## 2D Signal Processing

1D signal has one independent variable - f(t)

2D signal has two independent variables - f(x,y)

Concepts of linearity, spectra,filtering, etc, carry over from 1-D. But concept of causality not relevant as image is a fn of space, not time.

2-D systems are more complex, e.g we can factor 1-D polynomials into a product of 1st and 2nd order polynomial, and thus study stability and system response. Stability for a 1-D system can be determined from system poles, but for 2-D system, poles are surfaces in 4-D space.

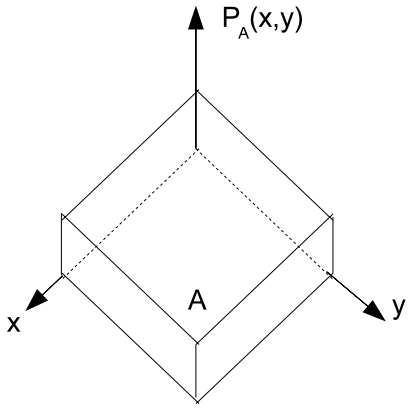
2-D algorithms can offer more flexibility implementation, e.g one can process image data in a non-causal way, in parallel. We now extend 1-D results to 2-D where possible.

65

A 2-D analog signal is a function of 2 continuous variables.

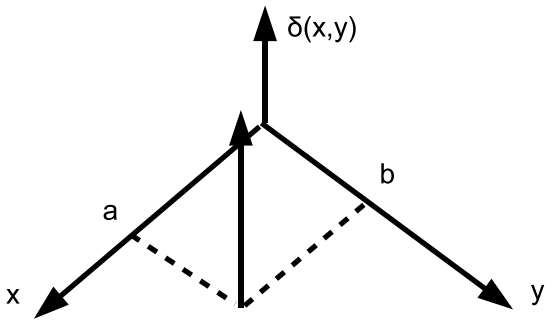
A 2-D pulse is defined by :

$$P_A(x,y) = \{ \begin{array}{ll} 1 & x,y \in A \\ 0 & x,y \notin A \end{array}$$




The 2-D (Dirac) unit impulse is defined by:

$$\delta(x,y) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y)\delta(x-a,y-b)dxdy = f(a,b)$$




It can be represented as product of two 1-D impulses:

$$\delta(x,y) = \delta(x).\delta(y)$$

Response of a system to 2-D impulse is termed point-spread fn.

**Linear Systems and Convolution**

A 2-D system is a mapping: $g(x,y) = T[f(x,y)]$ which is linear if superposition holds, i.e, if

$$T[af_1(x,y) + bf_2(x,y)] = aT[f_1(x,y)] + bT[f_2(x,y)]$$

and is shift-invariant if

$$T[f(x-x_1, y-y_1)] = g(x-x_1, y-y_1)$$

when $T[f(x,y)] = g(x,y)$

The point-spread fn is:

$h(x,y) = T[\delta(x,y)]$.

Since

$$f(x,y) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(m,n)\delta(x-m, y-n)dmdn$$

,

$$g(x,y) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(m,n)T[\delta(x-m, y-n)]dmdn$$

$$= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(m,n)h(x-m, y-n)dmdn$$

$= f**h$ - 2D convolution

67

**Seperable 2-D Systems**

A system is seperable if $h(x,y) = h_1(x).h_2(y)$

Then the 2D convolution reduces to:

$$g(x,y) = \int\int f(m,n)h_1(x-m)h_2(y-n)dmdn$$

$$= \int h_2(y-n)[\int f(m,n)h_1(x-m)dm]dn$$

$$= h_2(y) * [h_1(x) * f(x,y)]$$

i.e, two 1-D convolutions.

For a fixed but arbitary value of n, one performs the 1-D convolution over the 1st dummy variable m:

$$\phi(x,n) = \int f(m,n)h_1(x-m)dm$$

Then $\phi(x,n)$ is convolved with $h_2(y)$:

$$g(x,y) = \int \phi(x,n)h_2(y-n)dn$$

68

**Fourier Transform**

As for 1-D, "test" signal is complex exponential:

$$f(x,y) = e^{j(\omega_1 x+\omega_2 y)} = e^{j\omega_1 x}.e^{j\omega_2 y}$$

The response is:

$$g(x,y) = \int\int e^{j[\omega_1(x-m)+\omega_2(y-n)]}h(m,n)dmdn$$

$$= H(\omega_1,\omega_2)e^{j(\omega_1 x+\omega_2 y)},$$

where

$$H(\omega_1,\omega_2) = \int\int e^{-j[\omega_1 m+\omega_2 n]}h(m,n)dmdn$$

is the Fourier Tfm of h (x,y). The inverse formula is

$$h(x,y) = (\frac{1}{2\pi})^2 \int\int H(\omega_1,\omega_2)e^{j(\omega_1 x+\omega_2 y)}d\omega_1 d\omega_2$$

Since the complex sinusoid is seperable, the output can be got by two 1-D convolutions:

1. $\Theta(\omega_1, y) = \int f(x,y)e^{-j\omega_1 x}dx$

2. $F(\omega_1,\omega_2) = \int \Theta(w1,y)e^{-j\omega_2 y}dy$

As for 1-D :

$$f_1(x,y) * * f_2(x,y) < ----- > F_1(\omega_1,\omega_2).F_2(\omega_1,\omega_2)$$

as does Parseval's theorem:

$$\int\int |f(m,n)|^2 = (\frac{1}{2\pi})^2 \int\int |F(\omega_1,\omega_2)|^2 d\omega_1 d\omega_2$$

