The z-transform: motivation

Recall that you have seen a number of transforms:

- ♦ Fourier transform: continuous -> continuous
- ♦ Laplace transform: continuous -> continuous
  - * causal inputs, defined on $[0,\infty)$
- ♦ Fourier series: continuous -> discrete
- ♦ Discrete Fourier transform: discrete -> discrete

All share *the convolution property* for *linear, shift invariant* operations:

Eg. for continuous signals:

- ♦ the ability to represent filtering operations on continuous signals by a convolution:

$$h(t) = \int dt'\, g(t-t')\, f(t')$$

filter $g(t-t')$

- ♦ or, in frequency space by a simple product:

$$h(\omega) = g(\omega)\, f(\omega)$$

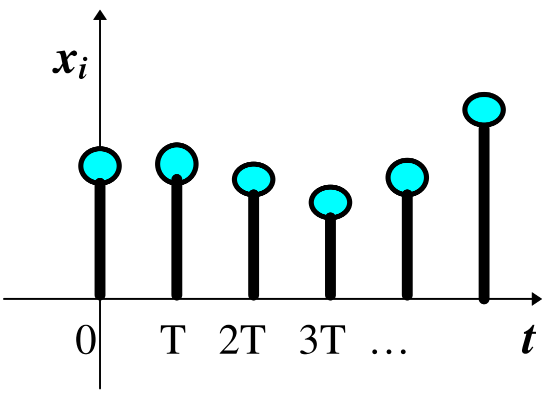
  is very useful

- ♦ cf: the *local* (stationary, shift invariant) and *non-local* (non-stationary) filtering operations we saw in low-level image processing.

Definition of the z-transform

The **z-transform** produces an analytic function for **discrete time series** or discretely sampled signals.

- Suppose we have a sequence of time series data, $x[i]$ for $i = 0,1,2,\ldots$ at regular time intervals, T



Note that this is causal: the signal is zero for t<0.

The **z-transform** of the sequence is defined as:

$$X(z) = x[0] + x[1]z^{-1} + x[2]z^{-2} + x[3]z^{-3} + \ldots..$$

$$= Z\left[\{x[n]\}\right] = \sum_{i=0}^{\infty} x[i]z^{-i}$$

Each power of z^{-1} tags the position of the sample in the sequence.

- In general, if the signal is non-causal, the two-sided z-transform may be used (though we shall not pursue it here): $X(z) = \sum_{i=-\infty}^{\infty} x[i]z^{-i}$

The delay operator

- multiplication by z^{-1} denotes a delay of one sampling interval, T. This *suggests*:
    - if a sequence $x[n]$ is represented by the transform $X(z)$, then a similar sequence $x[n-k]$ delayed by k sampling intervals has the transform $z^{-k}X(z)$.
    - We shall consider this more formally later.

Examples of z-transforms:

Calculating the z-transform for finite sequences is easy, eg:

$$X(z) = 1 + 1.5z^{-1} + 1z^{-2} + 2z^{-3}$$

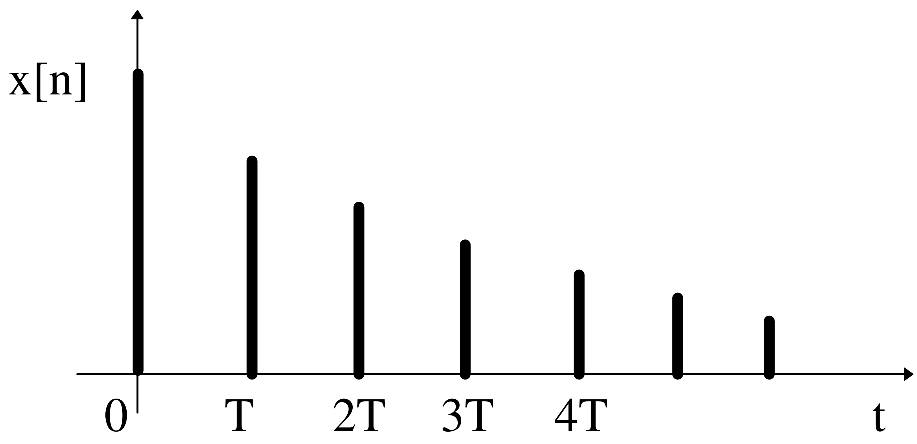
$$X(z) =$$

$$X(z) =$$

Calculating the z-transform

It is also easy to calculate the z-transform, for some simple signals:

