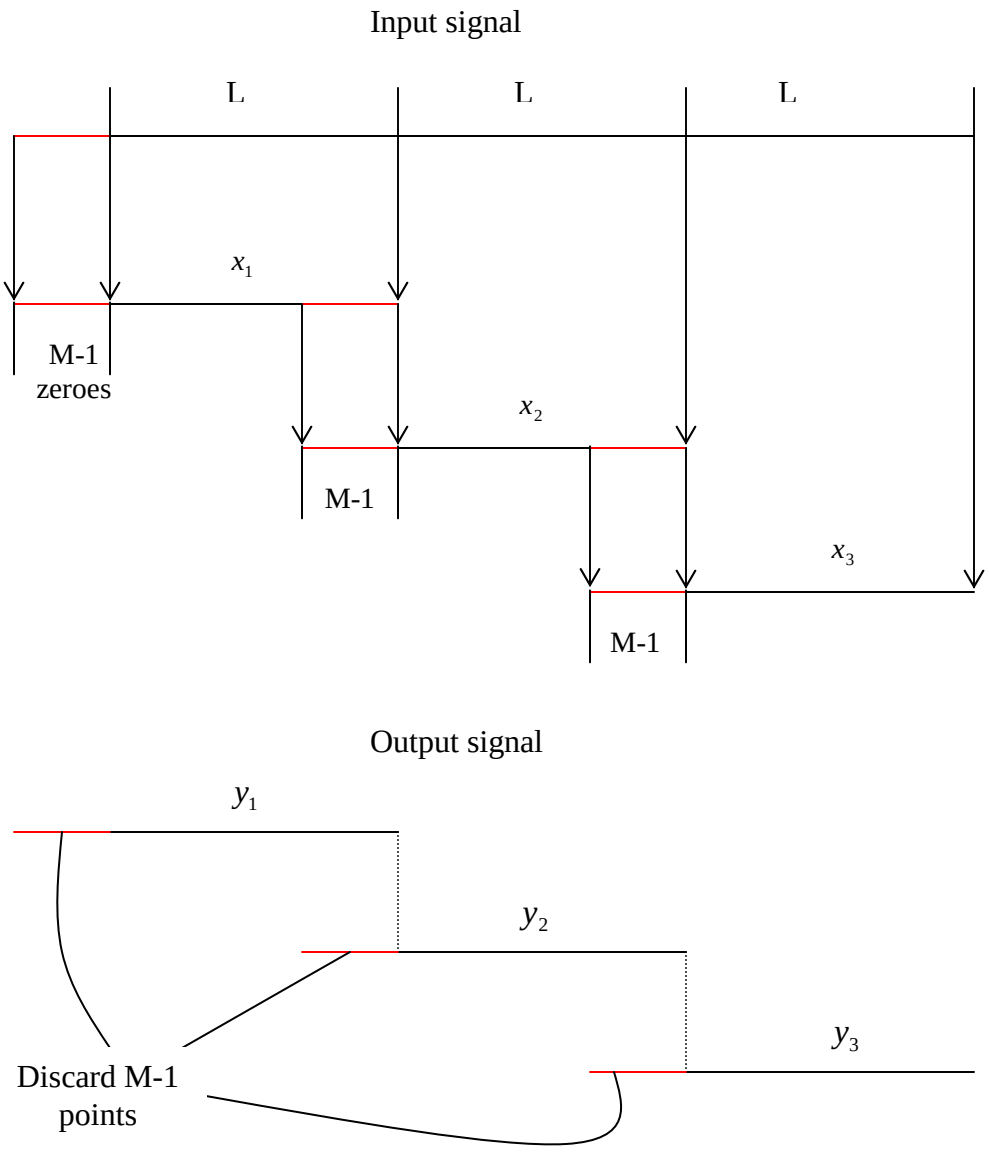


Overlap-save method:

M : size of FIR filter kernel
N=L+M-1: size of data blocks
N : size of DFT and IDFT

Each data block consists of M-1 points from the previous data block followed by L new data points to form a data sequence of N=L+M-1. The kernel length is increased by appending L-1 zeroes and an N-point DFT is computed and stored. The first M-1 points of this are corrupted by aliasing and must be discarded. The last L points are exactly the same as the result from linear convolution.
To avoid loss of data due to aliasing, the last M-1 points of each data record are saved and these points become the first M-1 points of the subsequent record. To begin processing, the first M-1 points are set to zero.

Overlap-save method



INTRODUCTION TO UPSAMPLING & DOWNSAMPLING

What is Sampling Rate Conversion?

As the name suggests, the process of converting the sampling rate of a digital signal from one rate to another is Sampling Rate Conversion.

Increasing the rate of already sampled signal is Upsampling whereas decreasing the rate is called downsampling.

Why to do it?

Many practical applications require to transmit and receive digital signals with different sampling rates. So at many stages in application we need to do sampling rate conversion.

How to do it?

There are two ways in which we can achieve sampling rate conversion:

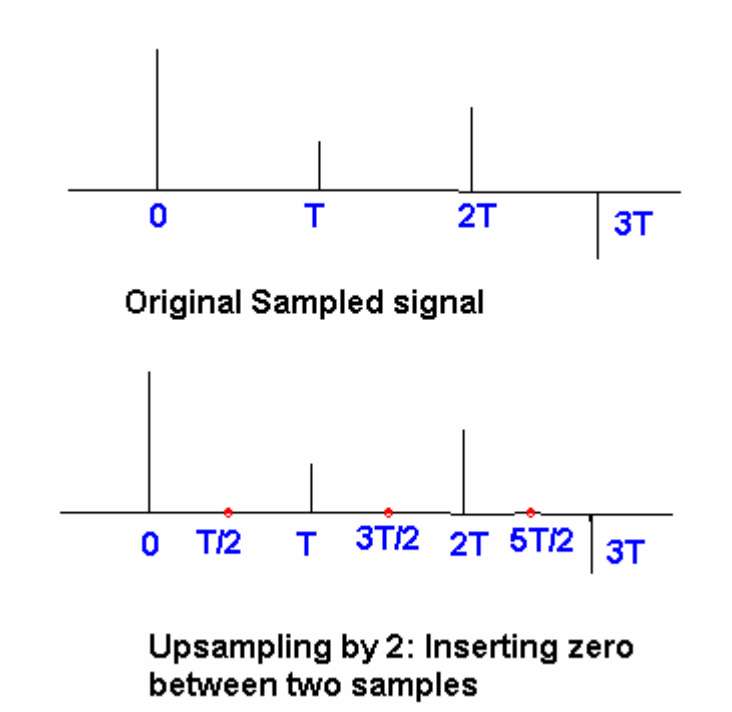
1. First approach is to do D/A conversion to recover back original analog signal. Then we can do A/D conversion with desired sampling rate. The advantage of this technique is that the second sampling rate need not hold any special relationship with old one. But the problem is signal distortion introduced by D/A and quantization effects of A/D.

2. Second approach is to work in digital domain only. So we'll have to predict the digital signal with desired rate using the already sampled signal in hand.

Now for this lecture, we'll look at the second choice of sampling rate conversion.

UPSAMPLING

Let's consider, simplest case of upsampling. We want to double the sampling rate of signal. So what we do is insert 0s in between two successive samples. As shown:

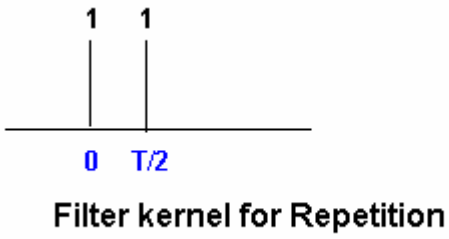


Obviously this is a bad approach. As we don't have data for intermediate samples, let's generate it.

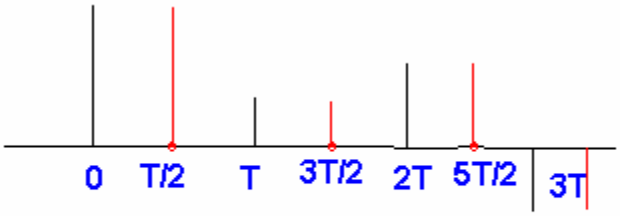
Method-1: Repetition

Repeat the current sample.

The corresponding filter kernel will be



The output waveform will be:

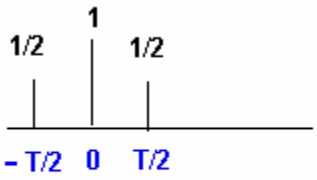


Upsampled by 2 using repetition

Method-2: Interpolation

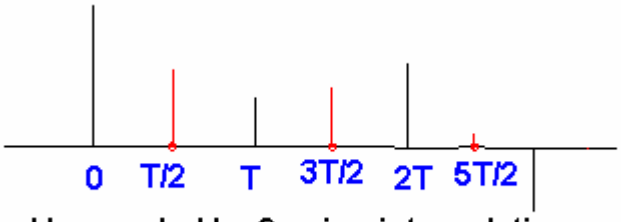
The intermediate sample value will be the average of its neighboring values.

The filter kernel will be:



Filter Kernel for interpolation

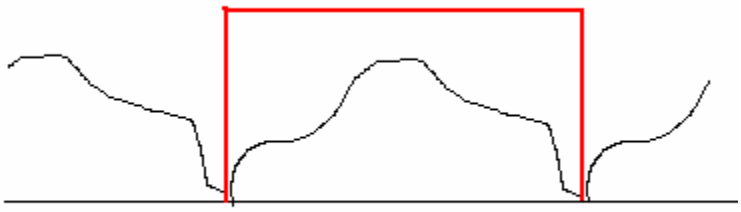
The output waveform will be:



Upsampled by 2 using interpolation

Now this a very good method but it produces, two aliases of each frequency in frequency domain . So we should cut the high frequency contents to avoid aliasing.

Why to cut?

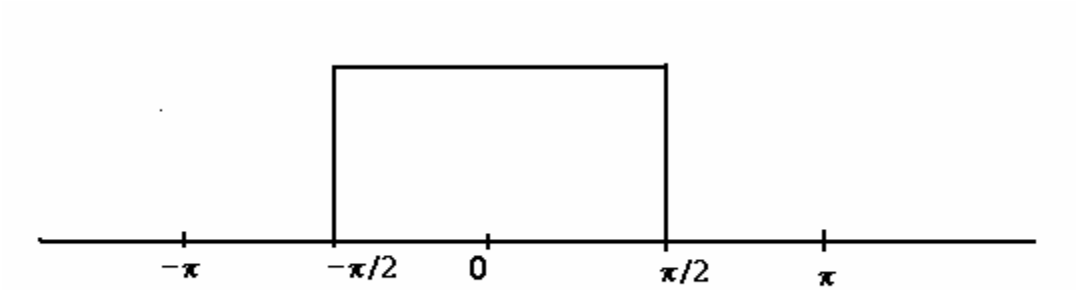


Use of rectangular window

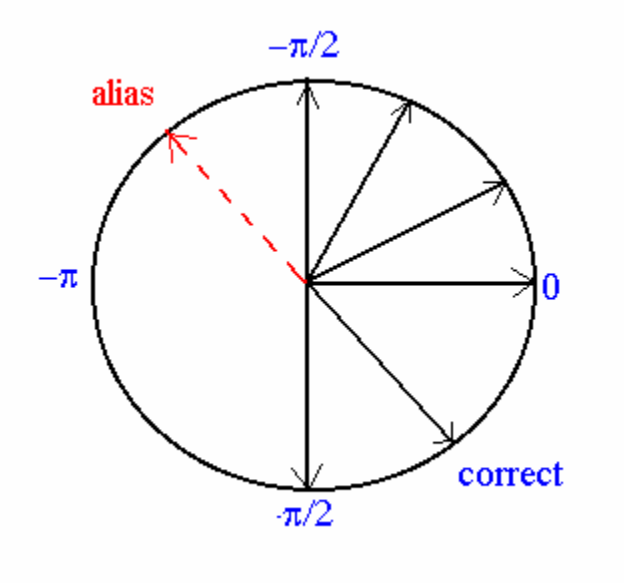
Because they don't contain any new information. They are just repeating information. We also know that maximum frequency content in original signal cannot be greater than **Fs/2** so there is definitely no more information in baseband. So we can afford to cut high frequency contents.

How to cut it?

Use a perfect low pass filter. That is we will keep slow moving spirals and reject fast moving spirals in frequency domain.



Low pass filter selecting frequecies between $-\pi/2$ to $\pi/2$



so filter kernel in frequency domain is as shown. Its amplitude is 1 for frequencies in the range - π/2 to +π/2 and zero for rest all frequencies. So we cut the high frequency aliases. So filter kernel in frequency domain is set of slow moving spirals having amplitude 1.

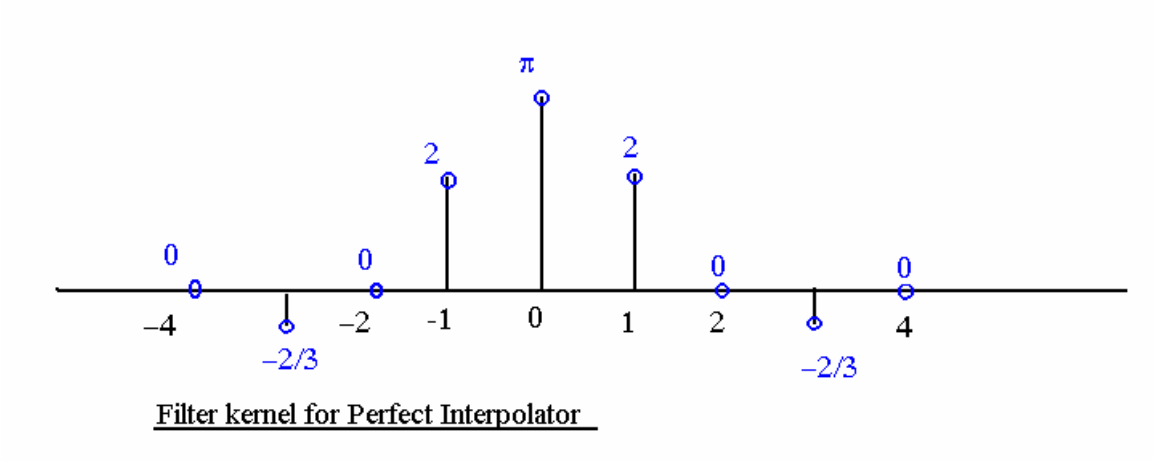
Let's figure out, what will be the corresponding filter kernel time domain.

$$\int_{-\pi/2}^{\pi/2} e^{0}\, d\omega = \pi/2 - (-\pi/2) = \pi$$

$$\int_{-\pi/2}^{\pi/2} e^{j\omega}\, d\omega = 1/j(e^{j\pi/2} - e^{-j\pi/2}) = 2$$

$$\int_{-\pi/2}^{\pi/2} e^{2j\omega}\, d\omega = 0$$

$$\int_{-\pi/2}^{\pi/2} e^{3j\omega}\, d\omega = -2/3$$



Filter kernel for Perfect Interpolator

Thus the perfect interpolator is an Infinite Impulse Response (IIR) filter. The filter is not causal hence cannot model using ARMA. We cannot implement the filter because it's infinite.

There is a class of functions called Analytic functions. According to Taylor, all the information in the analytical function is at zero, so you don't have to go far. We can express these functions as Lauriant series and model them. But for this, the function should be continuous at all points. But our perfect interpolator filter kernel is discontinuous at - $\pi/2$ and $\pi/2$.

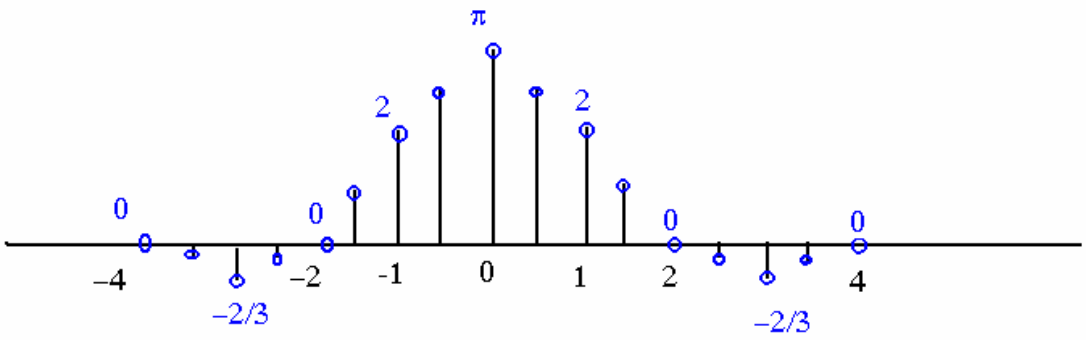
One interesting observation is that, we are getting zeros at previously sampled points in the filter kernel.

So to do upsampling faster we can use Multi-channel Polyphase Multi-sampler with two filter banks.