69

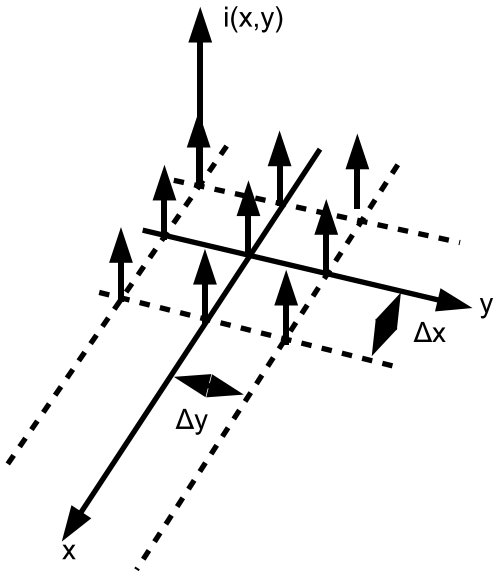
**Sampling 2-D signals**

Consider band-limited signal,f ( x, y ), where

$$F(\omega_1, \omega_2) = 0 \quad for \quad (\omega_1, \omega_2) \notin R$$

The 2-D sampling fn is

$$i(x, y) = \sum_m \sum_n \delta(x - m\Delta x, y - n\Delta y)$$




The Fourier Tfm of a 2-D series of impulses is also a series of impulses:

$$I(\omega_1, \omega_2) = \frac{2\pi}{\Delta x}\frac{2\pi}{\Delta y}\sum_m \sum_n \delta(\omega_1 - m\frac{2\pi}{\Delta x}, \omega_2 - n\frac{2\pi}{\Delta y})$$

The impulse - sampled signal is

$$f_s(x, y) = f(x, y).i(x, y) = \sum_m \sum_n f(m\Delta x, n\Delta y)\delta(x - m\Delta x, y - n\Delta y)$$

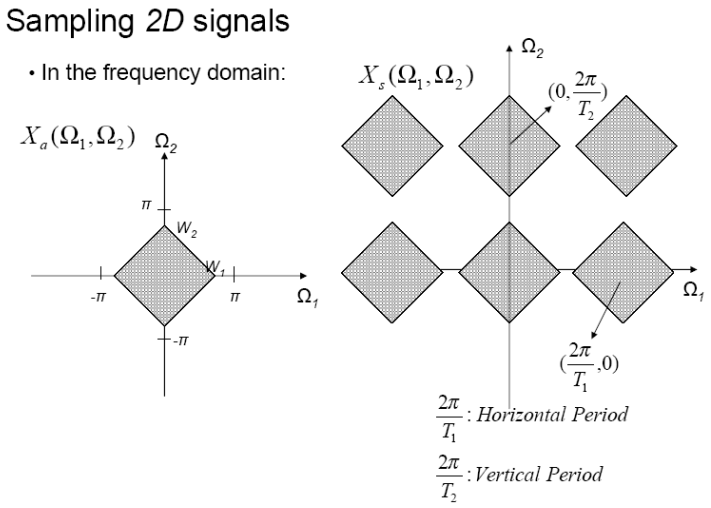
70

The Fourier Tfm of the sampled signal is

$$F_s(\omega_1,\omega_2) = \mathcal{F}[f(x,y).i(x,y)] = (\frac{1}{2\pi})^2 F(\omega_1,\omega_2) * *I(\omega_1,\omega_2)$$

$$= \frac{1}{\Delta x \Delta y}\sum_m \sum_n F(\omega_1 - m\frac{2\pi}{\Delta x},\ \ \omega_2 - n\frac{2\pi}{\Delta y})$$

i.e, the F.T of the sampled signal is the periodic repitition of $F(\omega_1,\omega_2)$ at a spacing of $\frac{2\pi}{\Delta x}, \frac{2\pi}{\Delta y}$




If $\Delta x$ and $\Delta y$ are small enough that the "basebands" don't overlap, we can recover the signal by a LPF:

$$F(\omega_1,\omega_2) = H(\omega_1,\omega_2).F_s(\omega_1,\omega_2),$$

$$H(\omega_1,\omega_2) = \{ \begin{array}{ll} 1 & (\omega_1,\omega_2) \in R \\ 0 & (\omega_1,\omega_2) \notin R \end{array}$$

$$f(x,y) = h(x,y) * * f_s(x,y)$$

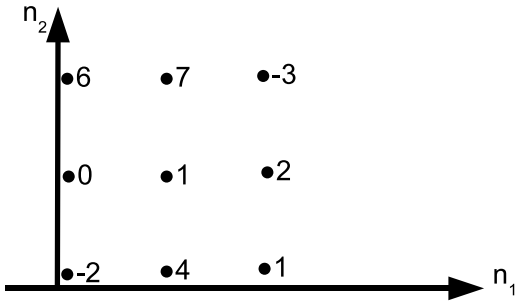
$$= \sum_m \sum_n f(m\Delta x, n\Delta y) h(x - m\Delta x, y - n\Delta y)$$

The impulse response of the LFP is:

$$h(x,y) = \frac{sin\Theta_1 x}{\Theta_1 x}.\frac{sin\Theta_2 y}{\Theta_2 y}, \Theta_1 = \frac{\pi}{\Delta x}, \Theta_2 = \frac{\pi}{\Delta y}$$