- eg. suppose we have an exponentially decaying signal, $e^{-\lambda t}$, uniformly sampled:



- sequence of discretely sampled signals

$$x[n] = 1, e^{-\lambda T}, e^{-2\lambda T}, e^{-3\lambda T}, e^{-4\lambda T}, \ldots.$$

=>

$$X(z) = 1 + e^{-\lambda T} z^{-1} + e^{-2\lambda T} z^{-2} + e^{-3\lambda T} z^{-3} + \ldots..$$

$$= \frac{1}{1 - e^{-\lambda T} z^{-1}}$$

on using the formula for sum of a geometric progression, provided that

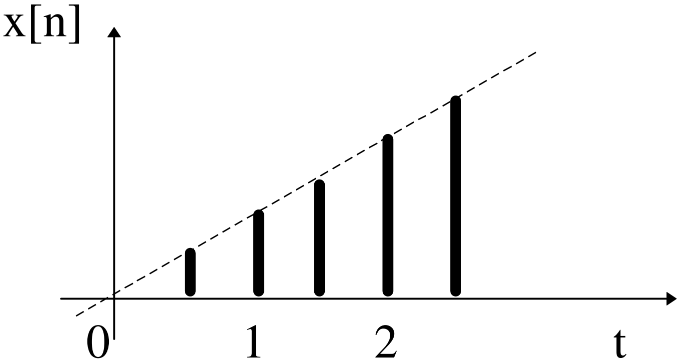
$\left| z^{-1} e^{-\lambda T} \right| < 1$, ie. $|z| > e^{-\lambda T}$ [Note this for later.]

More complicated transforms

Usually look up the z-transform from tables.

- some examples are given below together with the Laplace transform of the continuous signal for comparison:

- eg. a unit ramp, sampled every 0.5 secs



$$X(z) = \frac{z}{2(z-1)^2}$$

Properties of the z-transform

The z-transform is obviously **linear**:

- $Z[\alpha x[n] + \beta y[n]] = \alpha Z[x[n]] + \beta Z[y[n]]$

In addition, we note:

- the **shifting property**:

$$Z[x[n+1]] = \sum_{i=0}^{\infty} x[i+1] z^{-i} = z \sum_{i=1}^{\infty} x[i] z^{-i}$$
$$= zX(z) - zx[0]$$

ie. if x[n] is **causal** and vanishes for all n<0, =>

$Z[x[n-k]] = z^{-k} Z[x[n]]$

Similarly, from the definition

- **scale** change

$$Z\left[e^{-\lambda nT} x[n]\right] = \sum_{i=0}^{\infty} e^{-\lambda iT} x[i] z^{-i}$$
$$= z \sum_{i=1}^{\infty} x[i] \left(e^{\lambda T} z\right)^{-i} = X\left(e^{\lambda T} z\right)$$

An example of this can be seen above if we note that the transform of a constant signal is (for $|z| > 1$):

$$X(z) = 1 + z^{-1} + z^{-2} + z^{-3} + \ldots\ldots = \frac{1}{1 - z^{-1}} = \frac{z}{z-1}$$

Discrete differentiation

- recall, for continuous functions, we could say

$$f'(t) = \lim_{a \to 0} \left(\frac{f(t) - f(t-a)}{a} \right)$$

"backwards differentiation"

- for discrete time series, we analogously use the *difference*:

$$d[n] = \frac{f[n] - f[n-1]}{T} \qquad =>$$

$$D(z) = \frac{F(z) - z^{-1}F(z)}{T} \qquad \text{, ie:}$$

$$\frac{D(z)}{F(z)} = \text{ discrete differential operator}$$

$$= \frac{1 - z^{-1}}{T}$$

$$= \frac{z-1}{Tz}$$

Thus, for the k'th backward difference:

$$Z\left[d^k x[n]\right] = \left(\frac{z-1}{Tz} \right)^k X(z)$$

Discrete integration

- recall: $$y(t) = \int_0^t dt' f(t')$$

For discrete time series:

$$y[n] = \sum_{i=1}^{n} f[i]T \qquad (1)$$

"backwards integration"

=> **recursively**, for a sequence

$$y[n+1] = y[n] + f[n+1]T$$

- z-transform =>

$$zY(z) = Y(z) + zF(z)T$$

$$\Rightarrow \quad Y(z)(z-1) = zF(z)T$$

$$\Rightarrow \quad Y(z) = \frac{zT}{z-1} F(z)$$

- ie, discrete integration operator $= \dfrac{zT}{z-1}$
- sometimes known as a "summation operator"

Note: it is the inverse of the discrete, (*backwards*) differentiation operator on the previous page.