1. 1^{st} filter bank doing nothing (corresponding to already sampled points).
2. 2^{nd} filter bank with half sample delay.

UPSAMPLING BY 4

Here the filter kernel will look like this:



Filter kernel for Perfect Interpolator for upsampling by factor 4

In this case, we'll have to predict three new samples, between already present pair of samples. We now will have 4-filter banks.

1. Bank with 0 sample delay.
2. Bank with 1/4 sample delay.
3. Bank with 1/2 sample delay.
4. Bank with 3/4 sample delay.

PRACTICAL DESIGN OF FILTER:

Use of Linear Phase Filters:

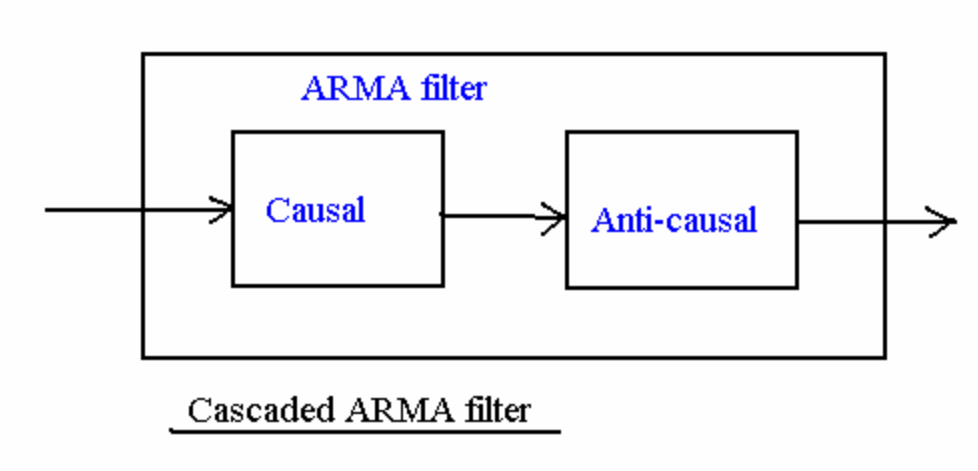
For practical design of upsampler filters, we use delayed symmetric filters.

Zero phase filters are symmetric about zero. Delayed symmetric filters have symmetry along a shifted point. They are also called as Linear phase filters. The advantage of this approach is the filtering can happen at real time. But disadvantage is that the output is not sharp.

Use of ARMA filters:

Using ARMA filters, we can get sharper output. But we cannot get real time processing using ARMA filters.

For offline processing, we can use two ARMA filters, one Causal and second Anti-causal filter. Then we just add the outputs and we'll have desired result. The filter kernel of Causal and Anti-causal filters will be just the flips of one another.

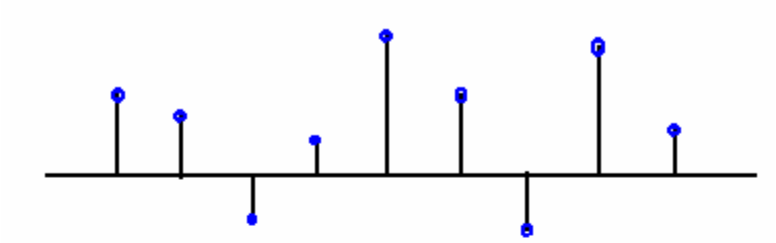


Cascaded ARMA filter

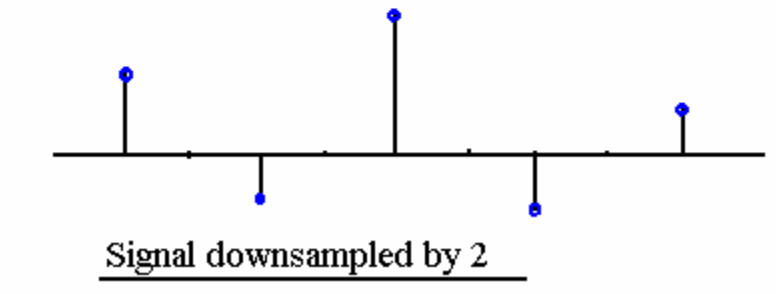
DOWNSAMPLING:

As said, Downsampling is decreasing the sampling rate of a signal. Let's consider a simple case of downsampling a signal to half of its original sampling rate.

Simplest way to do this is to forget every other sample and we'll have the desired sampling rate.

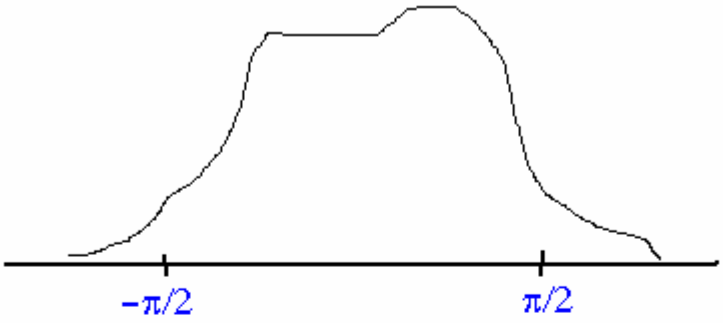


Original Signal

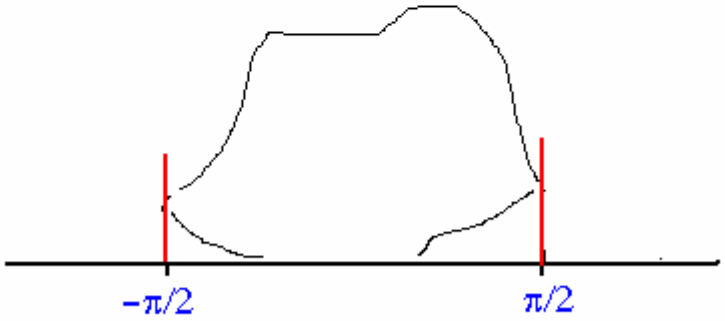


Signal downsampled by 2

But if we reduce the sampling rate just by selecting every other sample of x(n), the resulting with folding frequency Fs/2 . The frequency spectrum tries to spread up but it cannot do so. Hence it winds up on itself.



Original Spectrum



Spectrum wound up on itself

To avoid aliasing, we must first reduce the bandwidth of x(n) to $\omega_{max} = \pi/2$. So we should cut out high frequency contents to avoid aliasing.

Filter Design for Down/Up sampling

Last lecture we saw Downsampling and Upsampling techniques which concludes-
Down sampling
1.Information is Lost when signal is downsampled
2.down sampler causes aliasing
Up sampling
1.information is NOT lost when signal is up sampled
2.Up sampler produces Spectral images.

Solutions:
Down Sampler
Input Signal is passed through a low pass filter and Band limited to (π/D).
Then it is downsampled at the rate (Fx/D) where Fx=Sampling rate of input signal

Up Sampler
In Up sampler signal is upsampled at the rate Fx*I and then passed through a low pass filter which eliminates the spectral images.(by rejecting all the values above ($\omega=\pi$/I).

Efficiency in Filter design
Upsampler
In a typical Upsampler followed by Low pass filter (LPF) :
LPF works at the rate of : I*Fx
But we know that Upsampler inserts I-1 zeros in-between two samples so lots of filter coefficients will be zero which implies that I-1 multiplication and addition in the filter gives out output zero which is perhaps same as input i.e. upsampled signal). This leads some kind of redundancy.

So we come up with next solution
We can combine the Upsampler and Filter kernel such that input signal is multiplied by filter coefficients and then Upsampled to insert zeros (in effect instead of just bypassing zeros of upsampled data in filter we first pass each input sample through filter and insert (I-1) zeros at the output and combine the result of each sample of input signal)
Thus now Filter operates at input sampling rate Fx.

Thus reduction in Filter frequency is of 1/I.

Similarly it can be designed for downsampler i.e. by combining the filter kernel and downsampler, we first select the Dth sample and multiply it with filter coefficient. So filter works at Fx/D where in a typical case it would be operating at Fx.
Thus reduction in Filter frequency is of 1/D.

USE of symmetric property of Filter kernel

We can still reduce the multiplications in filtering operation by NOT calculating the h(i+M/2)*Xj which will be same as h(i)*Xj (M is size of filter kernel) .Thus we can reduce no. of multiplications by ½.It will also reduce space required for storing these multiplications.

Polyphase filter design for Upsampler

If we observe the Upsampling process ,it introduces I-1 zeros thus if filter neglects(actually holding I-1 samples) first I-1 inputs from upsampler then we will get a output of I samples in next time slot.

Now if M is size of filter kernel (that means it can hold M inputs before giving outputs) then in each of next time slot we will get FLOOR (M/I) no. Of NON-ZERO output samples, which are then multiplied with filter coefficients.
We are interested in finding out these NON-ZERO samples.
Hence we can consider Upsampler as a parallel combination of I filters which starts giving I output samples after first I-1 samples.
Working goes like this:
Assuming M as a multiple of I, in first I samples there are (M/I) non-zero sample output which are multiplied with a downsampled version of h(n)(filter sequence) h(0),h(I),h(2I).. For next input sample these samples will get shifted (delayed) by one and will still have M/I no. of non-zero samples which will get multiplied by h(1),h(I+1),h(2I+1)..
Size of each such filter bank will be (M/I).

If we observe the first filter bank it is
Downsampled original filter kernel h(n) with D=I
And
Each such next filter i is downsampled signal of shifted filter kernel h(n+i)
Where 1<=i<=I-1.
Thus each filter differs in phase hence called as POLYPHASE filters
Thus there are I different filters acting on I samples.
Similar kind of design can be done for Downsampling.

Now I-1 filters work at frequency Fx and Output of each filter is collected at rate I*Fx
Hence reduction in filter calculations we obtain is : (I-1)/I

Rational Sampling rate conversion (p/q)
There are two approaches for doing rational sampling.
DOWN-> UP sampling
Downsampling signal first and then upsampling looses information in signal.
Since downsampling selects every qth sample and upsampler then inserts p zero the output signal do not contain same information.
UP->DOWN sampling
Upsampling signal first (inserting zeros i.e. no information loss) and then downsampling do not have information loss but signal is aliased. But aliasing can be eliminated by anti-aliasing filter.

Hence in Rational sampling rate conversion Upsampling is done before Downsampling. This kind of sampling can be obtained by cascading two polyphase filters of an upsampler and downsampler.

Thus combining the 2 filter kernels of we can get the desired result.

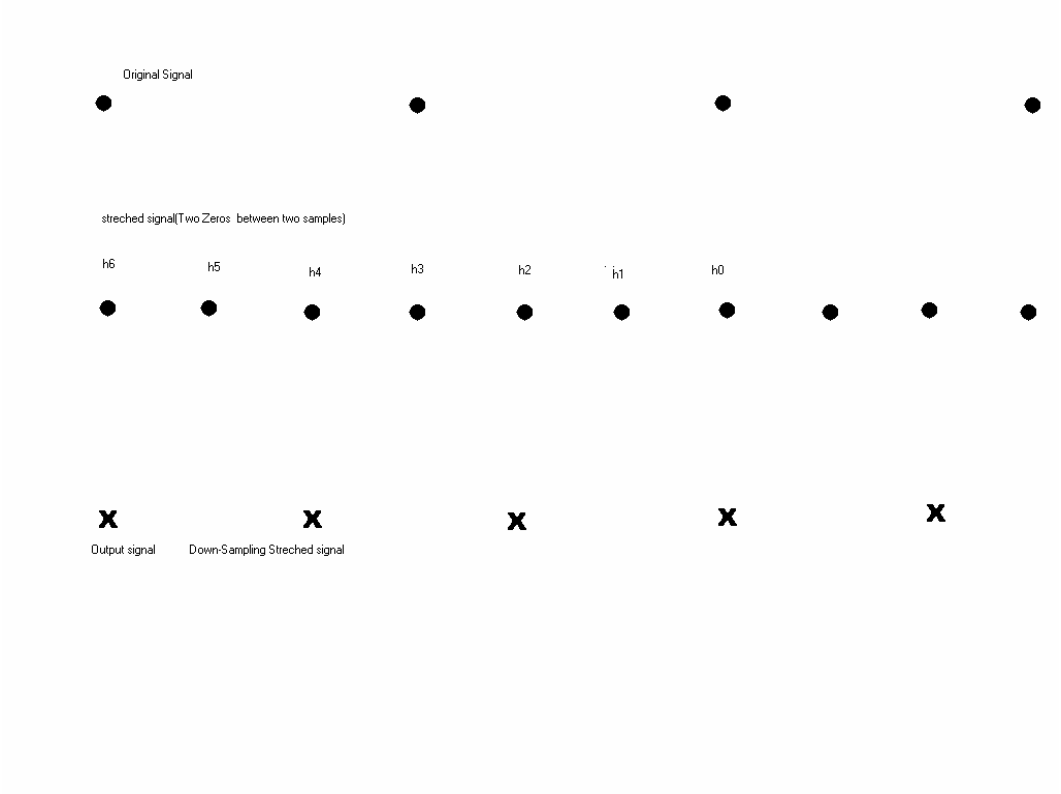
TIME -VARIANT Filters.
If p/q is the ratio we wanted …then for q inputs we want p outputs…

Getting polyphase filters from filter kernel we just downsample it at rate D=I
For example let us assume that we want to sample the input at rate 3/2

So when upsampler is followed by downsampler..
In upsampler we have filter banks of (h0,h3,h6..) (h1,h4,h7) (h2,h5,h8..) …..
So first non-zero output we get is from 1st bank….Second from 2nd bank and so on

When we down sample these stretched signal from upsampler we are interested in non-zero values of stretched signal only.
And since we want every second sample to be selected we design the filter using above filter banks …
So combininng effect will be arranging filter banks with gap of 2 as
(h0, h3,h6..) then (h2,h5,h8..)….. (h4,h7,h10..) and so on
But we need to take care of the previous coefficient of the filter also..
Assuming h(n) is causal i.e. h(n)=0 for n<0
We get the banks as
(0,h0, h3, h6..) then (0,h2, h5,h8…)then(h1,h4,h7…) and so on

Following diagram will best show this example

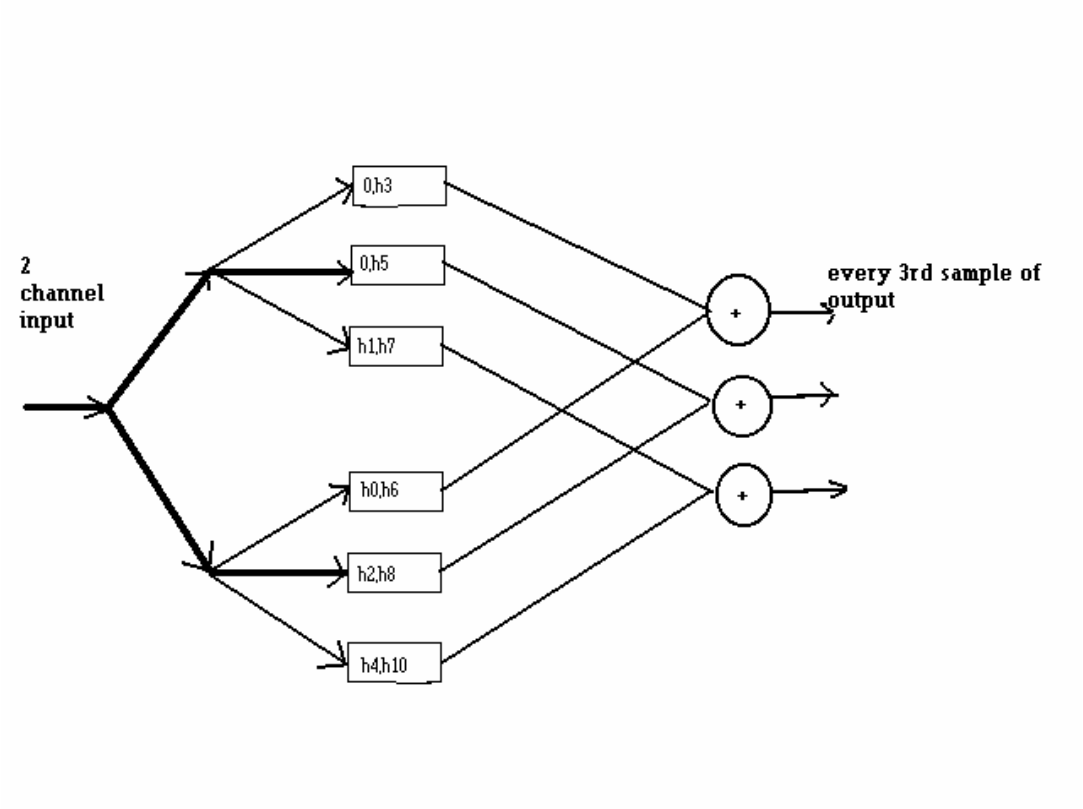


If u observe the diagram
first output sample(4th from left) can be obtained by using filter coefficients h(-3)=0,h0,h3,h6

Next one (3rd from left) can be obtained by h (-1)=0,h2, h5, h8
Similarly next one (2nd from left) can be obtained from h1, h4, h7,h10
Same sequence will get repeated after each 6 samples of stretched signal

Reduction we get by this is 1/(D*I).