**2D Discrete Signals**

Normalise $\Delta x = \Delta y = 1$ to give $x(n_1, n_2)$ which can be represented by array:




Some discrete signals:

1. 2-D impulse:

$$\delta(n_1, n_2) = \delta(n_1).\delta(n_2) = \{ \begin{array}{ll} 1 & at \;\; n_1, n_2 \\ 0 & otherwise \end{array}$$

2. 2-D line impulse:

$$x(n_1, n_2) = \delta(n_1, 1)\delta(n_1).1 = \{ \begin{array}{ll} 1, & n_1 = 0 \\ 0 & otherwise \end{array}$$

3. Unit step fn:

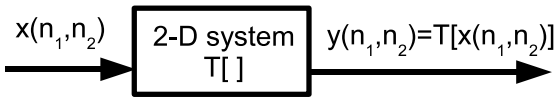
$$u(n_1, n_2) = \{ \begin{array}{ll} 1, & n_1 \geq 0, \; n_2 \geq 0 \\ 0 & otherwise \end{array}$$

4. ( Complex ) Exponential :

$$x(n_1, n_2) = a^{n_1} b^{n_2} \;\; (= e^{j(\omega_1 n_1 + \omega_2 n_2)} \; if \; a, b \; complex)$$

72

**2-D Systems**




The system is linear if

$$T[ax_1(n_1, n_2) + bx_2(n_1, n_2)] = aT[x_1(n_1, n_2)] + bT[x_2(n_1, n_2)]$$

Its impulse response is

$$T[\delta(n_1 - k_1, n_2 - k_2)] = h(n_1, k_1; n_2, k_2)$$

By superposition, system $o/p$, $y(n_1, n_2)$ is

$$T[x(n_1, n_2)] = T[\sum_{k_1} \sum_{k_2} x(k_1, k_2)\delta(n_1 - k_1, n_2 - k_2)]$$

$$= \sum_{k_1} \sum_{k_2} x(k_1, k_2)T[\delta(n_1 - k_1, n_2 - k_2)]$$

A system is spatially invariant if

$$T[x(n_1, n_2)] = y(n_1, n_2)$$

implies

$$T[x(n_1 - m_1, n_2 - m_2)] = y(n_1 - m_1, n_2 - m_2)$$

Then, impulse response reduces to

$$h(n_1, k_1; n_2, k_2) = h(n_1 - k_1, n_2 - k_2)$$

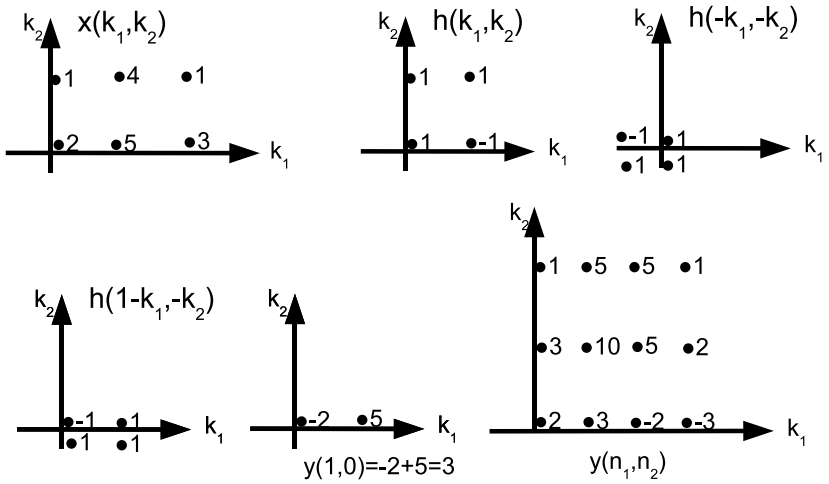
73

**Convolution**

For LSI systems, the output becomes

$$y(n_1, n_2) = \sum_{k_1} \sum_{k_2} x(k_1, k_2) h(n_1 - k_1, n_2 - k_2) = x * *h$$

E.g: Convolve 3 x 2 and 2 x 2 arrays $x(k_1, k_2)$ and $h(k_1, k_2)$:




Convolution of $N_1 x N_2$ and $M_1 x M_2$ arrays yields array of size $(N_1 + M_1 - 1) x (N_2 + M_2 - 1)$

2D Convolution simplifies for seperable signal. Then:

$$y(n_1, n_2) = \sum_{k_1} x_1(k_1) \sum_{k_2} x(k_2) h(n_1 - k_1, n_2 - k_2)$$

can be computed in 2 stages:

$$1.\ g(n_1, n_2) = \sum_{k_2} x_2(k_2) h(n_1, n_2 - k_2) = x_2 * *h$$

$$2.\ y(n_1, n_2) = \sum_{k_1} x_1(k_1) g(n_1 - k_1, n_2) = x_1 * *g$$

Needs $NM(N + M - 1) + M(N + M - 1)^2$ mpy/adds if $N_1 = N_2 = N$ and $M_1 = M_2 = M$ cf $(N + M - 1)^2 M^2$ for non-seperable signals.