- Similar operators may be defined for discrete *forward* differentiation and *forward* integration.

Differentiation and integration

	integration	**differentiation**
continuous		$\lim_{a\to 0}\left(\frac{x(t+a)-x(t)}{a}\right)$
discrete backward		$d[n]=\frac{x[n]-x[n-1]}{T}$
discrete forward		$d[n]=\frac{x[n+1]-x[n]}{T}$

Smoothing

Suppose we were averaging a signal by taking the mean of successive samples in the sequence:

$$y[n] = \frac{x[n] + x[n-1]}{2} \tag{2}$$

⇨for the z-transform:

$$Y(z) = \frac{X(z) + z^{-1}X(z)}{2}$$

$$= \frac{(1+z^{-1})}{2} X(z)$$

ie: the transfer function for such a moving average process is:

$$G(z) = \frac{z+1}{2z} \tag{3}$$

- this moving average is a simple example of a *discrete filtering* process

Exercise:

- Show how the above average G(z) may be calculated from the backward integration or summation operator.
- **Hint**: write (2) as the difference of two sums of the form (1) given on page 8.

Convolution filtering

Recall: for continuous signals

$$y(t) = \int_0^t dt' g(t-t')x(t') = \int_0^t dt'' g(t'')x(t-t')$$

In discrete form, for a sequence:

$$y[n] = \sum_{i=0}^{n} g_i x[n-i]$$

⇨z-transform *convolution theorem*:

$$Y(z) = \sum_{i=0}^{n} g_i z^{-i} X(z)$$

ie, $$Y(z) = G(z)X(z)$$

where the transfer function, $G(z)$, is the z-transform of the sequence of filter coefficients,

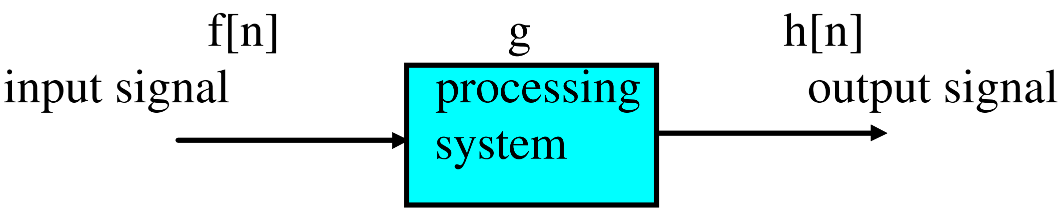
$$G(z) = \sum_{i=0}^{n} g[i] z^{-i} \qquad (4)$$

Exercise:

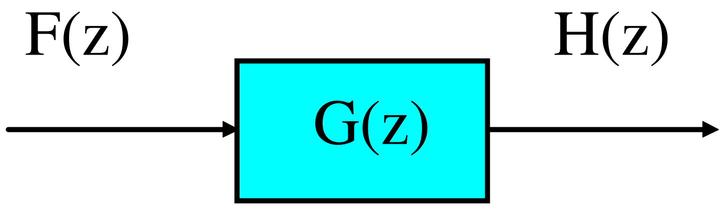
- Write down the filter coefficients for the two stage (2-tap) moving average filter (2) and hence obtain the transfer function $G(z) = \dfrac{z+1}{2z}$, as given in (3), from formula (4) above.

Transfer function

We often use the z-transform to represent what a *linear* process does to digital signals.



=> if we z-transform



ie: $$H(z) = G(z)F(z)$$

and: Transfer function $= \dfrac{output}{input}$, ie

$$G(z) = \frac{H(z)}{F(z)}$$

Note: the *transfer function*, $G(z)$,is the output obtained for a *standard, unit input pulse*:

ie: $f[n] = 1,0,0,0,0.....$, =>

$F(z) = 1$, =>

$H(z) = G(z)$.