Implementation of 3/2 sampler for 2 channel input can be shown as



Real-value upsampling
Suppose we want to upsample signal by 23.7.
In this case we can find p and q such that p/q best approximate to 23.7
But p and q may be too huge to design upsampler OR downsampler.
Hence method is not always useful.
Hence we first upsample signal by 23 and use approximation.
We can use linear filter to find the neighbor sample, which is close to 0.7 and use the "error" in approximation to predict next sample to approximate.

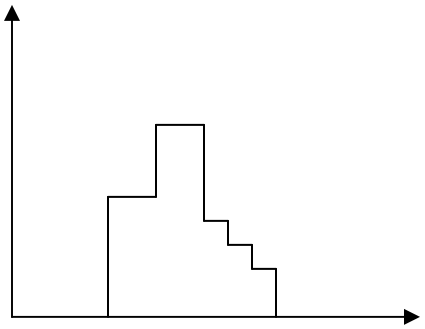
Applications of multirate sampling
1.speech scaling and speech shifters
2.compression/decompression algorithms
3.graphics

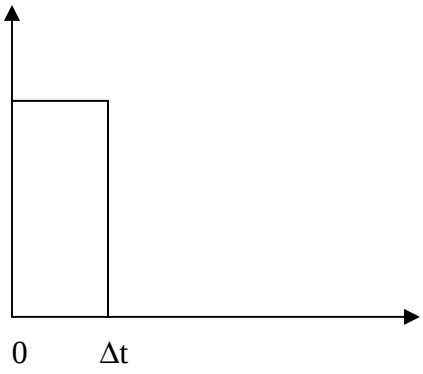
Filter Banks

Before we do filter banks some other stuff.

Sampling and reconstruction principles are used for designing devices like speakers and soundcard. They do reconstruction upto 8khz

You can use a hold for reconstruction



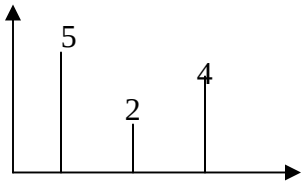


Time domain

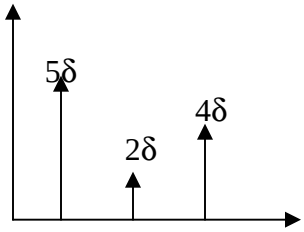
Frequency domain

Fig 1

To have this kind of a sample and hold effect
We need to first stretch and then convolve
Theoretical view:
If the signal is like below then the area under the curve is 0 as 5-2-4 have 0 area
This will result in 0 output



So what we need to do is to have unit area for 5-2-4.Hence convert them to dirac delta.

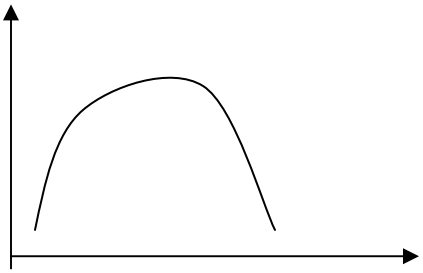


So the effect achieved as opposed to using just 5-2-4 are as follows

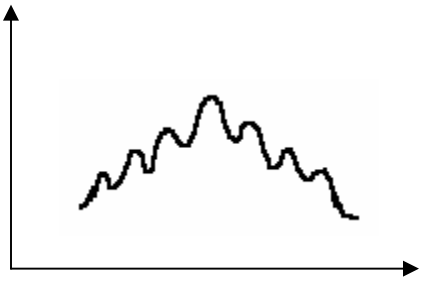
1. the first stretch caused one repeat in the $-\pi/2$ to $+\pi/2$ domain, but the dirac guys cause infinitely many repeats in the infinite freq domain
2. which will give you a fourier transform kind of a thing
3. so you have to use a low pass filter kernel to get the original freq response

Now the hold circuit shown in figure one is not so good to use
Mostly op-amps have this kind of a hold effect
So they use an analog lowpass filter after this

People who actually use this reconstruction technique are mainly LCD reconstruction and sometimes for huge T.V monitors
T.Vs ideally have to use the figure shown below for reconstruction ie this is how the phosphors have to get illuminated



But due to the R-G-B funda what they really do is use this for R-G-B each

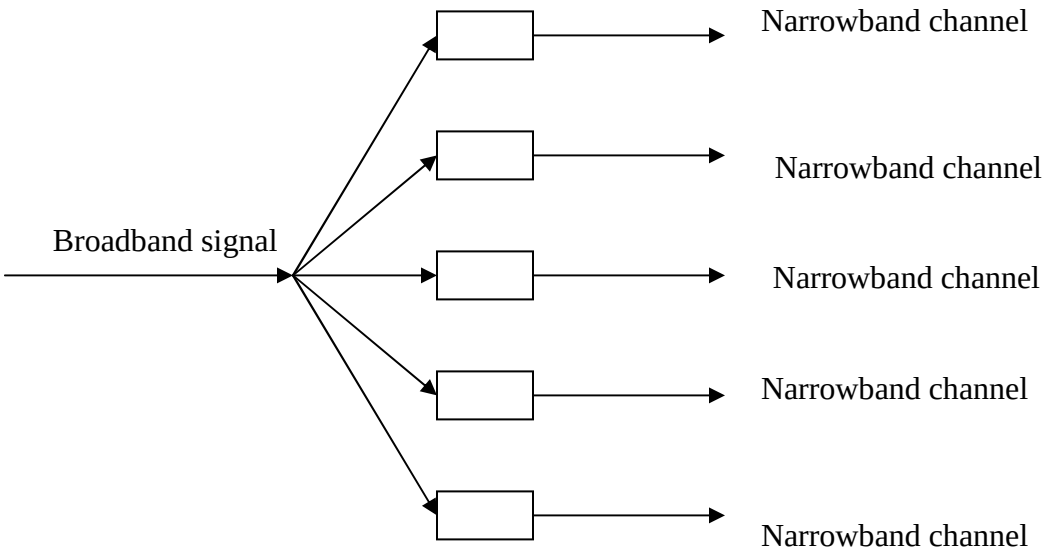


and due to persistence of vision it actually adds up to the above

.

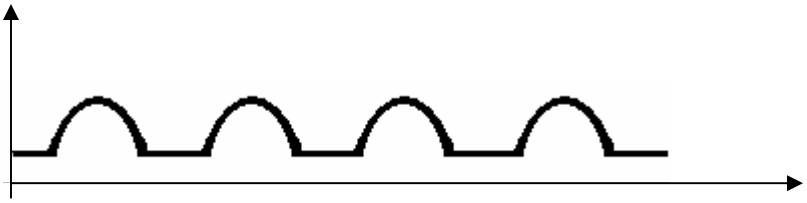
So coming back to what exactly is a **filter bank**

It looks something like this



Advantages:
This concept of filter banks is good for the following reasons
Since you can separate out the narrowband channels its good for analyzing or processing complex signals as you can definitely separate them out into many narrowband channels which are comparatively simple to process.

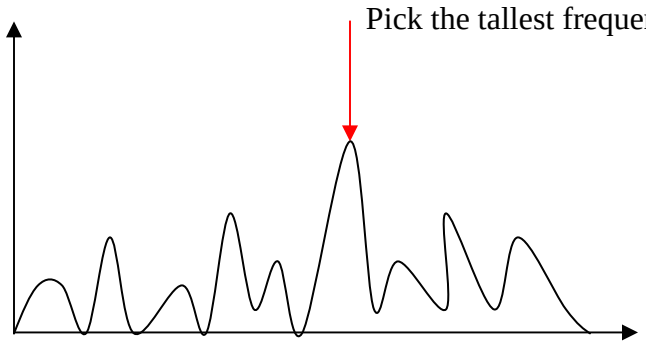
It can also be used for voice reconstruction or compression
Usually when you talk it's the air that passes the glottis, which produces some kind of vibrations in the air which look like this



The mouth cavity acts as a filter kernel for this signal.
This principal frequency and the overtones are actually your active frequencies.
Since you do windowing they appear as many.
These are the only frequencies that u have to actually care about
So what you can do is use a filter bank and separate out the active frequencies,

How do you find the active frequencies from the others, which are produced due to windowing

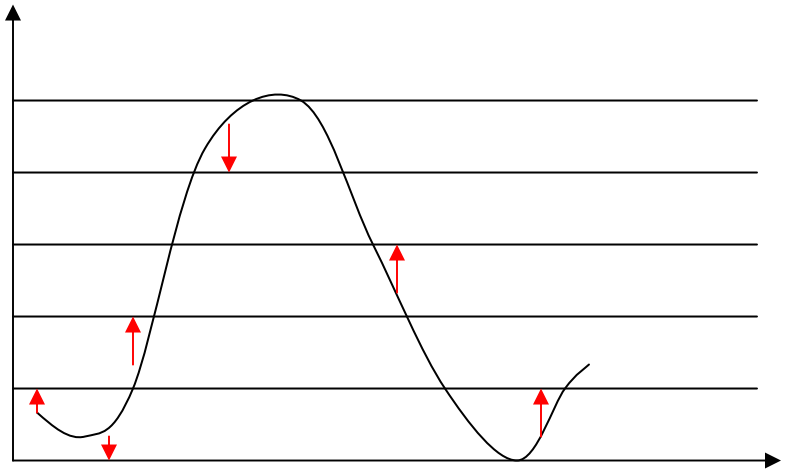
After the filtering you will get the frequency response that will look something like this



Compression

Using filter banks for compression of MP3 files

This is how quantization looks

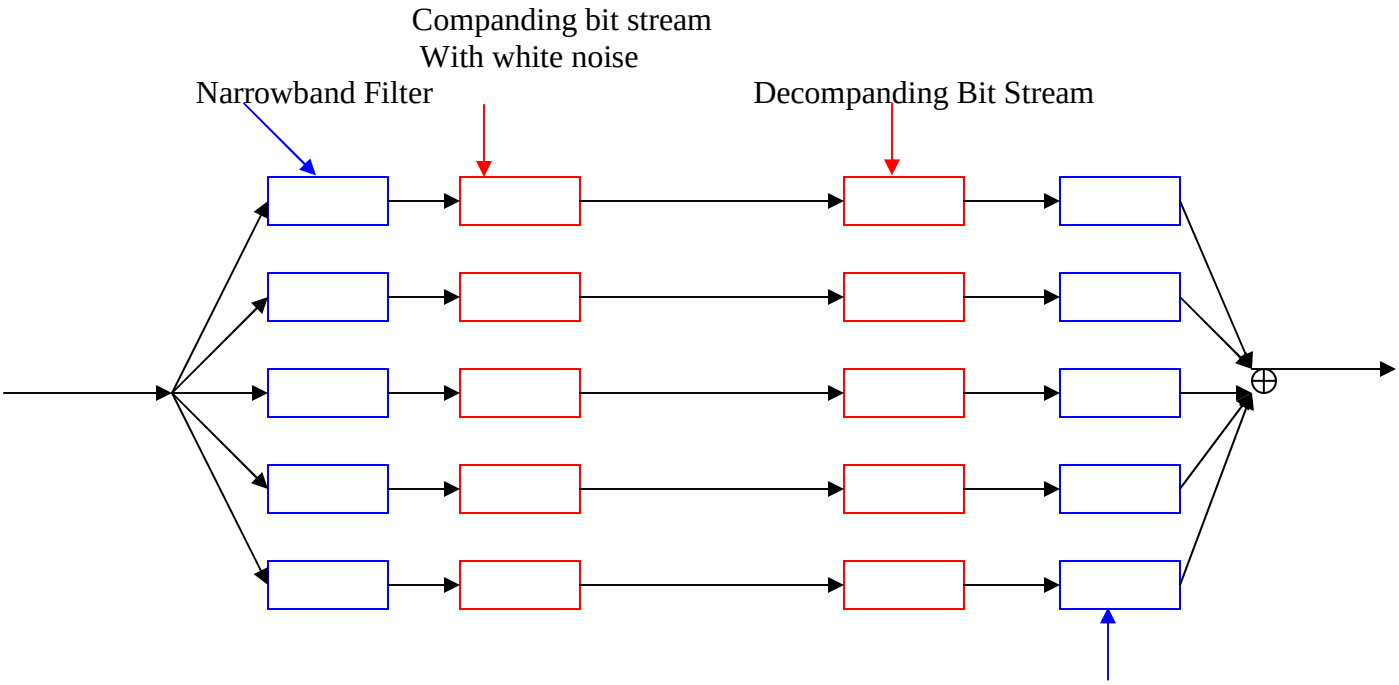


All the red arrows are the error or noise induced due to quantization. If the quantization levels are further apart (which means lower no. of bits), the noise will be more. But due to the persistence of hearing louder sound masks smaller sounds.

So if u are quantizing a loud sound u can use fewer levels, in spite of more quantization noise and thus have better compression.

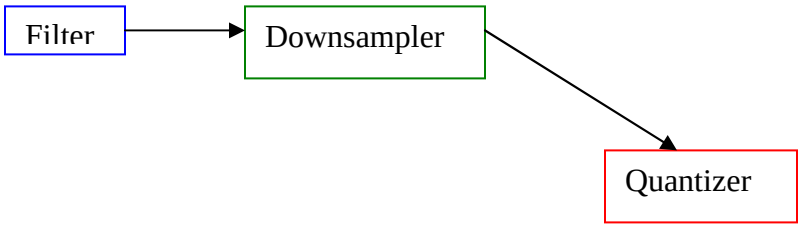
But in spite of the loud sound, you may not be able to mask all lower frequencies. Like an opera singer may not be able to mask the base voice of Amin Sayani.

So if you want to do some kind of compression you could use the following



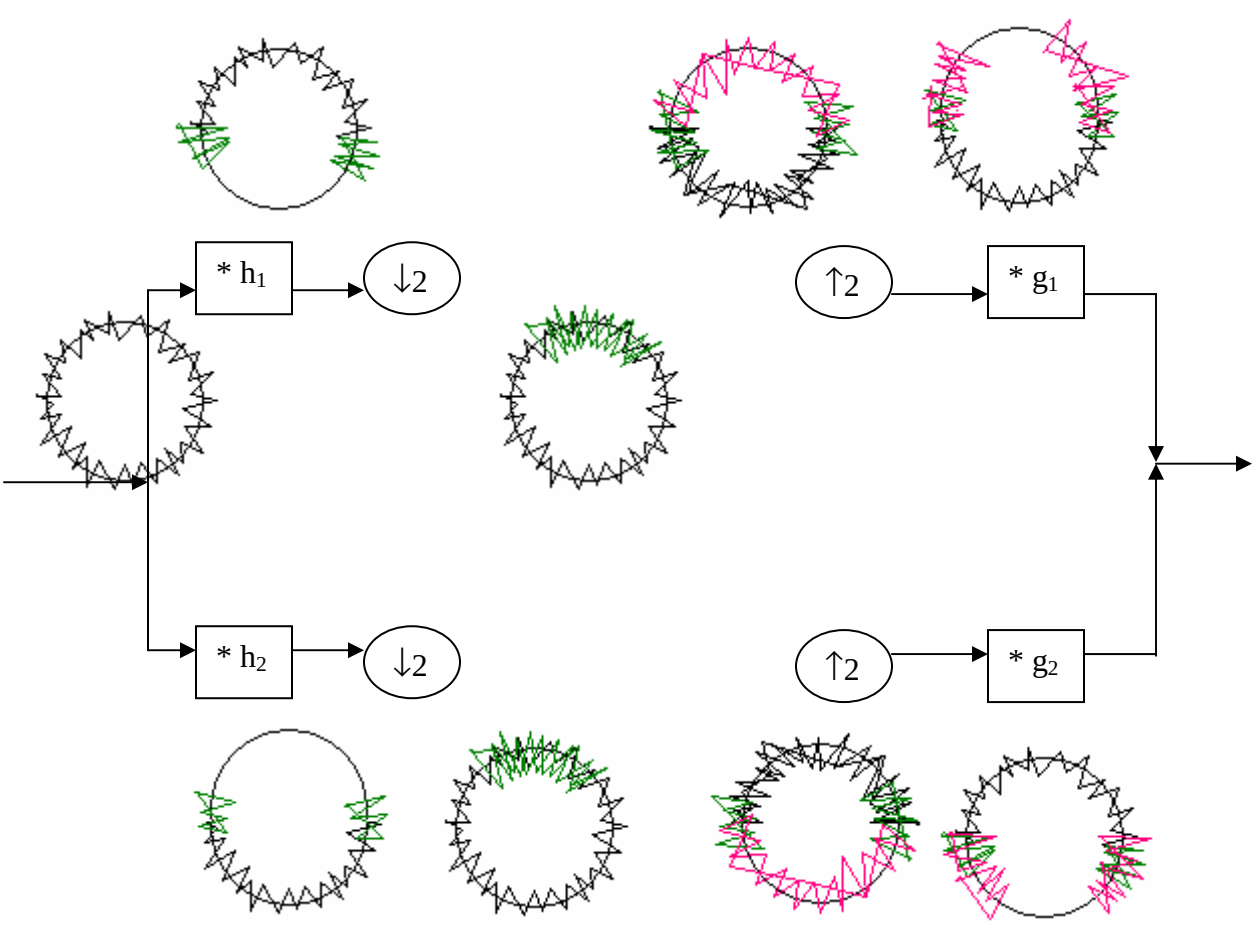
So this is how we are trying to achieve some compression.
But what we have ended up doing is just splitting the broadband signal into 12 narrowbands(in case of MP3 sound). This is not enough compression.