74

**Frequency response and Fourier Transform**

LSI system response to complex 2D sinusoid $e^{j(\omega_1 n_1+\omega_2 n_2)}$

$$y(n_1,n_2)=\sum_{k_1}\sum_{k_2}e^{j(\omega_1 n_1+\omega_2 n_2)}h(k_1,k_2)$$

$$=H(\omega_1,\omega_2)e^{j(\omega_1 n_1+\omega_2 n_2)}$$

$H(\omega_1,\omega_2)$ is Fourier series of periodic fn of 2 variables and h is Fourier coeff:

$$h(n_1,n_2)=\frac{1}{2\pi}\int_{-\pi}^{\pi}\int_{-\pi}^{\pi}H(\omega_1,\omega_2)e^{j(\omega_1 n_1+\omega_2 n_2)}d\omega_1 d\omega_2$$

If

$$y(n_1,n_2)=x(n_1,n_2)**h(n_1,n_2)$$

then

$$Y(\omega_1,\omega_2)=X(\omega_1,\omega_2).H(\omega_1,\omega_2)$$

2-D Correlation:

$$h\ {}_{oo}\ g=\sum_{n_1}\sum_{n_2}g^*(n_1,n_2)h(n_1+k_1,n_2+k_2)\overset{FT}{\leftrightarrow}G^*(\omega_1,\omega_2)H(\omega_1,\omega_2)$$

75

**DFT**

2-D DFT pair is extrapolated from 1-D version:

$$\tilde{X}(k_1,k_2)=\sum_{n_1=0}^{N_1-1}\sum_{n_2=0}^{N_2-1}\tilde{x}(n_1,n_2)e^{-j(2\pi n_1k_1/N_1+2\pi n_2k_2/N_2)}$$

$$\tilde{x}(n_1,n_2)=\frac{1}{N_1N_2}\sum_{k_1=0}^{N_1-1}\sum_{k_2=0}^{N_2-1}\tilde{X}(k_1,k_2)e^{j(2\pi n_1k_1/N_1+2\pi n_2k_2/N_2)}$$

The DFT for $N_1$ by $N_2$ array $x(n_1,n_2)$ is computed in 2 stages:

1. Compute 1-D DFT for each row:

$$T(k_1,n_2)=\sum_{n_1=0}^{N_1-1}x(n_1,n_2)e^{-j(2\pi n_1k_1/N_1)},\ n_2=0,1.....N_2-1$$

This takes $(N_2.N_1/2)log_2N_1$ complex mults using FFT

2. Compute 1-D DFT for each column of $T(k_1,n_2)$:

$$\tilde{X}(k_1,k_2)=\sum_{n_2=0}^{N_2-1}T(k_1,n_2)e^{-j(2\pi n_2k_2/N_2)},\ n_1=0,1.....N_1-1$$

which takes $N_1$ FFTs each length of $N_2$, i.e another $N_1\frac{N_2}{2}log_2N_2$ complex mults.

$\Rightarrow$ Total complex mults: $\frac{N_1N_2}{2}log_2(N_1N_2)$

76

**Z Transform**

Similar to 1-D case with 2 main limitations:

1. No polynomial factorization for 2-D polynomials so concept of pole and tests for stability are more complex

2. The difference eqn for a 2-D IIR filter must be recursively computable.

$$X(z_1, z_2) = \sum_{n_1} \sum_{n_2} x(n_1, n_2) z_1^{-n_1} z_2^{-n_2}$$

$$X(\omega_1, \omega_2) = X(z_1, z_2)|_{z_1 = e^{j\omega_1},\ z_2 = e^{j\omega_2}}$$

Eg

$$x(n_1, n_2) = a^{n_1} b^{n_2} u(n_1, n_2) = a^{n_1} u(n_1).b^{n_2} u(n_2)$$

$$X(z_1, z_2) = \sum_{n_1=0} (a z_1^{-1})^{n_1} \sum_{n_2=0} (b z_2^{-1})^{n_2}$$

$$= \frac{1}{1 - a z_1^{-1}} . \frac{1}{1 - b z_2^{-1}}$$

$X(z_1, z_2)$ converges for $|z_1| > |a|$ and $|z_2| > |b|$

In 1-D, poles of $X(z)$ are points in complex z-plane

In 2-D, poles of $X(z_1, z_2)$ are surfaces in 4-D complex $z_1, z_2$ plane

In above example, pole surfaces are seperable:

$z_1 = a$ for any $z_2$

$z_2 = b$ for any $z_1$

77

**2-D IIR Filters**

We need **Recursive Computability**, which permits calculation of o/p signal samples recursively; ie, in terms of adjacent previously calculated o/ps. Eg:

$$y(n_1, n_2) = a_1 y(n_1 - 1, n_2) + a - 2y(n_1 + 1, n_2) + x(n_1, n_2)$$

$$y(0, 0) = a_1 y(-1, 0) + a_2 y(1, 0) + x(0, 0)$$

$$y(1, 0) = a_1 y(0, 0) + a_2 y(2, 0) + x(1, 0)$$

This is impossible as computing y(0,0) needs y(1,0), but y(1,0) needs y(0,0)

[All FIR filters are recursively computable]

In 2D image processing, we can process pixels in any direction, so causality isn't an issue

78

**General diff. Eqn:**

$$\sum_{k_1}\sum_{k_2} a(k_1,k_2)y(n_1-k_1,n_2-k_2) = \sum_{r_1}\sum_{r_2} b(r_1,r_2)x(n_1-r_1,n_2-r_2)$$

The coeff. masks [a] and [b] are given over $R_A, R_B$

To compute o/p, normalise a(0,0) = 1 and rewrite:

$$y(n_1,n_2) = -\sum\sum a(n_1-k_1,n_2-k_2)y(k_1,k_2)+\sum\sum b(n_1-k_1,n_2-k_2)x(k_1,k_2)$$

The recursions are performed as follows:

1. At any point $(n_1,n_2)$, the i/p mask [b] is positioned over i/p array. I/p values mulitplied by b values and summed.