Recap: The z-transform

For a sequence of signals

$x[n] = x_0, x_1, x_2, x_3, ...$

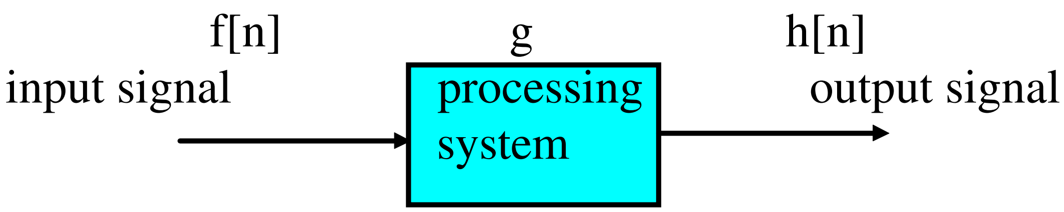
at regular time intervals, T, the z-transform is defined as:

$$X(z) = x[0] + x[1]z^{-1} + x[2]z^{-2} + x[3]z^{-3} + ...$$

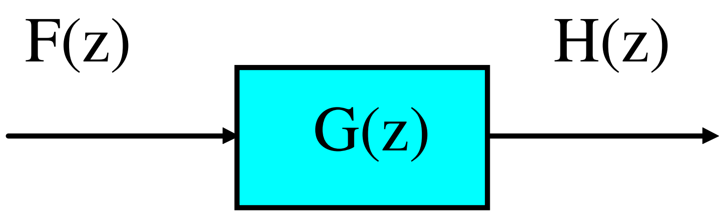
$$= \sum_{i=0}^{\infty} x[i]z^{-i}$$

Each power of z^{-1} tags the position of the sample in the sequence.

The transfer function



represented as a z-transform:



=> $H(z) = G(z)F(z)$

Examples:

- discrete differentiation ("backwards")

$$G(z) = \frac{z-1}{Tz}$$

- discrete integration ("backwards")

$$G(z) = \frac{zT}{z-1}$$

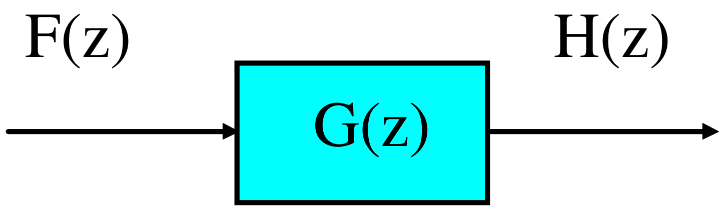
- smoothing (2-point moving average)

$$G(z) = \frac{z+1}{2z}$$

General form for an FIR filter

- FIR filter: $y[n] = \sum_{i=0}^{n} g_i x[n-i]$ (5)

$$G(z) = \sum_{i=0}^{n} g[i] z^{-i}$$



=> $H(z) = G(z)F(z)$

Recursive systems (feedback)

If the output at time *t* depends not only on the input as above in (5) but also on the output at a number of preceding sampling times, we have a sequence of the form:

$$y[k] = \sum_{j=1}^{m} a_j y[k-j] + \sum_{i=0}^{n} b_i x[k-i] \quad (6)$$

It is **causal**, since y[k] depends on *previous* values of y and *only* on x[k] and its *predecessors*.

The z-transform of (6) is:

$$Y(z) = \sum_{j=1}^{m} a_j z^{-j} Y(z) + \sum_{i=0}^{n} b_i z^{-i} X(z) \quad (7)$$

$$\Rightarrow \quad Y(z)\left(1 - \sum_{j=1}^{m} a_j z^{-j}\right) = X(z) \sum_{i=0}^{n} a_i z^{-i} \quad (8)$$

so the transfer function is:

$$G(z) = \frac{\sum_{i=0}^{n} b_i z^{-i}}{1 - \sum_{j=1}^{m} a_j z^{-j}} \quad (9)$$

Example

Consider a sequence representing a decaying average or "memory" function:

$$y[n] = \alpha y[n-1] + x[n] \qquad \text{(i)}$$

♦ the z-transform is:

$$Y(z) = \frac{1}{1-\alpha z^{-1}} X(z) \qquad \text{(ii)}$$

=> a transfer function

$$G(z) = \frac{z}{z-\alpha} \qquad \text{(iii)}$$

cf. earlier exponential decay example, page 4

Pole

♦ the transfer function $G(z)$ above has a pole at $z = \alpha$ at which the denominator vanishes and $G(z) \to \infty$

♦ If we have a standard unit input pulse:

$$x[n] = 1,0,0,0,0,......$$

♦ and, before the pulse, the output $y = 0$, then we see from the recursive memory equation (i) that:

$$y[n] = 1, \alpha, \alpha^2, \alpha^3, \alpha^4, ...$$

which decays with time if $|\alpha| < 1$ (*stable*), and grows if $|\alpha| > 1$ (*unstable*).