A better thing we could do is use a downsampler (where D = 12 in case of MP3)
Which would look like



Quadrature Mirror Filtering

Now lets take the example of a simple filter bank where we split the signal using a low-pass and high-pass filter



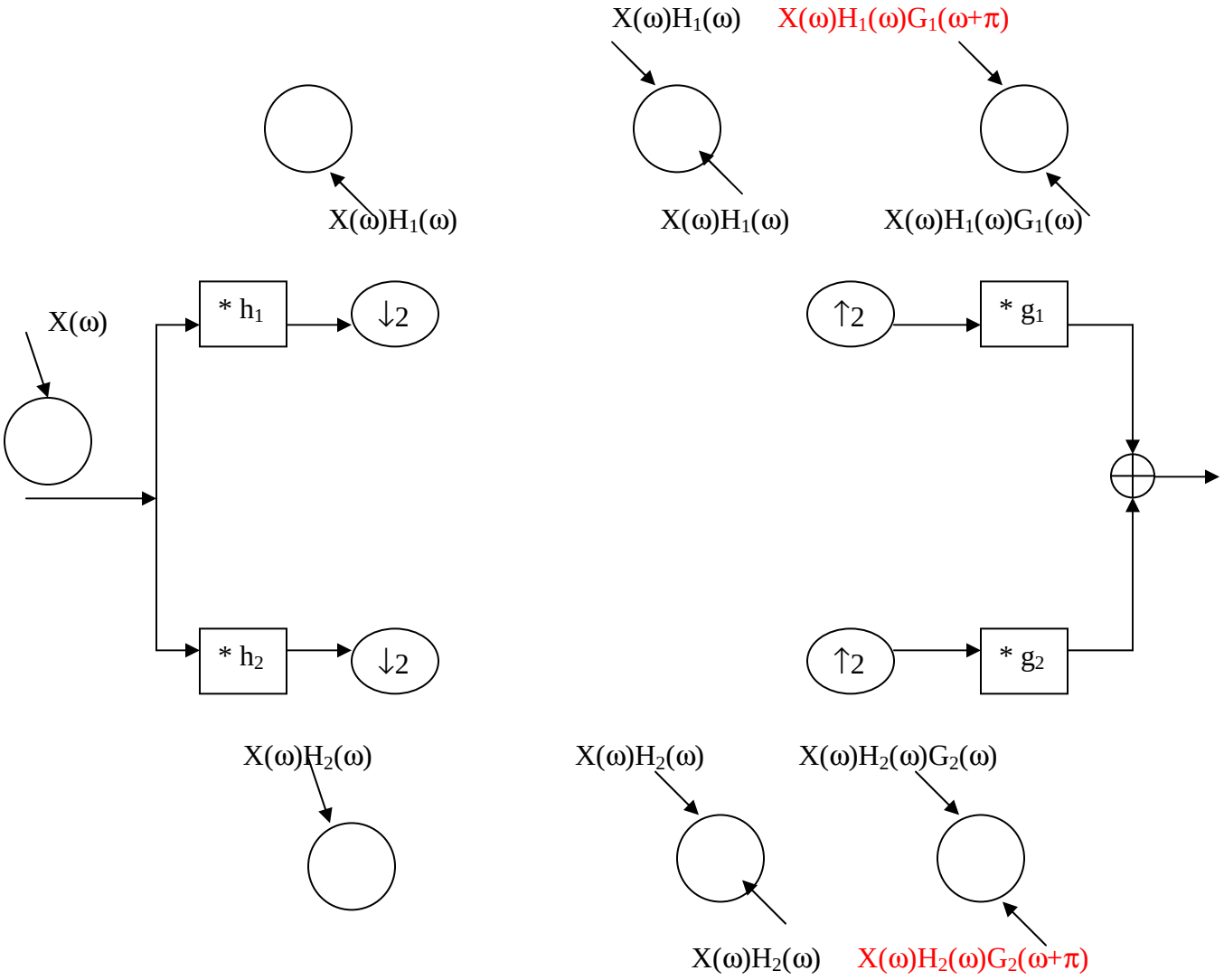
Now here we are really not concerned with what is happening in the middle layer (it could be a quantizer had we been thinking of compression, or anything)

Now initially the signal will get split into high and low frequencies. But as little of the higher frequencies pass through the low pass filter and the same for lower frequencies, there is a little smudging (indicated by the green part).

Next, both the downsamplers will cause a stretch in this smudged signal, to give the green part in the center.

Now we assume some process operating on this downsampled version of our signal, in the middle part, which will cause some smudging/distortion, and also after upsampling by 2, we will get a repeat in the signal spectrum. But here the green part,

becomes a part of the spectrum, and hence its aliases will actually not be recognizable in the signal (although the drawing shows the different colour). The g filters will remove the repeat, and the final spectrum is obtained.



The figure above shows the frequency response after every step of the low pass and high pass filters.

The red marked areas are the effects of aliasing(the green part which we want to eliminate). Just by itself we may not be able to design g_1 and g_2 such that the green part gets eliminated in the output, its because once you reach the right hand side, you really cant differentiate between the smudging and the actual signal.
However you can make use of the fact that the green part is some high frequency component in the upper half and some low frequency component in the lower half. Hence we can use the initial high pass filter to remove the high pass component in the RHS, and the same applies for the low pass filter.

We know that the final addition is this,

$X(\omega)H_2(\omega)G_2(\omega) + X(\omega)H_1(\omega)G_1(\omega) + X(\omega)H_2(\omega)G_2(\omega+\pi) + X(\omega)H_1(\omega)G_1(\omega+\pi)$
Out of that we need

$X(\omega)H_2(\omega)G_2(\omega+\pi) + X(\omega)H_1(\omega)G_1(\omega+\pi) = 0$

Because we want to cancel out the aliasing,

$H_2(\omega)G_2(\omega+\pi) + H_1(\omega)G_1(\omega+\pi) = 0$

$G_1(\omega+\pi) = H_2(\omega)$
$G_2(\omega+\pi) = - H_1(\omega)$

This thing is called Quadrature Phase Mirroring

Now if we have the low pass filter having frequency response $H_1(\omega)$
We know that for the Perfect Alias Canceling high pass filter we need the frequency response to just be shifted by π.
We could write its frequency response as $H_1(\omega+\pi)$

Now a shift of π in the frequency domain is equivalent to multiplying by $e^{j\pi t}$ in the time domain.
Hence every alternate sample will get inverted (because $e^{j\pi t}$ will effectively do $(-1)^n$)
Therefore h_2 is same as h_1 except for the fact that h_2 has every next sample of h_1 inverted.
And since g_1 and g_2 also depend on the h_1 and h_2 we only need to design one filter h.

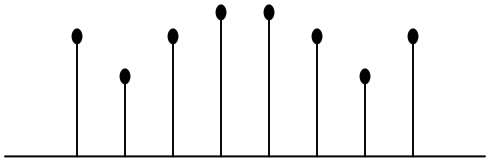
Each of these h_1,h_2,g_1 and g_2 need to be linear phase filters, because we cant afford to shift the phases of the narrowband signals in the filter banks, we finally have to synthesize them into one broadband again.
Therefore they have to be symmetric filters.

h_1

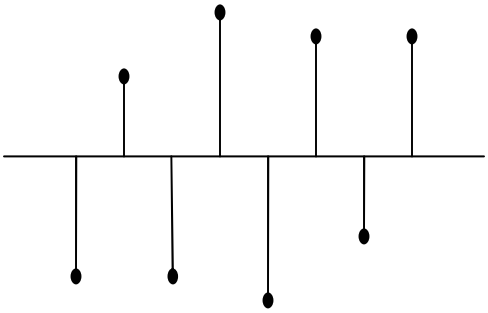
h_2(alternate samples inverted)

Even number of samples

symmetric

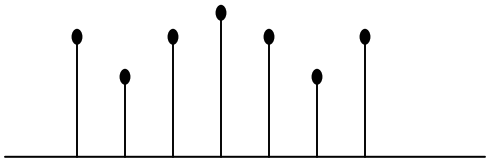


unsymmetric

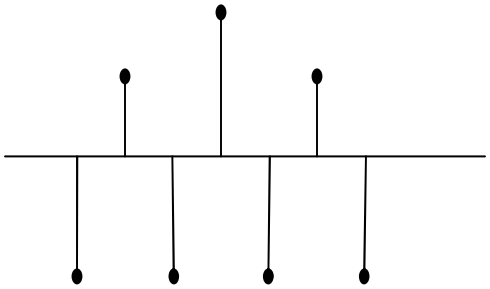


Odd no of samples

symmetric



symmetric



Therefore we need our linear phase filter h(based on which all h_1,h_2,g_1 and g_2 are designed) to have odd number of samples.

All this stuff combined together gives us our Quadrature Mirror filtering.

Lecture Date – 26/3/04

Statistical Signal Processing

We started by learning about Random Variables.

A random variable X is real valued and is something, which is unknown to us. It is the value the variable x would have in the future. It is possible that the event that gives x a value has taken place, but till we don't know the result, x remains to be the random variable X.

Random variables are like a secret (a real value) in a closed box, and we guess what this could be, without opening the box.

This guesswork is based on what we call a "belief system". E.g.: Suppose we wanted to guess the outcome of a fair coin toss. Some of us would say that there is an equal probability of getting a head or tail. We give the occurrence of head and tail both a probability of ½ each because we believe that for some reason.

Since we don't know the value of the random variable, it can take any one value from a given set of finite values. Such variables are known as Discrete Random Variables. With each such value a probability is associated (depending upon our belief system). This set of probabilities is the Probability Mass function (pmf).
E.g. If we throw a dice, 1,2,3,4,5 or 6 can occur each with lets say a probability of 1/6.
The pmf for this is {1/6, 1/6, 1/6, 1/6, 1/6, 1/6}.

It is denoted by $p_{\mathbf{X}}(x)$, the pmf of the random variable X. It is the probability that the value of X obtained on performance of the experiment is equal to x. 'x' is the argument of the function and is a variable and very different from X.
Some important properties of pmf

1. Since $p_{\mathbf{X}}(x)$ is probability, $0 \le p_{\mathbf{X}}(x) \le 1, \forall\, x$.
2. $\sum_{x} p_{\mathbf{X}}(x) = 1$. This true as the random variable will be assigned some value definitely.

Similarly if we have 2 events, E.g. throwing 2 dice. Then the pmf associated with this event (which, is made up of 2 events) is given by what is called the 'Joint pmf'. It is important to see that a Joint pmf contains more information then the individual pmfs.
Consider the example of throwing 2 dice. Suppose the outcome of this event was that the 1 dice always tries to match the other. If we looked at the die individually, we would think that all the values are equally probable and may not realize that they actually trying to be equal.

The joint pmf for this would be 2-D table, with values of dice X on 1 dimension and that of Y on the other.

$X \backslash Y$	1	2	3	4	5	6
1	1/6	0	0	0	0	0
2	0	1/6	0	0	0	0
3	0	0	1/6	0	0	0
4	0	0	0	1/6	0	0
5	0	0	0	0	1/6	0
6	0	0	0	0	0	1/6

Fig 1: Joint pmf of 2 die when they are thrown such that 1 always tries to match the other.

Like this we can have the joint pmf of 'n' random variables, which would be an N dimensional table. The joint pmf for 2 random variables can be written as

$p_{\mathbf{X,\,Y}}(x, y)$, the probability that X=x and Y=y.

In terms of closed boxes joint pmf means that we have 1 big box containing 2 boxes with X and Y inside them. Opening the big box means opening the 2 smaller boxes automatically and the joint pmf is the label on the big box.

Suppose, if we had to open only one of these boxes say X, then we can find the pmf for Y from the joint pmf as below. This is called the 'conditional pmf' and is read as the pmf of Y given X equal to 'x'

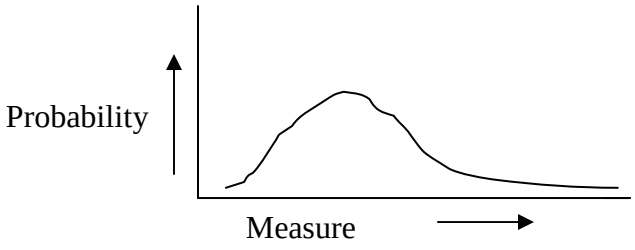
$$p_{\mathbf{Y|X=x}}(y) = \frac{p_{\mathbf{X,\,Y}}(x, y)}{\sum_{y} p_{\mathbf{X,\,Y}}(x, y)}$$

Suppose given the joint pmf of X,Y, we want to find the pmf of X. This is like saying we don't care about Y at all i.e. given the joint pmf we want to find the individual pmfs. This is called the 'Marginal pmf' and is found as

$$p_{\mathbf{X}}(x) = \sum_{i} p_{\mathbf{X,\,Y}}(x, y_i)$$

Pmf is possible in the discrete domain, but for continuous domain, the random variable can take a value from infinite values. Thus for continuous valued random variables we define the Probability density function (pdf). It is the ratio of the probability of the domain to the measure of that domain and at a point it is the limit of this ratio as the measure tends to zero.

E.g. We have a dart board, and want the probability of hitting a point on the board. In the 1-dimensional case, the measure would be length and the pdf could be shown as below.



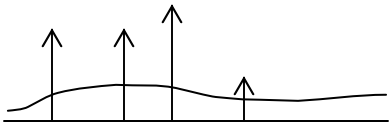
It is denoted as $f_X(x)$.

The pdf satisfies the following properties

1. $f_X(x) > 0$, $\forall\ x$.
2. The area under the curve must be 1 i.e

$$\int_x f_X(x)\, dx = 1$$

Suppose we have lumped and continuous probabilities together like in the regulator of a fan. We have lumped probability at the standard fan speeds, but continuous between them. This can be shown by dirac delta functions in the pdf as shown below.



Expected value of a random variable X

This is the expected value of - the average value of X over all the possible futures. In other words, if we could perform an experiment 'n' times to find the value of X, then the mean of all those values would be the expected value of the random variable X.
Mathematically, $E[X]$ the expected value of X is

$$E[X] = \sum_x p_X(x)\, x \qquad \textit{or}$$

$$E[X] = \int_x f_X(x)\, x\, dx \qquad (f_X(x)\, dx \text{ is the probability of } x)$$

Linearity of expectations

$$E[X+Y] = E[X] + E[Y]$$

Proof:

$$E[X+Y] = \int_x \int_y (x+y)\, f_{X,Y}(x,y)\, dy\, dx$$

$$= \int_x \int_y x\, f_{X,Y}(x,y)\, dy\, dx + \int_x \int_y y\, f_{X,Y}(x,y)\, dy\, dx$$

$$= \int_x x \int_y (f_{X,Y}(x,y)\, dy)\, dx + \int_y y \int_x (f_{X,Y}(x,y)\, dx)\, dy$$

$$= \int_x x\, f_X(x)\, dx + \int_y y\, f_Y(y)\, dy$$

$$= E[X] + E[Y]$$

Also, if we have a function g(X) (i.e. a function which depends only on X), then

$$E[g(X)] = \sum_x p_X(x)\, g(x)$$

This can be seen from the fact that,

$$p_{g(X)}(g(x)) = \sum_{x:\, g(x)} p_X(x)$$

Estimators of X

We want to predict the value of X, such that the expected value of the mean square error is minimum i.e. $E[(X-a)^2]$ is minimum, where a is the estimated value of X.

$E[(X-a)^2] = E[X^2] - 2a\, E[X] + a^2$

Differentiating w.r.t to a

– $2\,E[X] + 2a = 0$

$a = E[X]$

The value of 'a' comes out to be $E[X]$. This is the best estimator for X, which is the expected value of X itself.

Suppose we had the Joint pdf of X & Y. Now, if we open the box for Y, we can predict the value of X given Y=y.

From above we observe 'a' will be,

$a = E[X \mid Y = y]$ (the centroid of the row with Y=y)

Thus 'a' is a function of y**.** This again is the best estimator of X given Y, and 'a' can be any function of y such that the cost is minimum.