2. Simultaeneously, the o/p mask [a] covers the already computed o/p values, weights these values by the a's and sums them

3. Results of steps 1 and 2 are added to give $y(n_1,n_2)$

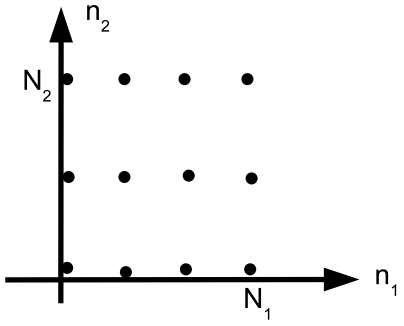
4. Masks then shifted by one position and repeat steps 1-3.

79

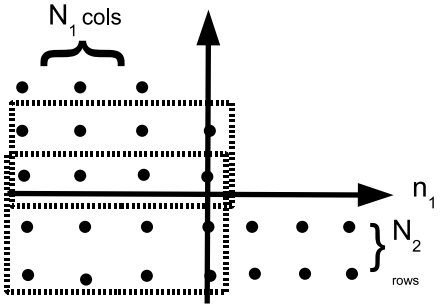
**Conditions for recursive Computability**

IIR system is recursively computable if the o/p mask has wedge support ( i.e, its values lie in region bounded by 2 lines from origin at less than 180 degrees).

A simpler sufficient condition is that coefs $a(k_1, k_2)$ have 1st quadrant support:




A recursive eqn also needs initial conditions. The i.c mask for above [a] is:




The i.c.s within the L-shaped region of $N_1$ cols and $N_2$ rows must be specified to allow recursive computation of $y(n-1, n-2)$ for $n_1 > 0$ and $n_2 > 0$.

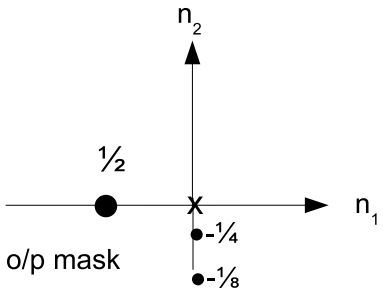
For example, to calculate y(0,0) needs i.c.s in dotted top rectangle,then y(0,1) is calculated using y(0,0) and i.c.s in bottom rectangle.

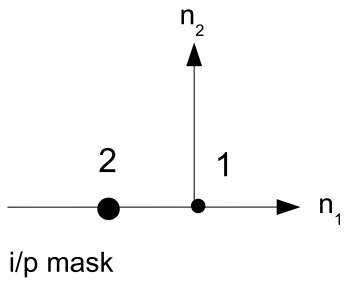
80

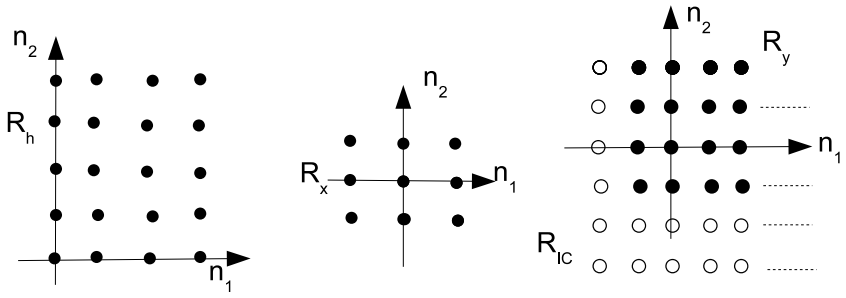
**IIR Eg:**

$$H(z_1, z_2) = \frac{Y(z_1, z_2)}{X(z_1, z_2)} = \frac{1 + 2z_1^{-1}}{1 - \frac{1}{2}z_1^{-1} + \frac{1}{4}z_2^{-1} + \frac{1}{8}z_2^{-2}}$$

$$y(n_1, n_2) = x(n_1, n_2) + 2x(n_1 - 1, n_2) + \frac{1}{2}y(n_1 - 1, n_2) - \frac{1}{4}y(n_1, n_2 - 1) - \frac{1}{8}y(n_1, n_2 - 2)$$




Let

$$x(n_1, n_2) = \begin{cases} 1, & -1 \le n_1, n_2 \le 1 \\ 0 & otherwise \end{cases}$$

$$y(-1,-1) = x(-1,-1) + 2x(-2,-1) + \frac{1}{2}y(-2,-1) - \frac{1}{4}y(-1,-2) - \frac{1}{8}y(-1,-3) = 1$$

$$y(-1,0) = x(-1,0) + 2x(-2,0) + \frac{1}{2}y(-2,0) - \frac{1}{4}y(-1,-1) - \frac{1}{8}y(-1,-2) = \frac{3}{4}$$

$$y(0,-1) = x(0,-1) + 2x(-1,-1) + \frac{1}{2}y(-1,-1) - \frac{1}{4}y(0,-2) - \frac{1}{8}y(0,-3) = 3\frac{1}{2}$$

80b

**Stability of IIR Filters**

A bi-variate polynomial can't, in general, be factored into lower order polynomials. Also poles of $H(z_1, z_2)$ are surfaces in 4-D complex $(z_1, z_2)$ space so stability tests are not simple extensions of 1-D results

But spatial domain condition for BIBO stability is:

$$\sum_{n_1} \sum_{n_2} |h(n_1, n_2)|^2 < \infty$$

- however, it's difficult to translate to tfm domain.