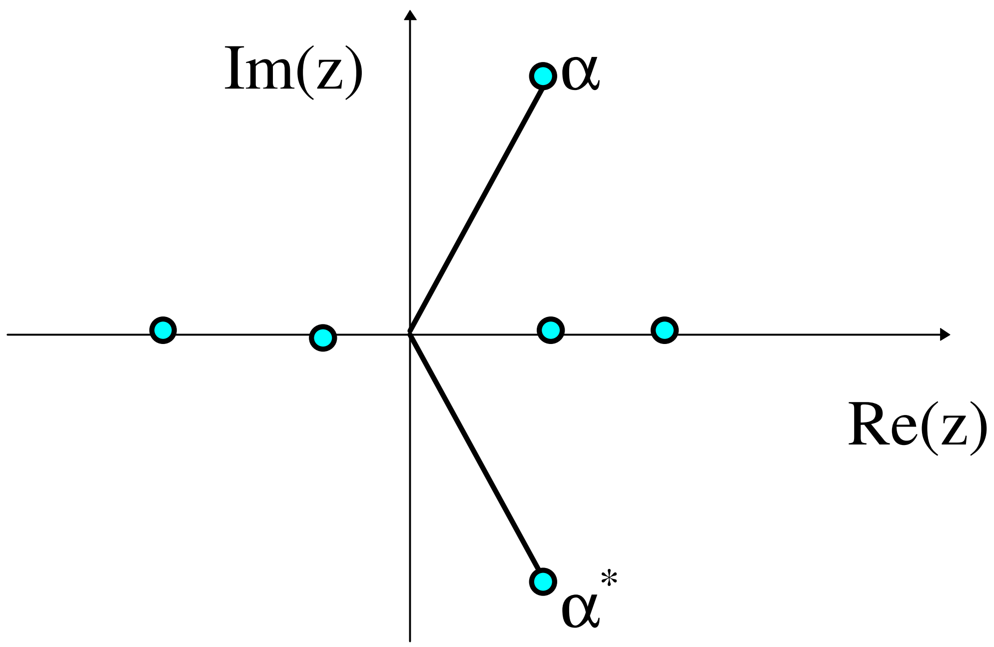
Poles and stability

A digital system with transfer function of the form (9) has **poles** at:

$$z^m - \sum_{j=1}^{m} a_j z^{m-j} = 0 \qquad (10)$$

where the denominator of (9) vanishes.

- since (9) is a polynomial equation in the variable z with real coefficients, its roots are real or occur in complex conjugate pairs, α and α^*
- the poles may thus be plotted in the z-plane:



Interpretation

Real poles may be interpreted as in the memory function example (pages 4 & 16):

(i) if the pole lies between $z = -1$ and $z = +1$, ie. $|\alpha| < 1$, the sequence will die away with time - the system is **stable**

(ii) if the pole lies in $0 < z < 1$, we may think of the response as a sampled version of a decaying exponential such that $\alpha = e^{-\lambda T}$

(iii) if the pole lies in the region $-1 < z < 0$, the response to the pulse is oscillatory with alternating positive and negative values

Complex poles with $\alpha = |\alpha| e^{\pm i\phi}$ represent a transient response whose magnitude changes by $|\alpha|$ between sampling intervals and phase by ϕ.

- ♦ if $|\alpha| < 1$ at a complex pole, the transient may be thought of as a sampled version of a decaying sinusoid:
  - of frequency ω such that $\omega T = \phi$, => in each period T, phase advances by ϕ
  - with decay rate λ such that $e^{-\lambda T} = |\alpha|$

Stability in general

A linear, time-invariant system is said to be **stable** if and only if its impulse response (9) is **bounded** above and below, ie iff:

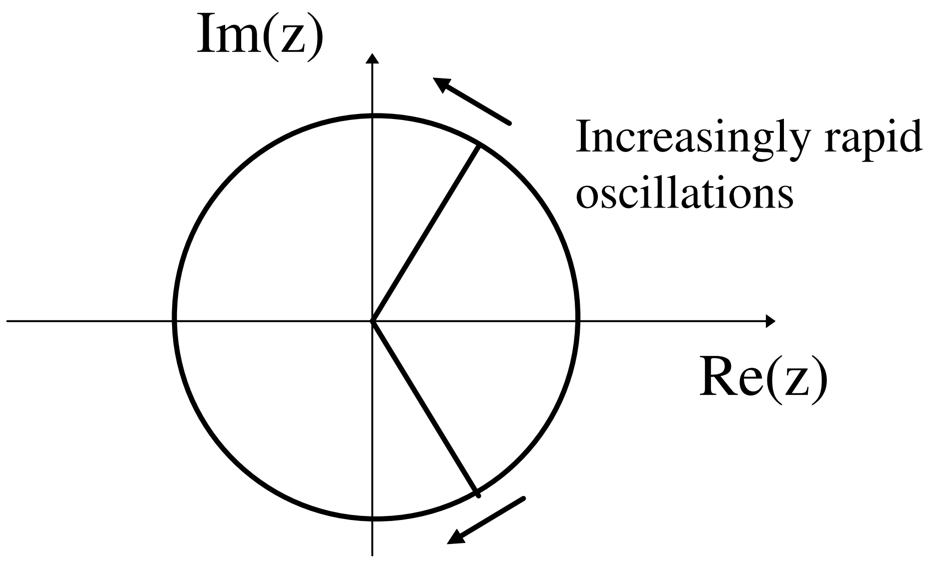
$$\sum_{i=0}^{\infty} |g[i]| < \infty \qquad (11) \qquad =>$$

(i) poles inside the unit circle are **stable**

(ii) poles outside the unit circle lead to **instability**

(iii) poles on the unit circle correspond to oscillatory behaviour, the frequency of the oscillation increasing from 0 at $z = 1$ to

(iv) a complete alternation in sign at $z = -1$.



Note: poles at $z = 0$ are unimportant as they simply correspond to a delay of one interval, T. (Recall effect of multiplication by z^{-1}.)

Complex poles: examples

Note: the maximum frequency is $\frac{1}{2T}$. Thanks to the sampling theorem, higher frequency oscillations than an alternation of sign are impossible.

Steady state and initial value

A simple pole at $z=1$, such as:

$$G(z)=\frac{1}{z-1} \qquad \Rightarrow$$

an input impluse generates a constant output. Thus:

- **provided all** the poles of $(z-1)Y(z)$ lie **inside** the unit circle,
- we can pick out the steady state, long-term behaviour of a system's output by evaluating

$$\lim_{z\to 1}\left[(z-1)Y(z)\right]=\lim_{k\to\infty}\left[y[k]\right]$$

Note: use of the output $Y(z)$ and the proviso required in order to ensure that the limit exists.

Initial value:

It follows from the definition:

$$Y(z)=y[0]+y[1]z^{-1}+y[2]z^{-2}+\ldots\ldots$$

that the **initial value**, $y[0]=\lim_{z\to\infty}\left[Y(z)\right]$.

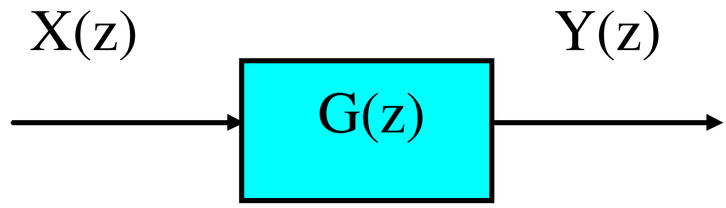
Example:

- Consider the initial value and long term behaviour of y[n], if $Y(z)=\dfrac{z^2-1}{z(z^2+z+1)}.\dfrac{z}{z-1}$.

Discrete linear systems

So far we have concentrated on a single linear system:

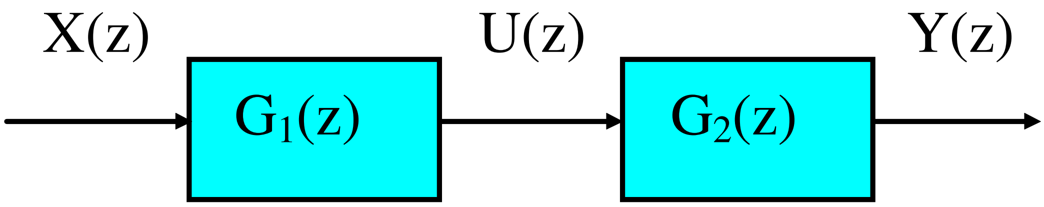
- eg: for signal processing



with transfer function $G(z) = \dfrac{Y(z)}{X(z)}$

- shall now consider systems with a number of such components combined in various ways:

Series combination



- since: $U(z) = G_1(z)X(z)$

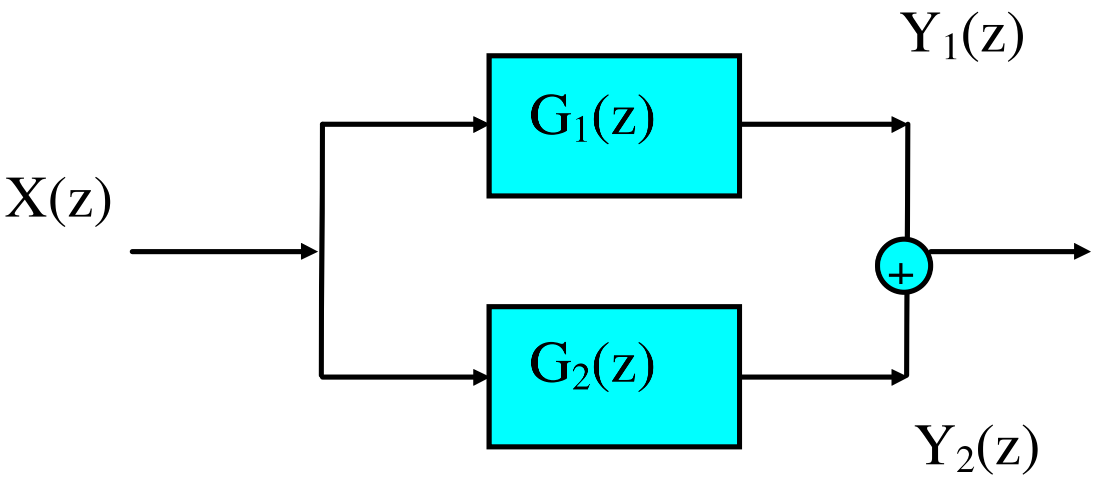
$$Y(z) = G_2(z)U(z)$$
$$= G_2(z)G_1(z)U(z)$$

- the combined transfer function is:

$$G(z) = G_2(z)G_1(z) \qquad (12)$$

Parallel combination

Similarly, if we drive two processing systems in parallel with the same input and add their outputs:



Since:

$$Y_1(z) = G_1(z)\,X(z)$$
$$Y_2(z) = G_2(z)\,X(z) \quad ,$$

and $\quad \mathrm{Y(z)} = \mathrm{Y}_1(z) + Y_2(z)$

$$\Rightarrow \; Y(z) = \left[G_1(z) + G_2(z)\right]X(z)$$

ie, the combined transfer function is:

$$G(z) = G_2(z) + G_1(z) \qquad (13)$$

Exercise:

- Use the above to show what happens when smoothing (eg by a 2-tap averaging filter) and differentiation of a discrete signal x[n] are combined (a) in series, (b) in parallel, and (c) recursively in a feedback loop with gain, α.

Inverting the z-transform

The discrete sequence $x[n]$ may formally be obtained from $X(z)$ by the inverse z-transform:

$$x[n] = Z^{-1}[X(z)] \qquad (14)$$

In practice, there are several ways of performing the inverse transform, for example:

(i) from the **power series** in z^{-1}, eg by expanding X in a Taylor series in $\zeta = z^{-1}$

$$X(z) = X(\zeta) = a_0 + a_1\zeta + a_2\zeta^2 + \ldots.. \;;$$

whilst, in general, if:

$$X(z) = \frac{a_0 + a_1 z^{-1} + a_2 z^{-2} + \ldots..}{1 + b_1 z^{-1} + b_2 z^{-2} + \ldots..} \qquad (15)$$

the inverse may be obtained by:

(ii) **synthetic long division**

(iii) or, equivalently, **recursively**, using

$$x[n] = a_n - \sum_{i=1}^{n} x[n-i]b_i \quad \text{with } x[0] = a_0$$

(iv) or, if the numerator and denominator of (15) are finite series, by writing X in **partial fractions**.

Example: $X(z) = \frac{Az}{z-1}$ then:

The power series (i) =>

$$X(z) = A\left[1 + z^{-1} + z^{-2} + z^{-3} + z^{-4} + \ldots\right]$$

and, for $n \geq 0$, $x[n] = A, A, A, A, A, \ldots.$

as to be expected from the integration operator.