If we had to restrict 'a' to be only a linear function of Y, then we would be basically trying to minimize $E[(X - aY)^2]$ and 'a' comes out to be

$$a = \frac{E[XY]}{E[Y^2]}$$

We notice that this expression is the exactly the same as that for – the scalar 'a', when two vectors y and x are given and we want to fit them together (we would scale one of them by a the scalar 'a').

We realize that random variables are vectors. They can be scaled and added just like any two vectors and the above expression enforces this fact.

What remains to be seen is why would we want a linear estimation for X as opposed to the best estimation.

1. One reason is that it is mathematically simpler.
2. A lot less data is required for the linear estimate than for the best estimate. For the linear estimate we just need E[XY] i.e. is covariance of X & Y (if X , Y are zero mean) whereas for the best estimate we need the entire joint pdf.
3. In a lot of cases, the linear estimate is the best estimate, so basically it is good enough.

LINEAR PREDICTION TECHNIQUES

27 March 2004

Summary of the lecture

We continued the theory of joint probability function, and understood what is meant by covariance of two random variables.

Then, we discussed how to express a random variable X as linear function of another random variable Y, minimizing the error in the L2 sense.

Then we extended this to a more unrestricted case where we expressed X as an affine function of Y. This yields an estimation technique called Wiener's method. Lastly, we extended this to a case where we express a random variable X in terms of N other random variables

Recapitulation of the previous lecture

What we saw in the last lecture is that associated with every random variable, is a belief structure, called its probability mask function or the probability density function.

When there is more than one variable their combined random variable is described in terms of the joint probability distribution.

Then we saw the concept of marginalization and conditional probability distribution.

Finally, we saw what is expected value of a random variable. We also found that if we were to predict a value for outcome of a random variable X, such that the error (X - guessVal) is minimum in the mean square or L2 sense, and then the guessVal should be the expected value of X.

If we restrict ourselves to X being estimated as a linear function "aY" of another random variable Y, then best value of 'a' = $E[XY]/E[Y^2]$

Covariance of Random Variables

What is the meaning of the symbol "E [XY]" that we saw last time ?

XY is a random variable such that it takes the outcome of variable X and multiplies it to the outcome of variable Y, and that is the value of random variable XY.

The expected value of the covariance of X and Y, namely E [XY] cannot be expressed in terms of E [X] and E [Y]. Let us see why.

Consider the examples given below

1. A coin is tossed. Suppose X is the random variable such that the value of X is +1 when the coin lands heads up, and has value -1 when it lands tails up.

Suppose Y is the same coin toss. Assuming a fair coin, the joint probability function is as shown below in figure 1.

$E[XY] = 0.5 * (1)(1) + 0.5 * (-1)(-1) = 1$ But, E[X] = 0 E[Y] = 0

2. Consider X to be outcome of a fair coin toss, same as X above. But Y is another coin toss, this time. So both the random variables are independent of each other.

The joint probability function would be,

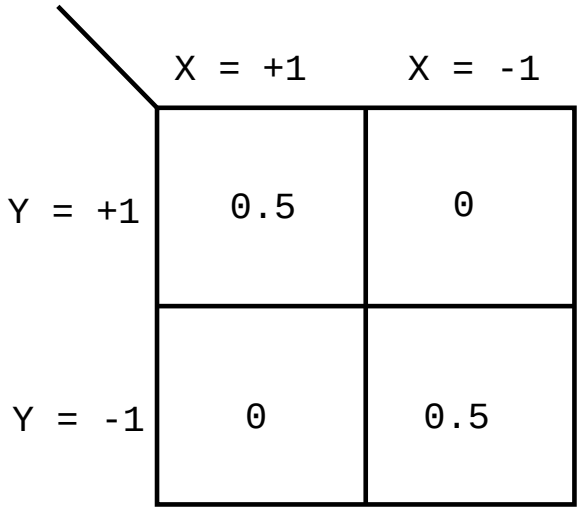


Figure 1: Joint Probablility distribution of two zero mean dependent random variables

$E[XY] = 0.5 * (1) + 0.5 * (-1) = 0$ But, E[X] = 0 E[Y] = 0

So, these two examples show that the E [XY] cannot be expressed in terms of E [X] and E[Y], as for the same values of E [X] and E [Y] different values of E [XY] are obtained.

In fact, random variables can be uncorrelated even if they are not independent. So if X and Y are independent, then surely they are uncorrelated. But, converse is not true, i.e. if X and Y are uncorrelated it does not necessarily imply their independence.

Mathematically,

$E[XY] = \int_x \int_y F_{X,Y}(x,y)x.y.dy.dx$

Linear Estimation Technique

If we were to estimate X using a vector of real numbers 'a', then 'a' is the vector that contains the centroid of each row.For each Y=y , the row would be some values. The centroid of , say row Y=+2,is the best estimate a_1 is the best estimate,i.e. $a_1 = E[X|Y = +2], a_2 = E[X|Y = +1]$....

Refer to Figure 3.

The curved line 'a' is the vector that minimizes the function (X-a).

Suppose we have to estimate the random variable X using a variable Y. In this technique we try to find out real numbers 'a' and 'b' such that the error function "X - aY - b " is minimum in the L2 sense.

We recall that if X is independent of Y, then the scale 'a' = 0, and b is simply the expected value of X. This is the first case discussed in the previous lecture, when the error function is (X - b).

But in general X and Y may not be independent.

Suppose X and Y are zero mean random variables. Zero mean random variables are ones whose expected values are equal to zero. Then, we can prove that 'b' is 0. Why this is so is simple to see. We know that X = a Y + b. So E [X] = a E[Y] + b, If E [X] = E[Y] =0, then b =0.

This implies that line aY + b passes through (0,0).

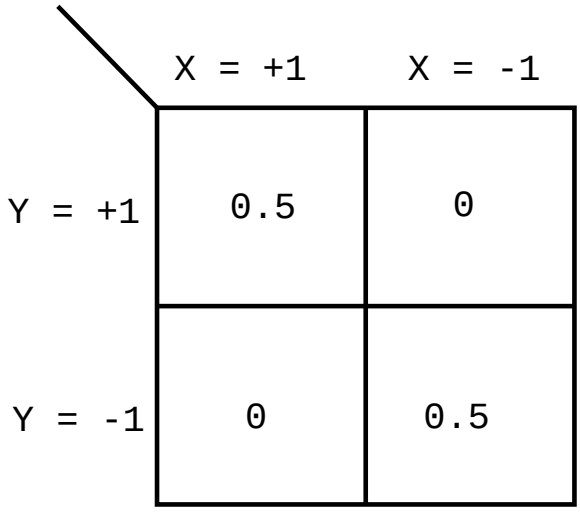


Figure 2: Joint Probablility distribution of two zero mean independent random variables

Mathematically,

$E[(X - aY - b)^2]$

$= E[X^2 + a^2Y^2 + b^2 - 2aXY - 2bX - 2baY]$

$= E[X^2] + E[a^2Y^2] + E[b^2] - 2E[aXY] - 2bE[X] - 2bE[aY]$

Taking partial derivatives w.r.t a and b, we get

$a = E[XY]/E[Y^2]$

as, $E[X] = E[Y] = 0$

and

$b = E[X] - bE[Y] = 0$

Now, suppose that X and Y are not zero mean random variables, or E[X] and E[Y] are not zero. The line 'X = aY +b' is say somewhere , (in Fig 4, as line 1).

We can prove that line 'X= aY +b ' passes through (E[X] , E[Y]).

Let us make the random variables X and Y zero mean. This means we arithmetically shift the outcomes of X and Y by -E[X] and -E[Y] respectively.Now, that line stays where it is , but the labels of each row and column have have changed. So, the co-ordinate system has shifted. Now, we know that this line 'X = aY + b' has to pass through (0,0) in the shifted co-ordinate system.This corresponds to line 2 in Fig 4. The point now called (0,0) was called (E[X] , E[Y]) earlier.So that line passes through (E[X],E[Y]) in the "unshifted" system.

So, we know that '$X = aY + b$' always passes through ($E[X]$, $E[Y]$).

$E[X] = aE[Y] + b$ This implies that $b = E[X] - aE[Y]$

To find 'a', We try to minimize the mean square of error function E $[(X - aY - b)^2]$. If we do so we would find that we obtain a similar result as in the case of zeromean random variables.

$a = E[(X - E[X])(Y - E[Y])]/E[(Y - E[Y])^2]$

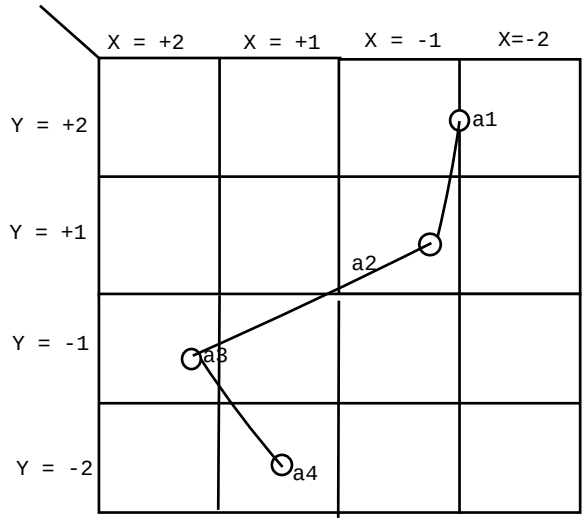


Figure 3: Expected Value curve, where expected values are a vector of real numbers

and the constant shift is,

$b = E[X] - aE[Y]$

This is just the same as for zero mean random variable case but for that both random variables are shifted by their expected values.

This method is called as Wiener's method of estimation.

Why is the linear estimator used?

1. Its computationally faster to compute and is easier to apply. This is so because we just have to calculate scalars 'a' and 'b' rather than functions. Once we have a, b it is just a multiplication and an addition to get a vector element of estimated X, rather than a table lookup.

2. The estimation can be done with much lesser data, i.e. only $E[XY]$ and $E[Y^2]$.

3. Many times it so happens that even the unrestrained vector is an affine function of Y. So in such cases, we might as well save computation time by estimating using linear or affine estimation.

Y estimated as a linear combination of N other random variables

Suppose we want to estimate the random variable Y using a linear combination of random variables $X_1, X_2..X_N$. This can be viewed as estimating the value that will come out of the unopened box , given the outcomes of N other random variables boxes.

So,

$Y = a_1.X_1 + a_2.X_2 + ...a_N.X_N$

Again, in this case we will try to see what values of a's will minimize the mean square estimation error. The error function is

$Y - A^T X$

where A is a vector of real nos. ,Y is a random variable and X is a vector of random variables.

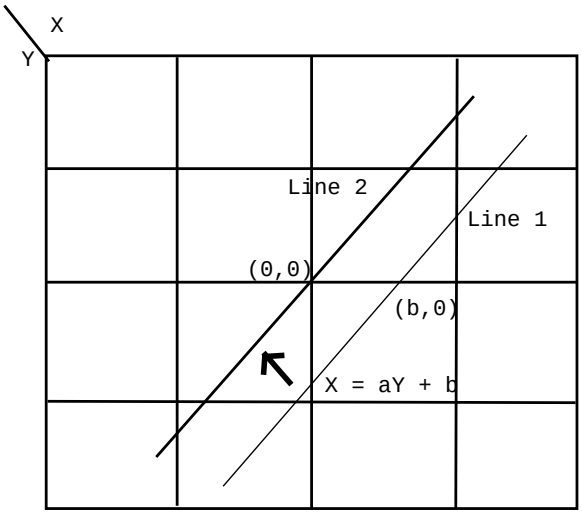


Figure 4: Expected Value curve, where expected values are "aY + b"

$$A = (\, a_1 \quad a_2 \quad . \quad . \quad . \,)$$

The expected Value of

E $[(Y - A^T X)^2]$

is to be minimized

$E[(Y - A^T X)^2]$

$= E[Y^2 - 2YA^T X + A^T X A^T X]$

By Linearity of expectation,

$= E[Y^2] - 2A^T E[XY] + E[A^T X A^T X]$

Since $A^T X$

$$= \begin{pmatrix} a_1 \\ a_2 \\ \vdots \end{pmatrix} (\, X_1 \quad X_2 \quad . \quad . \quad . \,)$$

$$= \begin{pmatrix} X_1 \\ X_2 \\ \vdots \end{pmatrix} (\, a_1 \quad a_2 \quad . \quad . \quad . \,)$$

$= X^T A$

So, using this to modify the last term of the above equation,

$= E[Y^2] - 2A^T E[XY] + E[A^T X X^T A]$

$= E[Y^2] - 2A^T E[XY] + A^T E[X X^T] A$

This is a familiar quadratic in A. The solution of this is,

$A = E[X X^T]^{-1} E[XY]$

This is again very similar to the result we obtained in the 2nd lecture, except that now we have shown the same concept in terms of random variables, i.e. in the vector space defined by random variables X ,Y.

The application of this result will be seen in the coming lectures.

Random Processes

30th March 2004

Summary of the lecture

We defined the term *random process*.

We said that we want to predict it, using something called a Wiener filter.

We defined a class of random processes, called stationary random processes, that are especially suited to Wiener prediction.

We defined filtering of a random process, and discussed what happens to a random process when it is filtered.

We defined the power spectrum of a stationary random process.

Random Processes

A random process is a sequence of random variables.

Usually we look at this sequence as time-ordered. Suppose there is this guy with nothing else to do, tossing a coin every second. Associated with each time t, there is a random variable—the result of the coin toss. Before he starts tossing coins all the results are unknown and therefore random variables. At time t_n, the n^{th} random variable is "opened".

Because there are infinitely many random variables in a random process, the PDF is not defined as an infinite dimensional table, but by defining the PDF of every finite subset of the sequence. So there are (infinitely many) finite dimensional tables now.

Prediction of Random Processes and Stationarity

We are interested in random processes because we can use them to model things like speech, and then use this model for compression.

The idea of compression is to predict the next outcome of a random process, and then transmit only the error between the actual outcome and the prediction. If our prediction is good, the error will have much less energy, so fewer bits have to be transmitted.

One method of prediction, called *Wiener prediction*, is to predict an outcome by a linear combination of the previous n outcomes. The coefficients in this linear combination must be constant numbers—we can't change them for every prediction.

Consider what Wiener prediction does:

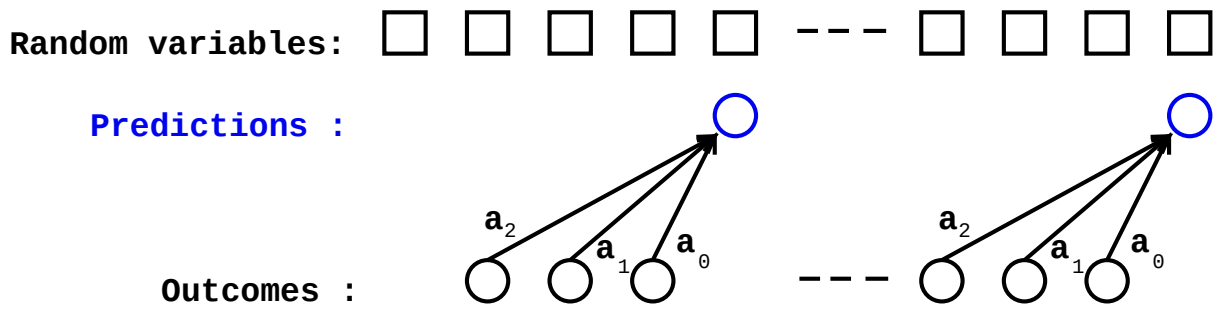


Figure 1: Wiener Prediction

Shown above are two predictions. Each uses the previous three outcomes, weighted by the **same** numbers a_0, a_1, and a_2.

What kind of processes are suitable for this kind of prediction ?

The a's depend on the joint PDF of the random variable being predicted and the one being used for the prediction. So the suitable processes are ones where any random variable X_n has the same joint PDF with the previous three random variables X_{n-1}, X_{n-2}, and X_{n-3}.

Such a process is called *stationary of order 3*. In general you can have a stationary process of order L, where the joint PDF of X_n and X_{n-i} is same for all n for each $i <= L$.

If a process is stationary of order L it is stationary of all orders less than L. A process that is stationary of all orders is called a *strict-sense stationary process*.

Strict-sense stationarity is a very restrictive. Not many real life random processes are strict-sense stationary. Instead, a wider restriction can be made on the random process. Instead of saying that the entire joint PDF of X_n and X_{n-i} should be same for all n, we will merely say that $E[X_n X_{n-i}]$ should be same for all n.

Such a process, where $E[X_n X_{n-i}]$ depends only i, is called a *wide-sense stationary process*.

Henceforth when we say "stationary process", we will mean "wide-sense stationary process".

The auto-correlation function

Define

$$\gamma_{xx}(i) = E[X_0 X_i]$$

Properties worth noting:

1. $\gamma_{xx}(i) = \gamma_{xx}(-i)$. (Because $E[X_0 X_i] = E[X_i X_0]$.)