We need consider only

$$H(z_1, z_2) = \frac{1}{A(z_1, z_2)}$$

Stability tests for these IIR filters have been developed as De Carlo - Strintzis theorem:

A 2-D IIR filter with 1st quadrant o/p support is stable if and only if:

1. $A(z_1, z_2) \neq 0 \ for \ |z_1| = 1, \ |z_2| = 1$

2. $A(z_1, 1) \neq 0 \ for \ |z_1| \geq 1$

3. $A(1, z_2) \neq 0 \ for \ |z_2| \geq 1$

81

**2-D FIR Filters**

Simpler than IIR as $a(k_1, k_2) = 0$:

$$y(n_1, n_2) = \sum\sum b(n_1 - k_1, n_2 - k_2)x(k_1, k_2)$$

$$\Rightarrow b(n_1, n_2) = h(n_1, n_2)$$

FIR filter has zero phase if $h(n_1, n_2)$ is symmetrical If $h(n_1, n_2)$ is real and even, then $H(\omega_1, \omega_2)$ is a real, even fn:

$$H(\omega_1, \omega_2) = H(-\omega_1, -\omega_2)$$

Since spatial causality is not a constraint it is easy to make

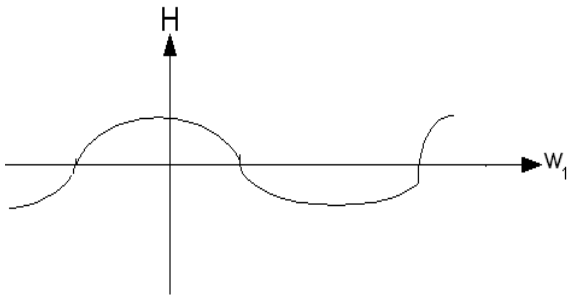
$$h(-n_1, -n_2) = h(n_1, n_2)$$

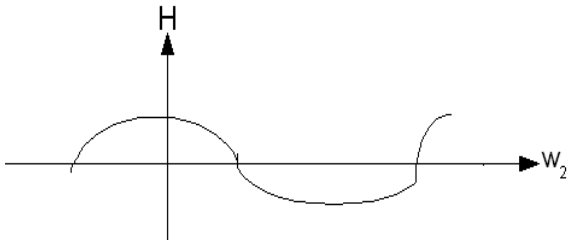
E.g 1.

$$h(n_1, n_2) = \delta(n_1 - 1, n_2) + \delta(n_1 + 1, n_2) + \delta(n_1, n_2 - 1) + \delta(n_1, n_2 + 1)$$

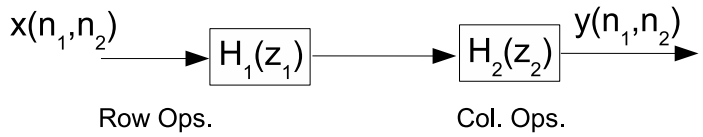
$$H(z_1, z_2) = (z_1^{-1} + z_1) + (z_2^{-1} + z_2)$$

$$H(\omega_1, \omega_2) = 2cos\ \omega_1 + 2cos\ \omega_2$$



**Seperable 2-D filters** can be reduced to two 1-D filters in tandem:




So design of seperable 2-D FIR filter reduces to the design of 1-D FIR filters by, e.g window method, freq-sampling or Parks-McClellan algorithm.

Eg 2

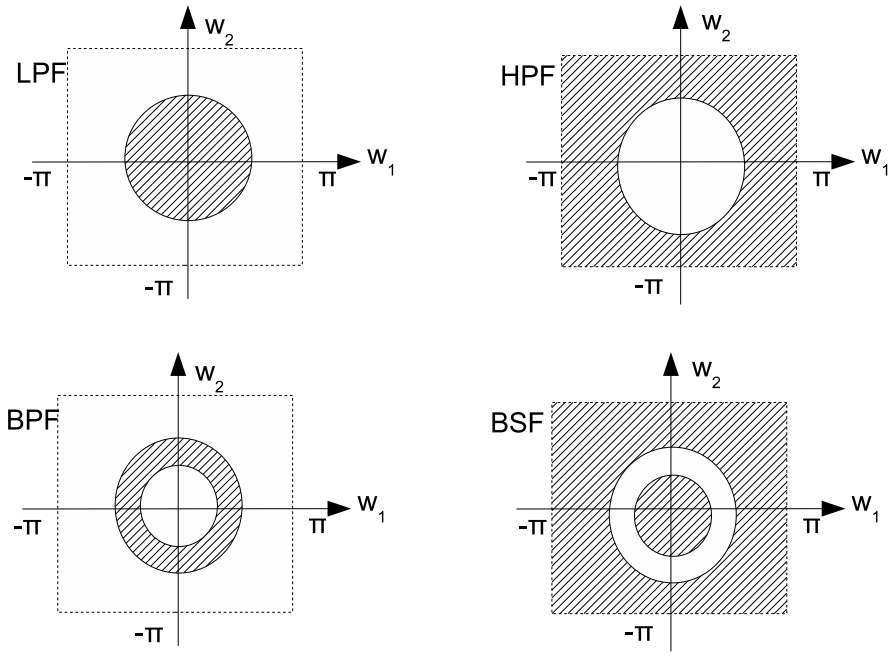
$$H(z_1, z_2) = (z_1^{-1} + 2 + z_1)(z_2^{-1} + 2 + z_2)$$

$$= z_1^{-1}z_2^{-1} + 2z_1^{-1} + z_1z_2^{-1} + 2z_2^{-1} + 4 + 2z_2 + z_2z_2^{-1} + 2z_1 + z_1z_2$$

$$H(\omega_1, \omega_2) = (2 + 2cos\ \omega_1) + (2 + 2cos\ \omega_2) = (4cos\ .5\omega_1 cos\ .5\omega_2)^2$$