2. $\gamma_{xx}(0)$ is the variance of X_0 (or any X_i, because they all have equal variance).

3. $\gamma_{xx}(0) \geq \gamma_{xx}(i), \forall i$. (Because nothing can be more correlated with a random variable than that random variable itself.)

4. If $\gamma_{xx}(0) = \gamma_{xx}(k)$, then γ_{xx} is periodic with period with k.

Filtering of Random Processes

Take a random process x, convolve it with a filter f, and you get another random process y. Going to the analogy of closed boxes, this new random process consists of boxes which when opened cause some boxes in the original process to be opened and added up using the gather kernel.

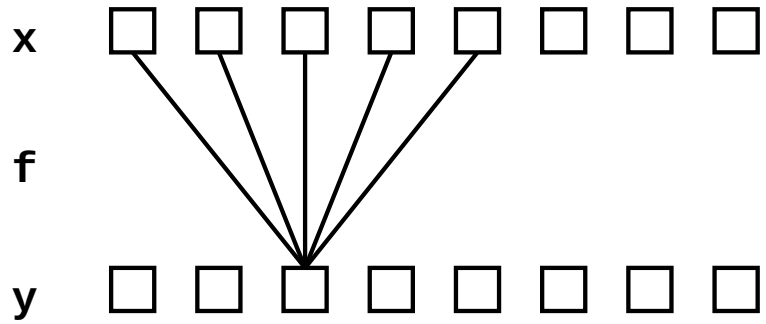


Figure 2: Filtering a random process makes another random process

If x is strict-sense stationary, then so is y.

Define a *white noise process* as one in which all the variables are uncorrelated. The autocorrelation function looks like the Kronecker delta:

$$\gamma_{xx}(i) = \delta_0(i)$$

If x is white noise, what is kind of process is y ?

For example if f is of order 3:

$$Y_0 = f_0 X_{-1} + f_1 X_0 + f_2 X_1$$

Since the $E[X_i X_j] = 0$ for $i \neq j$,

$$E[Y_0^2] = E[f_0^2 X_{-1}^2 + f_1^2 X_0^2 + f_2^2 X_1^2]$$

Therefore:

$$\gamma_{yy}(0) = \sum_i f_i^2$$

Next,

$$E[Y_0 Y_1] = E[(f_0 X_{-1} + f_1 X_0 + f_2 X_1)(f_0 X_0 + f_1 X_1 + f_2 X_2)] = f_0 f_1 + f_1 f_2$$

Therefore:

$$\gamma_{yy}(1) = \sum_i f_i f_{i+1}$$

Similarly,

$$\gamma_{yy}(j) = \sum_i f_i f_{i+j} = r_{ff}(j)$$

And so y is wide-sense stationary.

Now consider what happens if x is wide-sense stationary. We will prove that y is wide-sense stationary and find a relation between γ_{yy}, γ_{xx}, and r_{ff}.

Consider an example f of order 3.

Then:

$$E[Y_0^2] = E[(f_0 X_{-1} + f_1 X_0 + f_2 X_1)(f_0 X_{-1} + f_1 X_0 + f_2 X_1)]$$

$$\gamma_{yy}(0) = E[Y_0^2] = \left(\gamma_{xx}(0) \sum_i f_i^2\right) + \left(\gamma_{xx}(1) \sum_i f_i f_{i+1}\right) + \left(\gamma_{xx}(-1) \sum_i f_i f_{i+1}\right) + ...$$

Next,

$$E[Y_0 Y_1] = E[(f_0 X_{-1} + f_1 X_0 + f_2 X_1)(f_0 X_0 + f_1 X_1 + f_2 X_2)]$$

$$\gamma_{yy}(1) = E[Y_0 Y_1] = \left(\gamma_{xx}(1) \sum_i f_i^2\right) + \left(\gamma_{xx}(0) \sum_i f_i f_{i+1}\right) + ...$$

We can see that :

$$\boxed{\gamma_{yy} = \gamma_{xx} * r_{ff}}$$

In the Fourier domain this is:

$$\Gamma_{yy} = \Gamma_{xx} \circ R_{ff} \qquad \text{where } \circ \text{ denotes pointwise multiplication.}$$

But since $r_{ff} = f \star \overleftarrow{f}$, so $R_{ff} = F \circ \widehat{F} = |F|^2$. So,

$$\Gamma_{yy} = \Gamma_{xx} \circ |F|^2$$

Γ_{xx} is called the *power spectrum* of x.

Lecture 14: Stationarity and the Innovations Process
Date: 30th March 2004
Scribed by Aditya Thakur

Why is noise 'white'?

Most noise is from 'seemingly' independent processes.
eg. The 'hisss' sound caused due to turbulent flow of air seems to be made by independent particles though they are interacting with each other. The same can be said about the weather.
So basically two samples may be chaotically related with each other, giving us the impression that they are actually doing something 'new' every time (i.e. random).

Another important source of noise is *Electron noise*.
There are 2 basic types of e- noise:

1. Ensemble noise: the buzz of the e-
2. Shock Noise[1]: which is a random point process

Why does some noise get colored?

Basically, noise gets coloured due to filtering (as we saw in the previous lecture). So what sort of filters are we talking about here?
Let us look at a *signal passing through the conductor*...

- We have the white noise due to e-
- Then you have the distributive capacitance which acts as a high/ low pass filter depending on current/voltage.
- And of course, you have the induction wall interference[2] (multipath conduction) which causes symbol distortion (overtaking of symbols). We saw this with respect to fibre optic cables, where different wavelengths used to take different paths and suffer different reflections on the walls of the cable.

So these filters acting on the white noise produce enough correlation to give the noise 'colour'.

Innovations!

What we will show now is that given a stationary random process Y having correlation γ_{YY}, we can get a causal, stable filter f such $\gamma_{YY} = r_{ff}$.
What this implies is that we can take a white noise process, filter it with f to give it the same colour as Y.

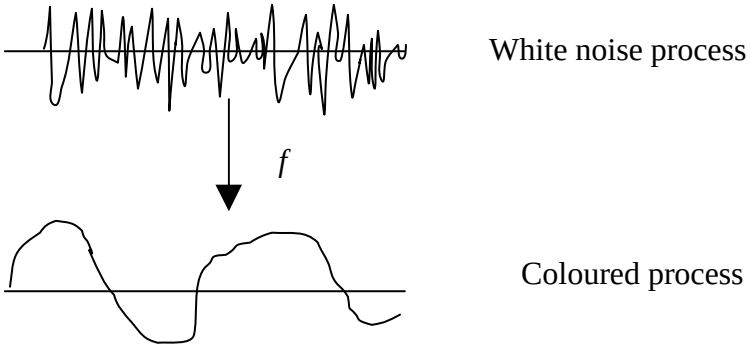


Figure1: A white noise process when filtered gives the coloured process

If f^{-1} exists, passing Y through f^{-1} we get an uncorrelated process Q, which will have the same information but has lesser energy. By having the same information, we mean that we can get Y back from the Q.

So how do we filter a stationary random process Y to get an uncorrelated process?

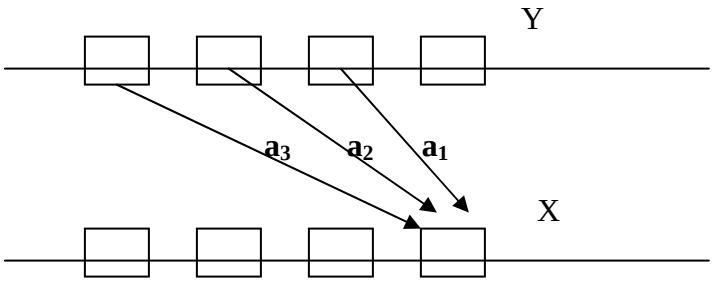


Figure2: Linear prediction of X using Y. $a_1, a_2, a_3, \ldots$ **are the filter coefficients.**

Well, if we do linear prediction, then the error we get is orthogonal to the signal used to estimate it. Lets use this fact and see what we can do:

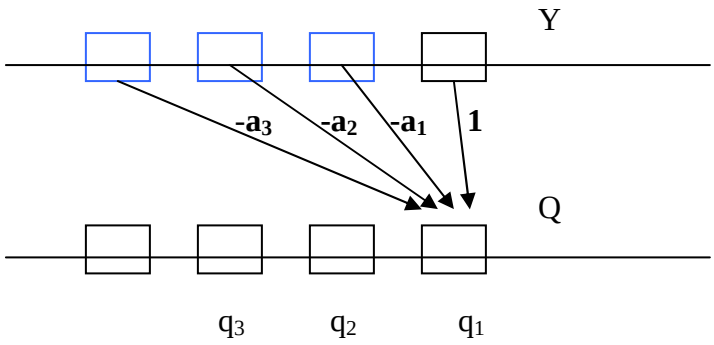


Figure3: The filter which gives the error corresponding to the optimal linear prediction

Looking at the above figure, we see that

1. q_1 is orthogonal to all the blue ys and thus to all vectors in the subspace of the blue ys..
2. q_2 is a linearly dependent on the same blue ys. And hence lie in the subspace of the blue ys.

From 1 and 2, we see that *q_1 is orthogonal to q_2* (and similarly to q_3, q_4, …).

Tadaa! We have Q, our uncorrelated process got by filtering Y. Q is often called the **innovations process** generating Y.

The filter $a_1, a_2, a_3, \ldots$ is called the **Wiener filter**, and is the optimal linear prediction filter.
The filter $1, a_1, a_2, a_3, \ldots$ is[3] the **Whitening filter**, and is the *prediction error filter*.
The inverse filter would be called the **Colouring filter**.

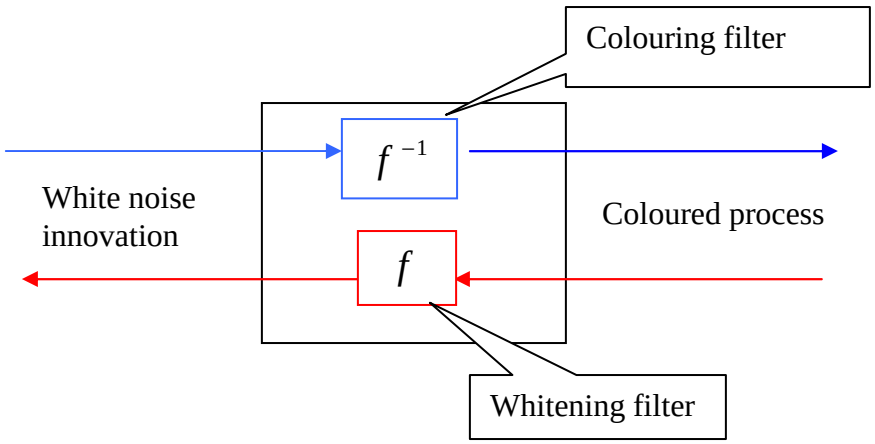


Figure4: Innovations representation of WSS process

Now what we have to show is that this Q exists for all stationary random processes i.e. the wiener filter exists, is causal and stable.

Before we go on to that, lets look at the **applications of the existence of such a whitening filter f .**
Suppose you have a speech signal to be transmitted. You pass it through a whitening filter to get the uncorrelated signal. Now you only send the filter coefficients across.
At the other end, you take an 'equivalent' white noise process, pass it thorigh f^{-1} to get the original speech signal. This may sound (no pun intended) a little fuzzy, but what they would do is model the innovations process of all speech signals in some way and use it; also the human ear is quite tolerant so the reconstructed speech signal would pass off as the same signal.

Anyway, getting back to the proof: fs
Given r_{ff} , we can find the filter f having that autocorrelation (where r_{ff} is γ_{YY}).

We are given[4]

$$f * \overleftarrow{f} = r_{ff}$$

Taking Fourier transform

$$F\overline{F} = R_{ff}$$

Great! We have the **magnitude**. If we put ***any*** **phase**, we're through!
Not quite.
We have shown that we can get the filter f , but *our main goal should be to get a causal stable filter whose inverse filter exists and is also causal and stable.*

Taking logarithms on both sides,

$$\log F + \log \overline{F} = \log R_{ff}$$

If,

$$q = re^{j\alpha}$$

$$\log q = \log r + j\alpha$$

$$\log \overline{q} = \log r - j\alpha$$

$$\therefore \log \overline{q} = \overline{\log q}$$

Using the above result,

$$\log F + \overline{\log F} = \log R_{ff}$$

Let

$$\log F = G$$

$$\log R_{ff} = S$$

$$G + \overline{G} = S$$

Taking Inverse Fourier transform

$$g + \overleftarrow{g} = s$$

We can see that s is symmetric
Let us take g to be the positive half of s.

What we will now show is that if g (called the ***cepstrum***) is causal, then so is f .
We will assume that G is analytic and is expressed by the Laurent series

$G = g_0 + g_1 z + g_2 z^2 + \ldots,$
which form the Fourier transform coefficients of g.

Now,
$F = \text{antilog}\, G$

This implies that F is also analytic and can be expressed as
$F = f_0 + f_1 z + f_2 z^2 + \ldots$

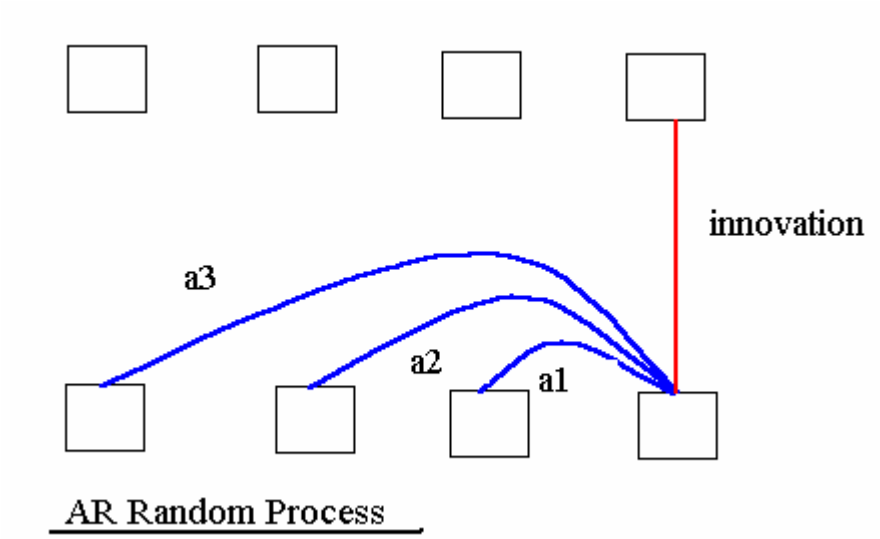
So $f_0, f_1, f_2, \ldots$ is the sequence that has F as its z-transform, which is the required **causal** filter f.
Further, G being analytic also implies that f is a *minimum phase filter*, which implies that if f is a rational function of z, then it has a **stable and causal inverse**.

Udayan's notes:
[1] "Shot Noise"
[2] "Inter symbol interference"!!!
[3] $1, -a_1, -a_2, -a_3, \ldots$
[4] The symbol $\overleftarrow{f}$ (read "f flipped") stands for the time-reversed f. $\overleftarrow{f}(n) = f(-n)$.

Weiner Filtering

WHAT IS AR RANDOM PROCESS?

AR process is a stationary random process whose coloring filter is Auto-Regressive(AR). Thus the present output is a linear function of previous outputs plus some innovation. The innovation can be white noise.



AR(p) denotes a filter with 'p' poles for the corresponding AR random process. An AR(p) filter will have corresponding MA(p) inverse or whitening filter.

Now even though, we have infinitely long auto-correlation filter. But its corresponding de-correlating filter is finite.

POWER SPECTRUM ESTIMATION:

What is this?

Power spectrum estimation means estimating the Fourier Transform of auto-correlation function.

Why?

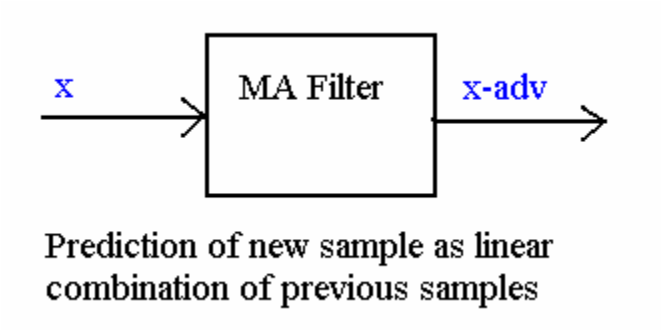
At a time, we'll have only finite samples of a random process in hand and we'll have to estimate the power spectrum from that may be for some prediction.

Ways of Power Spectrum Estimation:

1. Finding the auto-correlation function of the random process and taking its Fourier Transform.
2. Directly estimate the power spectrum.
3. First, we find the whitening filter. Then we invert it to get the corresponding coloring filter. From the coloring filter, we find the auto-correlation function of coloring filter which is the auto-correlation function of the stationary random process. In many applications, we don't have to go back to power spectrum from whitening filter. We can get Weiner filter directly from the whitening filter.
4. Estimate Reflection coefficients of lattice filter.

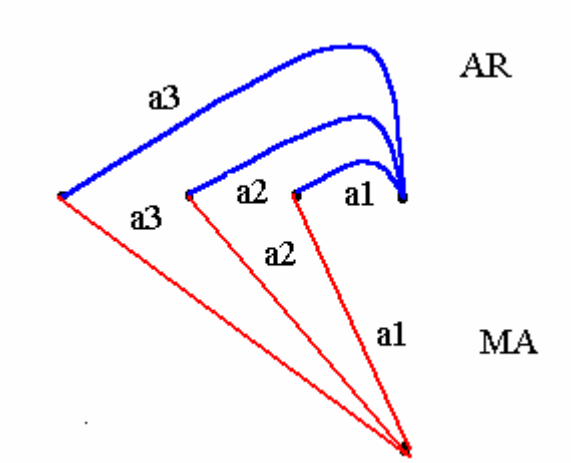
METHOD:

We have outcomes and we have an AR random process. We trying to estimate the coloring filter for this process provided order 'p' is known. We can do this using Least Square Deconvolution.



$$\begin{pmatrix} x0 & x1 & x2 & x3 \\ x1 & x2 & x3 & x4 \\ x2 & x3 & x4 & x5 \\ x3 & x4 & x5 & x6 \end{pmatrix} =$$

We'll estimate using MA filter. But as input x and output x_{adv} are same. It is actually an AR filter.



AR coefficients = MA coefficients

Now we want best estimate with whatever samples we have in hand. Suppose, we have 25 samples viz $x_0 \ldots x_{24}$, then we'll get nice estimate for the 1-step correlation i.e. correlation between x_0 and x_1, x_1 and x_2 ….. and so on till x_{23} and x_{24}. This is because we have 24 pairs for the interval of 1. But for 23-step correlation value, we have just two pairs x_0 and x_{23}, x_1 and x_{24}. Thus more no of samples means better estimate.

So for LMS, we give x, x_{adv} and $x_{backward}$ to increase the no of samples. This is valid because stationary random processes look the same from backwards.

$$\begin{pmatrix} x0 & x1 & x2 & x3 \\ x1 & x2 & x3 & x4 \\ x2 & x3 & x4 & x5 \\ x3 & x4 & x5 & x6 \\ \hline x4 & x3 & x2 & x1 \\ x5 & x4 & x3 & x2 \end{pmatrix} = \underline{\quad}$$

Here a1,a2,a3,a4 are the coefficients of prediction filter. We can get whitening (prediction error filter) filter by putting 1 in present sample and negating previous coefficients.

Consider Z-domain,
Here the prediction filter will be-
$a_1 z^{-1} + a_2 z^{-2} + a_3 z^{-3} + a_4 z^{-4}$

the prediction error filter will be –

$$1 - a_1 z^{-1} - a_2 z^{-2} - a_3 z^{-3} - a_4 z^{-4}$$

the corresponding coloring filter will be –

$$\frac{1}{1 - a_1 z^{-1} - a_2 z^{-2} - a_3 z^{-3} - a_4 z^{-4}}$$

flipped coloring filter will be-

$$\frac{1}{1 - a_1 z^{1} - a_2 z^{2} - a_3 z^{3} - a_4 z^{4}}$$

We have to convolve these two filters to get the power spectrum in rational form. Convolution will be Multiplication in z-domain.

So the rational function for power spectrum will be-

$$\frac{1}{(1 - a_1 z^{-1} - a_2 z^{-2} - a_3 z^{-3} - a_4 z^{-4})(1 - a_1 z^{1} - a_2 z^{2} - a_3 z^{3} - a_4 z^{4})}$$

MA Processes:

For these processes, coloring filter is MA so corresponding whitening filter will be AR.

ARMA Processes:

For ARMA processes, both coloring and whitening filter will be ARMA with poles and zeros interchanged between them.

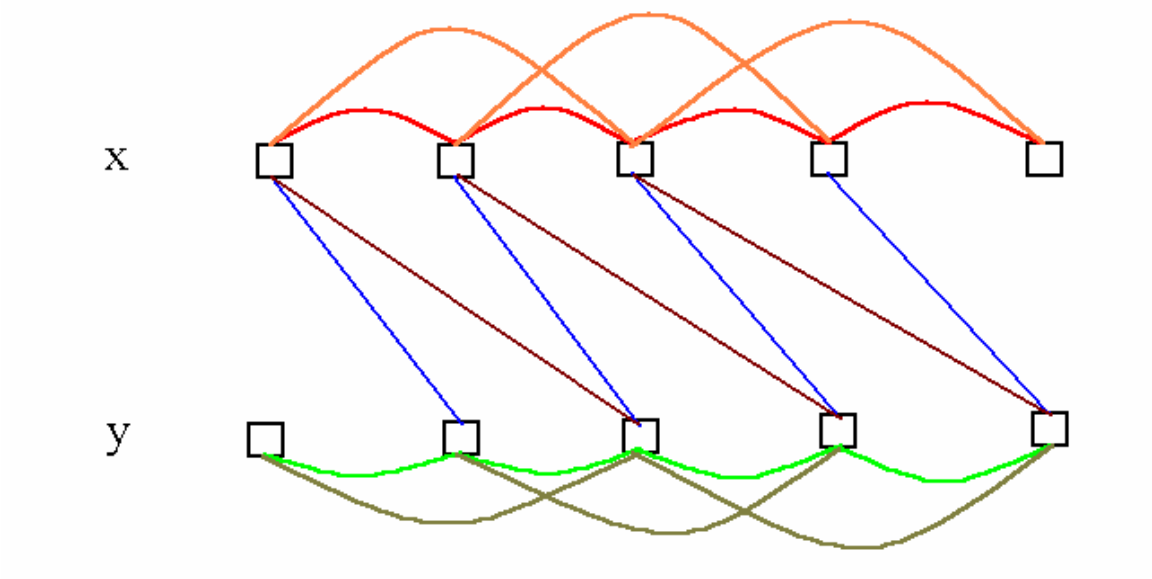
WEINER Filtering:

Weiner filtering is estimating a random process from few other random processes. Prediction is a special case of Weiner filtering where same random process is estimated in advance from previous samples.

Jointly Stationary Random Processes:

X and Y are said to be Jointly Stationary Random Processes if and only if-

1. If X is a stationary random process.
2. If Y is a stationary random process.
3. If the cross-correlation $r_{xy}(k,l)$ depends only on the difference (k-l).



Jointly Stationary Random Processes X and Y

In the above figure, we can see two jointly stationary random processes X and Y. The correlation values shown by same color are equal. So we can see that X and Y are independently stationary plus their cross-correlation is also same for same interval. This makes them Jointly stationary random processes.

If two processes are jointly stationary, then we can design the Weiner filter for them.

FIR Weiner Filter:

If we have two jointly stationary random processes X and Y and we want to predict Y from the samples of X. We now have auto-correlation of X and cross-correlation between X and Y.

Here

The auto-correlation of X is given by-

$$E\{x(n-l)x^*(n-k)\} = r_x(k-l)$$

The cross-correlation of X and Y is given by-

$$E\{y(n)x^*(n-k)\} = r_{xy}(k)$$

Now if 'w' is the Weiner filter response, we have-

$$w * r_{xx} = r_{xy}$$

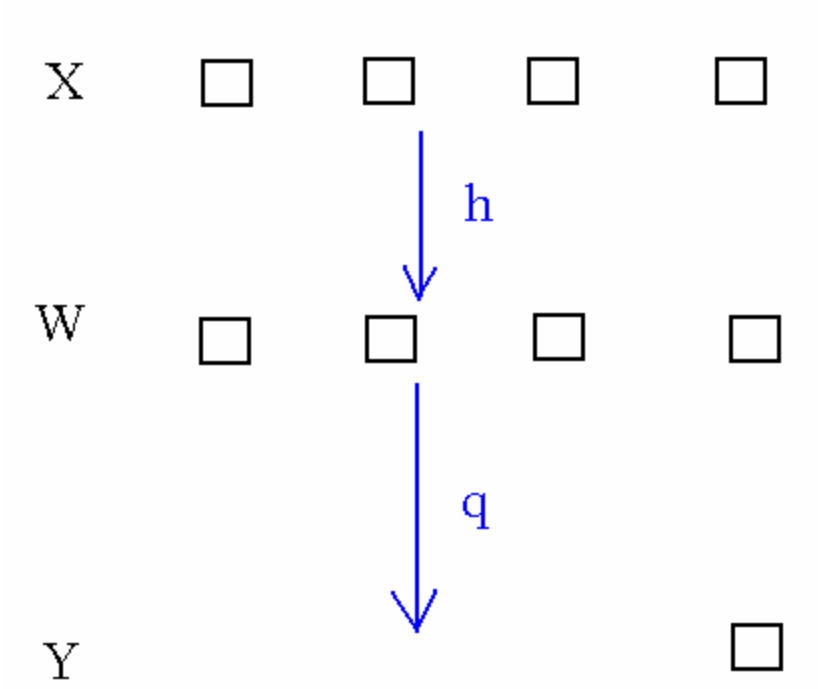
$$\begin{pmatrix} rx(0) & rx^*(1) & \ldots\ldots & rx^*(p-1) \\ rx(1) & rx(0) & \ldots\ldots & rx^*(p-2) \\ rx(2) & rx(2) & \ldots\ldots & rx^*(p-3) \\ . & & \ldots\ldots & \\ . & & \ldots\ldots & \\ rx(p-1) & rx(p-2) & \ldots\ldots & rx(0) \end{pmatrix} \begin{pmatrix} w(0) \\ w(1) \\ w(2) \\ . \\ . \\ w(p-1) \end{pmatrix} = \begin{pmatrix} rxy(0) \\ rxy(1) \\ rxy(2) \\ . \\ . \\ rxy(p-1) \end{pmatrix}$$

so

$W = E[XX^T]^{-1} E[XY]$

Causal IIR Weiner Filter:

Now we want to design a causal IIR Weiner filter for prediction of a random process Y from samples of another random process X.



We first calculate W which is filtered version of X. W is the innovation process of X. Now X and Y are jointly stationary, hence W and Y are also jointly stationary.

Now we have cross-correlation of X and Y. But we need cross-correlation of W and Y.

$E[w_{k-i}\, y_n] = E[(x_{n-i}h_0 + x_{n-i-1}h_1 + x_{n-i-2}h_2 + \ldots.)\, y_n]$

$= h_0 E[x_{n-i}\, y_n] + h1\, E[x_{n-i-1}\, y_n] + h\, E[x_{n-i-2}\, y_n] + \ldots..$

$r_{wy}(i) = h_0 r_{xy}(-i) + h_1 r_{xy}(-i-1) + \ldots\ldots\ldots$

$r_{wy} = h * r_{xy}$

$q(i) = E[W(n-i)y(n)] = r_{wy}(-i)$

as we want causal filter only,

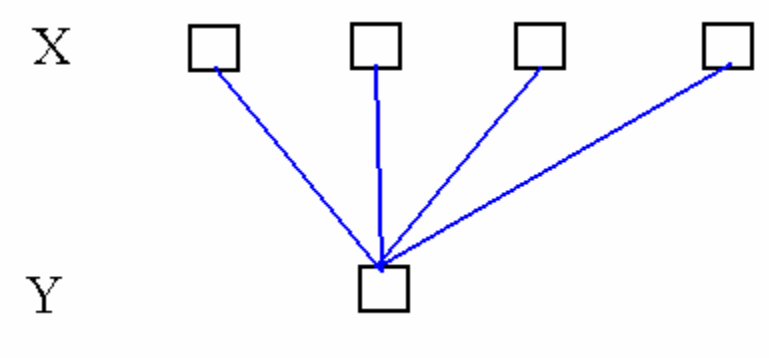
q = [flipped r_{wy}]+

= [(flipped r_{xy}) * (flipped h)]+

h*q = h * [(flipped r_{xy}) * (flipped h)]+

Here h and q both are causal so even (h * q) is also causal.

UNREALIZABLE Weiner filter (Non-Causal):



Non-Causal Weiner Filter

Here we have all boxes(analogy!) open. So we know all samples of X. So now comapared to previous case of causal filter, we have

q = [flipped r_{wy}]

h*q = h * [(flipped r_{xy}) * (flipped h)]

h*q = h * (flipped h)* (flipped r_{xy})

So in Z-domain –

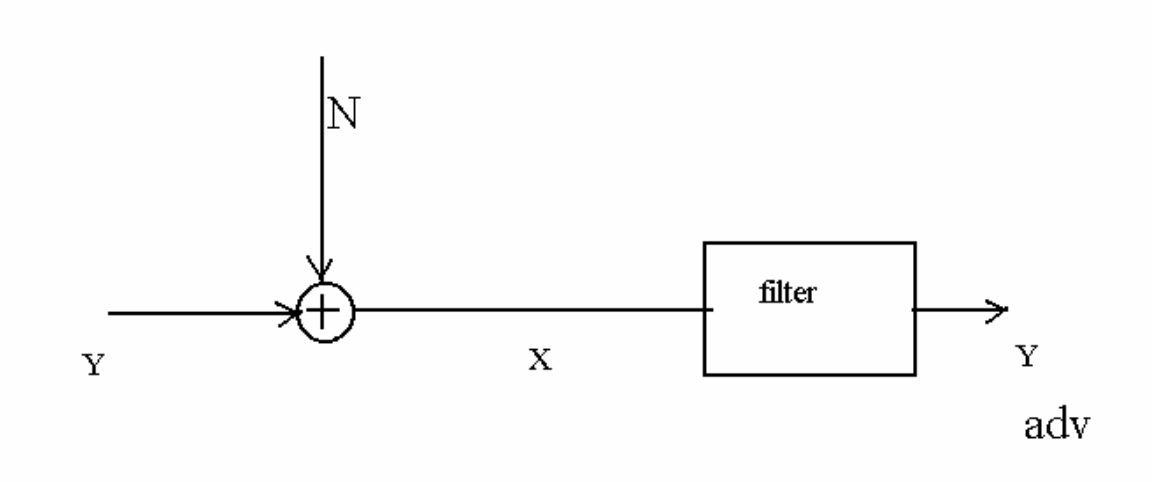
$H(z)Q(z) = R_{hh}\lceil yx$

let ‘g’ be inverse filter of h. Then R_{gg} will be auto-correlation of X.

$= \lceil yx / R_{gg}$

$H(z)Q(z) = \lceil yx / \lceil xx$

Weiner Filtering for noise removal:



Removal of noise from the past samples is also called smoothing.

$E[X_t X_{t-i}] = E[(y_t + N_t)(y_{t-i} + N_{t-i})]$

$= E[(y_t\, y_{t-i})] + E[(N_t\, N_{t-i})] + E[(y_t\, N_{t-i})] + E[(y_{t-i}\, N_t)]$

$r_{xx}(i) = r_{yy}(i) + r_{NN}(i) + r_{yN}(i) + r_{Ny}(i)$

Now last two terms are zero cause noise is most of the times un-correlated with signal Y.

If noise is white noise, then $r_{NN}(i)$ is a delta function.

$E[Y_t X_{t-i}] = r_{xy}(i)$

$= E[Y_t Y_{t-i}] + E[Y_t N_{t-i}]$

$r_{xy}(i) = r_{yy}(i) + r_{yN}(i)$

Power Spectrum Estimation

There are basics representations of signal which can be converted to other one in different ways.

Auto-correlation	Power density function
γ	Γ

a	κ
Coloring Filter coefficients	Reflection coefficients

Methods to find Time–Average auto-correlation function

Unbiased estimation method

Here we partly open the boxes and find the auto-correlation such that

$$E[XnXn\text{-}i] = A_n \; XnXn\text{-}i$$

$$= \frac{1}{N-i} \sum Xn\,Xn\text{-}i \qquad n \text{ going from } i \text{ to } N$$

Here variance is $E\left([1^{st}\text{ term} - 2^{nd}\text{ term}]\right)^2$

Since there are few data points for larger lags we have more variance at larger lags.

Biased estimation method

$$E[XnXn\text{-}i] = \frac{N-i}{N} E[Xn\,Xn\text{-}i] \qquad n \text{ going from } i \text{ to } N$$

This is similar to windowing signal (auto-correlated) with a triangular window and doing average.

Less variance than unbiased estimate.

Relationship between Energy, Power spectrum and Periodogram.

Periodogram is Fourier transform of average (windowed) auto-correlation function.
Expected value of Periodogram is Power spectrum.

So by averaging Periodogram we get the Power Spectrum of signal.
Also in direct method by taking Fourier transform of auto-correlation function we can get the Power spectrum.

Auto-correlation gives the energy spectrum.