83

**Types of 2-D filter (circularly symmetric)**




**Non-seperable 2-D FIR Filters** can be designed by simple extension of 1-D techniques

E.g Windowing Method:

1. Specify $H(\omega_1, \omega_2)$

2. Compute $h(n_1, n_2)$ by inverse FFT

3. Truncate $h(n_1, n_2)$ by window $w(n_1, n_2)$: $h(n_1, n_2) = h(n_1, n_2)w(n_1, n_2)$

4. Pad $h(n_1, n_2)$ with 0s and compute $H'(k_1, k_2)$ via FFT

84

**"'Frequency"' transformation method** does not have a 1-D counterpart but is practical method of designing 2-D filters from 1-D:

1st design 1-D prototype 0-phase FIR Filter of form:

$$H(\omega) = \sum_{n=-M}^{M} h(n)e^{-jn\omega} = h(0) + \sum_{n=1}^{M} 2h(n)cos\ \omega n = \sum_{n=0}^{M} a_n cos\ \omega n$$

which can be shown to be $= \sum_{n=0}^{M} b_n (cos\ \omega)_n$

2-D freq response $H(\omega_1, \omega_2)$ is obtained by substituting 2-D Transformation $T(\omega_1, \omega_2)$ for $cos\ \omega$:

$$H(\omega_1, \omega_2) = H(\omega)|_{cos\omega = T(\omega_1, \omega_2)} = \sum_{n=0}^{M} b_n (T(\omega_1, \omega_2))^n$$

Where $T(\omega_1, \omega_2)$ is finite-extent, 0-phase, real, even fn, which can be expressed as:

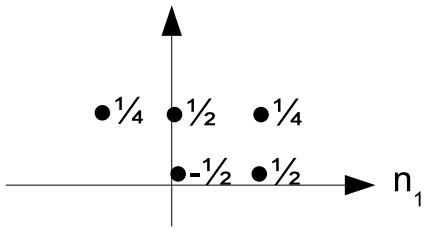
$$T(\omega_1, \omega_2) = \sum\sum c(n_1, n_2) cos(\omega_1 n_1 + \omega_2 n_2)$$

An eg of $T(\omega_1, \omega_2)$ often used in practice is the Mc Clellan Transformation:

$$T(\omega_1, \omega_2) = -.5 + .5cos\ \omega_1 + .5cos\ \omega_2 + .25cos(\omega_1 + \omega_2) + .25cos(\omega_1 - \omega_2)$$

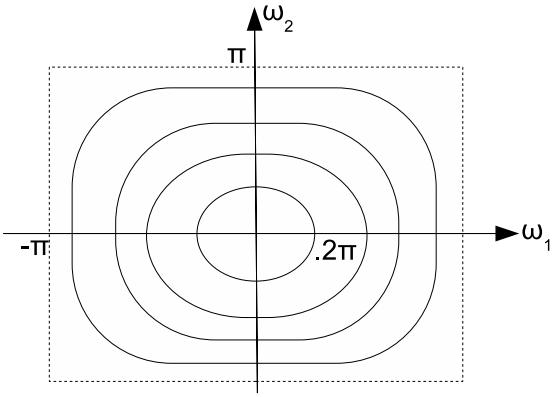
$$= -.5 + .5cos\ \omega_1 + .5cos\ \omega_2 + .5cos\ \omega_1\ cos\ \omega_2$$

Then $c(n_1, n_2)$ is:

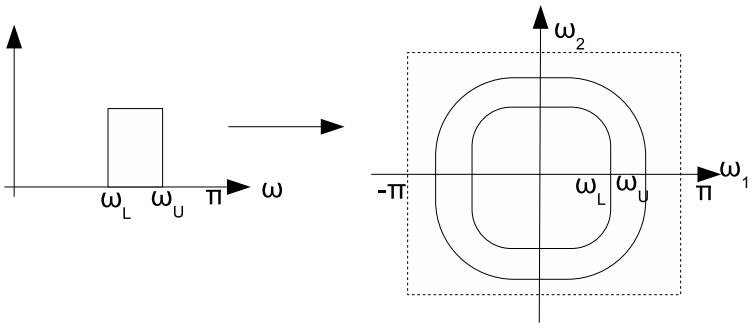



85

For McClellan Tfm, set of contours for $cos\ \omega = T(\omega_1, \omega_2)$ is:




$T(\omega_1, \omega_2)$ maps 1-D frequency response to 2-D Eg 1. BPF:




E.g 2. Apply transformation method to binomial LPF:

$$H(\omega) = 6 + 8cos\ \omega + 2cos2\ \omega = 4 + 8cos\ \omega + 4cos^2\ \omega$$

$$\Rightarrow H(\omega_1, \omega_2) = 4 + 8T(\omega_1, \omega_2) + 4[T(\omega_1, \omega_2)]^2$$