Non-parametric methods for Power Spectrum Estimation

1. Bartlett Method

In Bartlett method we divide the signal into blocks, find their periodograms and average them to get the Power spectrum. (The data segments are non-overlapping).
The final effect is true power spectrum convolved with a window.
Due to windowing (leakage frequency due to side lobes) the frequency resolution is low.

2.Welch Method

It is same method than above with some modifications –

I) Data segments can be overlapping.
II) Window the data (signal) before computing Periodogram (we may use different windows for each segment)

This method has got better precision but less frequency resolution than Bartlett method.

3. Blackman-Tukey Method

In this method we windowed the auto-correlation sequence and take Fourier transform to get power spectrum estimate (Periodogram) in effect we smooth out the Periodogram.
It has better variance (even at large lags) and better precision than above two methods.
But frequency resolution is less than the others.

Parametric method for Power Spectrum Estimation

Theme:
In these methods we assume that signal is output of a system having white noise as an input . We model the system and get its parameters i.e. coloring filter coefficients and predict the power spectrum.

Yule-Walker method

We estimate the Auto-correlation. then we find the ‘a’ s coloring filter coefficients which are model parameters.
To find the ‘a’s we use Levinson-Durbin algorithm.

From these a’s we again find the γ and then Power spectrum.

Burg Method

We have seen the lattice filter equations for forward and backward prediction error filters.

$$Q_5(n) = Q_4(n) - \kappa_4 R_4(n-5)$$

$$R_5(n) = R_4(n-5) - \kappa_4 Q_4(n)$$

Q is the error quantity (least square).

So if we minimize the error by selecting κ we can model the 'a's

For that we predict the $\kappa 1$ and using this we find other reflection coefficients using same lattice structure.
Using Levinson-Durbin algorithm we model the system and find the 'a's from which we can get the power spectrum estimate.

Predict : $\kappa 1$ =(autocorrelation of x(n))/(energy in x(n))

And we can find the

$$\kappa_4 = \frac{-\langle R_4 \text{ delay } 5, Q_4 \rangle}{\langle R_4, R_4 \rangle}$$

Using property that forward and backward coefficients are the same one

$$\kappa_4 = \frac{-\langle R_4 \text{ delay } 5, Q_4 \rangle - \langle Q_4, R_4 \text{ delay } 5 \rangle}{\langle R_4, R_4 \rangle + \langle Q_4, Q_4 \rangle}$$

Thus we can find R1 and Q1 from K1…then K2 from R1 and Q1 …then K2 and so on…

From κ's we find a's and the Power spectrum.
This method has higher frequency resolution and computationally efficient.

Lecture Date: 16th & 17th Apr '04

Solving Least Squares

The problem

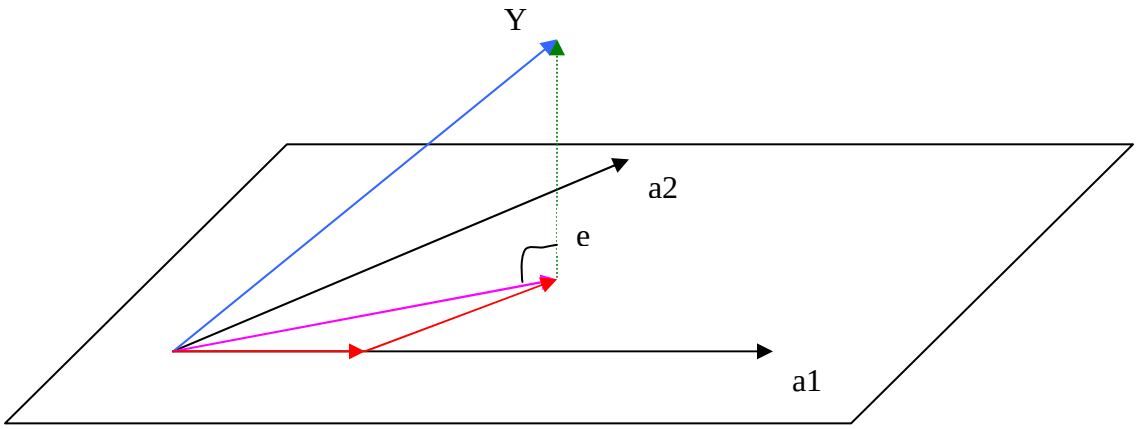
Given a vector Y and a set of basis vectors A = $\{a_1, a_2, \ldots a_n\}$, we want to find the measure of Y in terms of A i.e. find M such that

MA = Y

The vector space spanned by Y and A differs, giving us a set of over-determined equations. So, we find an M such that

|| Y – MA || is minimum.

So, we are basically trying to - find the shadow of Y in the subspace spanned by A, and also find the linear combinants, which add up to that shadow. This is illustrated by the figure below. The vectors in red are the linear combinants in the direction of a1 & a2.



The vector in pink is the shadow of Y in the subspace of A. 'e' is the error, which is **orthogonal to the subspace spanned by A.**

We notice that if the A is an orthogonal basis, the solution to the problem is simplified. If the basis is orthogonal, then the linear combinants are just the coordinate values of Y (the analysis and synthesis matrix is same).

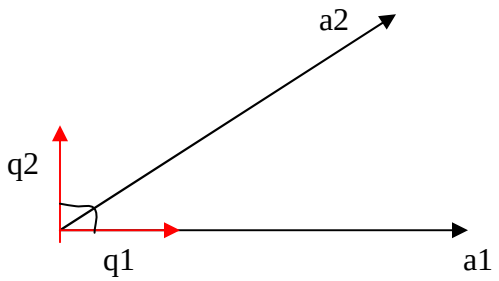
Mathematically, the same can be justified by the fact that the inverse of an orthogonal matrix is its conjugate transpose.

To find the linear combinants we need the inverse of the basis matrix as shown below.

AC = Y, C is [λ_1, λ_2….., λ_n], the scales by which each direction in A needs to be scaled so that they all add up to Y.
$C = A^{-1} Y$.

Thus the least square problem is reduced to finding a set of orthogonal set of basis Q such that
Span { Q } = Span { A }
i.e. a set of Orthogonal vector space spanning the same subspace as that of the given basis.

For example in a 2-D subspace, it can be shown as below. q1 and q2 are the normalized orthogonal vectors for the subspace of a1 & a2

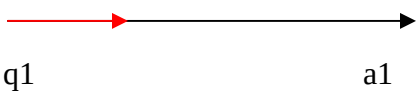


Therefore we have different methods of orthogonalizing the basis vector A.

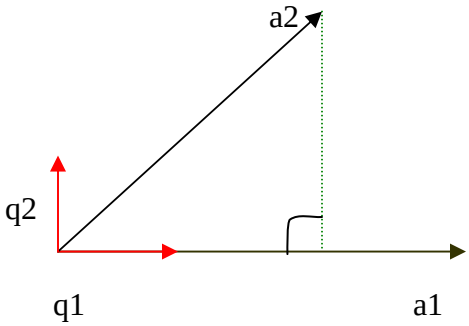
I. Graham Schmidt Method

This is the successive orthogonalization (the QR method).
Let A contain n linearly independent vectors. We want to find the orthonormal basis for A

q1 is a1/|a1|

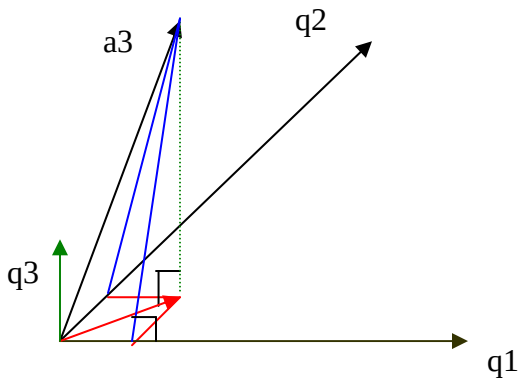


For a2, we take the projection of a2 on a1, to find the orthogonal vector q2



∴ q2 = a2 – proj of a2 on a1 / |a2 – proj of a2 on a1|
q3 is a vector orthogonal to subspace of a1& a2. Its simple to see this a least squares problem. (given vector a3, and need to find its shadow in a1 & a2, so that we can find the direction of the error which is orthogonal to a1 & a2)

Hence we use the orthogonal vectors q1 and a2 to find q3.
First we drop perpendicular to q1 and q2 (refer to the figure below) shown in blue. Assume a3 is coming out of the monitor.



Now, if we look at a3 to be made up of 2 parts, 1 which lies in the subspace of q1, q2 and the other remaining is the component orthogonal to q1 & q2.
By, dropping perpendiculars on q1 and q2 we find the projection of a3 in q1 & q2. Since q1,q2 are ortho to each other, these projections when added up will give the shadow of a3 in q1 & q2. Subtracting this from a3 gives us the component that not in q1 & q2 i.e orthogonal.

In the figure, the shadow of a3 is in red.

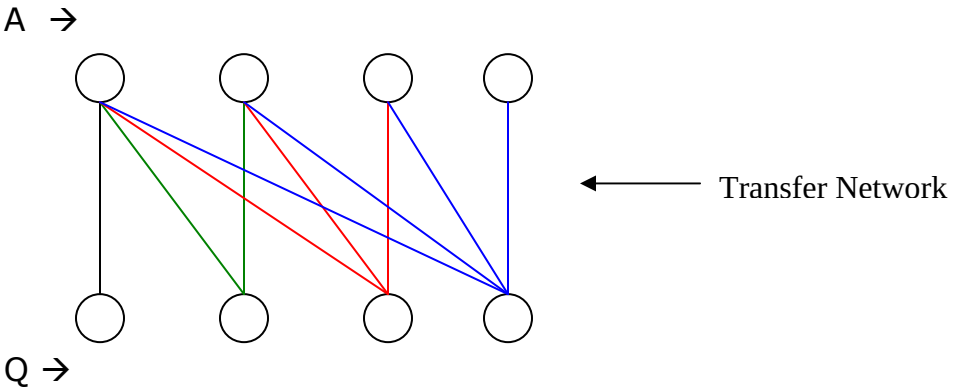
∴ q1 = a1
∴ q2 = a2 – proj $_{q1}$ a2
∴ q3 = a3 - proj $_{q1}$ a3 - proj $_{q2}$ a3

The same can be shown by the following transfer network

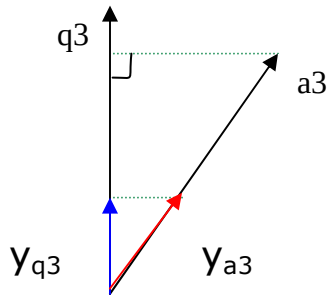
A can be considered to be made of 2 vectors Q and R, such that

A = QR, Q is the orthogonal basis and R is an upper triangular matrix, which is the transfer n/w shown in the figure.

The reason it is easy to invert A is that triangular matrices are simple to invert.



We have the linear combinants of Y in terms of Q, namely Y's coordinate values. But, what we actually want are the linear combinants in terms of A. This can be done in a simple way shown by the figure below.



We find the projection vector of a3 on q3. (Note, for a 3-D space, only a3 can have any energy along q3). We use the same proj vector to find a vector along a3 such that y_{q3} is its projection. By similarity we see that this all the energy Y can have along a3. We subtract this from Y, so the only components left in Y would be of a1 and a2. They can too be found in the same way.

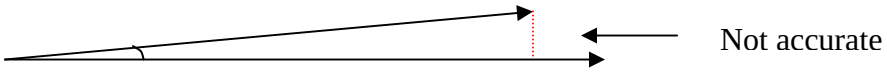
<u>Why this method is not that good</u>
It is easy to see in this method that the error seeps through. If we have to find the Q for n vector space, the final error is n times as large as the initial error.

<u>Solution</u>: Modified Graham Schimdt's Method
Similar to GSM, but after finding an ortho vector qn, we find the energy of each remaining vector in A along qn and subtract this energy. It is easy to see that the two methods are equivalent, but the

energy is reduced stepwise from each vector. This gives better numerical results.

Why both methods are bad

For ill-conditioned vectors, we get numerically bad results. Consider two vectors very close to each other, the result after subtraction has low resolution, giving bad results.



II Givens Rotation

Instead of finding the orthogonal basis, we start with an orhto basis and then rotate it into to mach the given subspace. The error here will be because of the fact that the subspaces spanned by the 2 basis may not be identical, but the error doesn't get amplified proportionally to the no. of vectors in the basis.

The important thing to note here is that are doing planar rotations, so at a time only 2 co-ordinates changes, the ones which are in the plane of rotations

We need to match q1 with a1, q2 with a2 and so on. We do this in steps.

1. Take a1 and rotate it into q1 (the inverse rotations are the ones required to rotate q1 into a1).
2. Then we rotate a2 into the plane of q1 & q2 such that not component along q1 does not change.

We successively do this till A is rotated into Q.

Consider a 3-D space. If we want to rotate a1 into X axis, we could first rotate in the X-Z plane and then in the X-Y plane or first in the X-Y plane and then in the X-Z plane. We choose the following method

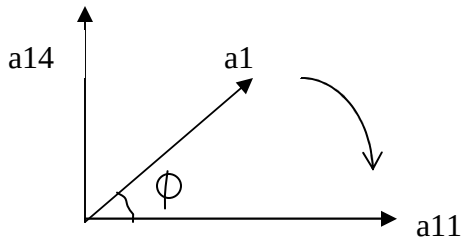
> Choose the plane with one axis as the one you want to zero out, and the other the axis which we want to move into.

For a 4-D space

$$a1 = \begin{pmatrix} a11 \\ a12 \\ a13 \\ a14 \end{pmatrix} \quad a2 = \begin{pmatrix} a21 \\ a22 \\ a23 \\ a24 \end{pmatrix}$$ and so on.

For moving a1 in q1, we want to zero out a14 component, so choose the a11-a14 plane of rotation.

$$\therefore \text{ Rotation matrix } = \begin{pmatrix} \cos\Phi & 0 & 0 & \sin\Phi \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -\sin\Phi & 0 & 0 & \cos\Phi \end{pmatrix}$$



Then we use a11-a13 plane of rotation followed by a11 – a12.

For rotating a2 into X-Y plane, we have to ensure that X component is not in the plane of rotation.
So, we choose a22 – a24 as plane of rotation and then a22 – a23 as plane of rotation.

We want to solve for 'x' using least squares

$\therefore$ Ax = B

$\because$ A = QR

QR x = B
Rx = Q' B (Q' = Q^{-1})
x = R^{-1} Q'B

Since, we need the inverse of Q anyway, we do not solve for the inverse rotations. The above rotation matrices are Q^{-1} , and the upper triangular matrix formed by applying these transformations to A, is R.

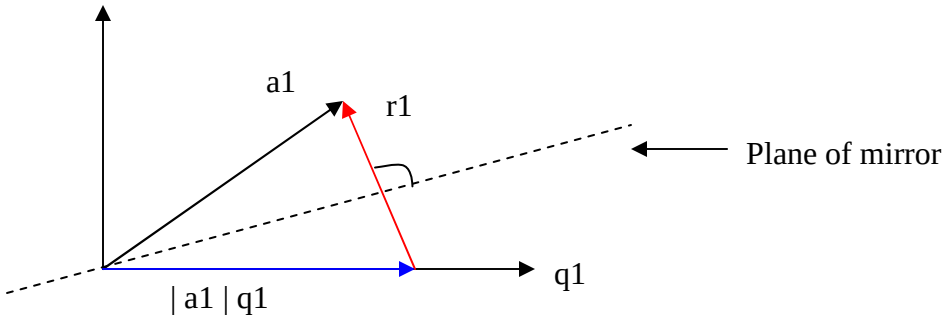
Householder Reflections

"No our reflections are upside down, left-right flipped, back-fwd flipped. The man in the mirror is actually an alien, with no symmetry or is it up down symmetric. It's all about how we turn!"

"Dabum dubum thubum…!"

Householder Reflections

The reason we use reflections to match the orthogonal basis to A, is that reflections are less expensive than rotations.



Therefore, r1 = a1 - |a1 | q1

We want to reflect A in the direction of r1.
This is given by r_1^T A. If we think of A to consists of two parts, one component along in the mirror, and the other ortho to it. So, the component in the mirror does not change, and the other gets flipped.

Therefore, after flipping we get $- r_1^T A$.
Then we reconstruct the vector i.e $- r_1 r_1^T A$.
The component that doesn't change is A $- r_1 r_1^T A$.

$\therefore$ the final vector is the addition of these two vectors

$A - r_1 r_1^T A - r_1 r_1^T A$
$= (I - 2 r_1 r_1^T) A$

For, reflecting a2 into q2, we must ensure that component along q1 doesn't change. So, we flatten this dimension, so that it is just point and lies in the mirror.

$$a2 = \begin{pmatrix} a21 \\ a22 \\ a23 \\ a24 \end{pmatrix}$$ ← Don't consider this component

Thus Q is the matrix formed by ($I - r\, r^T$) and R is the triangular matrix formed transforming A.