For $T(\omega_1, \omega_2) = .5cos\ \omega_1 + .5cos\ \omega_2 + .5cos\ \omega_1 cos\ \omega_2 - .5$,

$$H(\omega_1, \omega_2) = 1+2cos\omega_1+2cos\omega_2+4cos\omega_1 cos\omega_2+cos^2\omega_1+cos^2\omega_2+2cos\omega_1 cos^2\omega_2+2cos^2\omega_1 cos\omega_2+cos^2\omega_1 cos^2\omega_2$$

$$= (1 + cos\ \omega_1)^2(1 + cos\ \omega_2)^2 = 16(cos\ .5\omega_1)^4(cos\ .5\omega_2)^4$$

**Implementation of 2-D FIR Filters**

(a) Direct computation of convolution sum:

$$y(n_1, n_2) = \sum\sum x(n_1 - k_1, n_2 - k_2)h(k_1, k_2)$$

This needs N muls and N adds for each o/p point, where N = no. of non-0 coeffs of $y(n_1, n_2)$. E.g a 512 x 512 image filtered by 10 x 10 filter needs 26,214,400 mult/adds

As for 1-D FIR 0-phase filters, computation reduced 50% by exploiting symmetry of h (n1,n2)

(b) Using FFT: $Y(\omega_1, \omega_2) = H(\omega_1, \omega_2)X(\omega_1, \omega_2)$

For 512 x 512 image with 10 x10 filter this needs inverse DFT $\geq$ (512+10-1)x(512+10-1).

Use of FFT for this will need less computation (approx $\frac{N^2}{2}log_2(N^2)x2 + N^2 \approx 5,000,000$ mults)than direct convolution, but needs all $x(n_1, n_2)$, $H(\omega_1, \omega_2)$ and $X(\omega_1, \omega_2)$ to be stored.

An alternative is to divide data into blocks and do smaller FFTs. As FFT gives circular convolution, overlap-add or overlap-save method can be used.

87

**Linear vs Circular Convolution**

Eg (1-D)

Linear:

$$x_1(n) = x_2(n) = [3, 2, 1],\ x_3(n) = x_1(n) * x_2(n) = \sum_m x_1(m)x_2(n-m)$$

$$x_3(0) = \sum x_1(m)x_2(0-m) = 9$$

$$x_3(1) = \sum x_1(m)x_2(1-m) = 12$$

$$x_3(2) = 10,\ x_3(3) = 4,\ x_3(4) = 1,\ x_3(n) = 0, n > 4$$

Circular:

$$x_{3p}(n) = \sum_{m=0}^{N-1} x_1(m)x_2(n-m, (mod\, N)),\ n = 0...N-1$$

$$x_{3p}(0) = \sum_{m=0}^{2} x_1(m)x_2(0-m, (mod\, 3)) = 13$$

$$x_{3p}(1) = \sum_{m=1}^{2} x_1(m)x_2(1-m, (mod\, 3)) = 13$$

$$x_{3p}(2) = \sum_{m=2}^{2} x_1(m)x_2(2-m, (mod\, 3)) = 10$$

$$x_{3p}(3) = x_{3p}(0),\ x_{3p}(4) = x_{3p}(1)...repeating$$

If $x_1$ and $x_2$ are padded with 2 zeros to give $\acute{x}_1 = \acute{x}_2 = 3, 2, 1, 0, 0$, then 1 period of $\acute{x}_1 \circledast \acute{x}_2 = x_3$ - same as linear convolution.

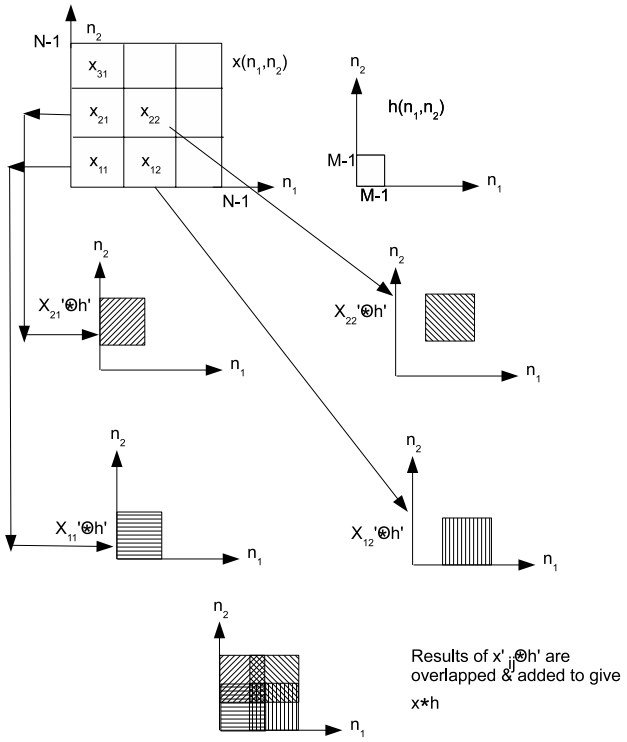
88

**2-D Overlap Add**

Image $x(n_1, n_2)$ divided into $L_1 L_2$ segments $x_{ij}(n_1, n_2)$:

$$x(n_1, n_2) = \sum_{i=1}^{L_1} \sum_{j=1}^{L_2} x_{ij}(n_1, n_2)$$

$$x(n_1, n_2) = \sum_{i=1}^{L_1} \sum_{j=1}^{L_2} x_{ij}(n_1, n_2) * h(n_1, n_2) :$$




Since $x_{ij}$ is much smaller than x, $\acute{x}_{ij} \circledast \acute{h}$ can be computed by much smaller DFT, IDFT and storage.

89