Exercise:

- Show that synthetic long division (ii) and the recurrence relation (iii) yield similar results.

In order to use partial fractions, (iv), we note that if all the poles of $X(z)$ are first order, with linear factors, such as:

$$X(z) = \sum_{\mu=1}^{M} \frac{C_\mu z}{z - \alpha_\mu} \qquad (16)$$

then each pole generates a response of the form

$C_\mu \left(\alpha_\mu\right)^n$ for $n \geq 0$.

- => for the above $X(z) = \frac{Az}{z-1}$, $C = A$, $\alpha = 1$,

  and, for $n \geq 0$, $x[n] = A, A, A, A, A, \ldots.$

Formulae and the response for more complicated partial fractions may be obtained from tables.

Use of the z-transform

The z-transform is extensively used in digital signal processing, for example, to:

- ♦ design discrete filters,
- ♦ analyse how best to implement them,
- ♦ calculate the frequency response of digital signal processing algorithms,
- ♦ analyse the stability of DSP algorithms, and
- ♦ to analyse the effect of quantisation errors.

For details, see (eg) Ifeachor and Jervis, 1993.

In particular, if a *recursive filter* is used:

- ♦ although it may require fewer coefficients than the corresponding FIR, non-recursive, filter,
  - * => advantageous for applications where high throughput is required
- ♦ without care, such filters can become unstable or their performance degraded.

Exercise:

- ♦ Show how a uniform averaging filter of length 5 may be implemented recursively by means of a filter with just two coefficients. Note that the filter obtained is recursive but not strictly IIR.

Example: a uniform averaging filter

Consider the uniform, top-hat filter of length (or order) n, according to which:

$$y[k] = \sum_{i=0}^{n-1} x[k-i] \qquad \text{(i)}$$

This is easy to implement directly as an FIR filter (provided we take care of the potential numerical overflow), However,

- its computational complexity depends on the size of the filter, which can cause problems in real-time applications where the size of the filter may be programmable or adaptive.

The filter may be written *recursively* as:

$$y[k] = y[k-1] + x[k] - x[k-n] \qquad \text{(ii)}$$

- which requires only two numerical operations, *irrespective* of its length, n.

Exercise:

- Use the z-transform to show that (ii) is formally equivalent to (i) and determine any additional conditions, not specified above, that must be imposed.

Hint: for the last part, consider how to initialise y.

Small generating kernels

Another instance where the z-transform is often useful is in the design of small generating kernels.

- For example, the z-transform may be used to show immediately that repeated convolution of (say) a two point moving average with itself (page 10) leads to a triangular filter, since:

$$[G(z)]^2 = \left(\frac{z+1}{2z}\right)^2$$

$$= \frac{z^2+2z+1}{4z^2} = \frac{1}{4}\left(1+2z^{-1}+z^{-2}\right)$$

More, significantly, since the transfer function for an FIR filter g is a polynomial in z^{-1}, $G(z)$ may be factorized in order to decompose g into a number of small generating kernels, operating sequentially.

- ie; if $\quad G(z) = \prod_{v=1}^{n}\left(1+\beta_v z^{-1}\right) = \prod_{v=1}^{n} G_v(z)$ =>

  $$g = g_1 * g_2 * \ldots * g_n$$

  where each g_v is a small 2-tap filter.

- Note that in order to ensure that all the filter coefficients are real, $G(z)$ is often written in terms of quadratic and linear factors.

Recursive filter design

If we compare the z-transforms of a non-recursive filter

$$G(z) = \sum_{i=0}^{n} g[i] z^{-i} \tag{4}$$

and that of a recursive one, for example:

$$H(z) = \frac{\sum_{i=0}^{n} b_i z^{-i}}{1 - \sum_{j=1}^{m} a_j z^{-j}} \tag{9}$$

we see that, in general, it is not possible to find a set of coefficients a_j and b_i so that (9) will be exactly equal to (4). In practice, it is usual therefore to seek the filter $H(z)$ that best approximates $G(z)$, for example in a least squares sense:

$$\min_{a_j, b_i} \left\{ \sum_{i=0}^{n} \left| g[i] - h[i] \right|^2 \right\} \tag{17}$$

In general, this is a non-linear optimisation problem, because, for example, the a_j determine the poles of $H(z)$, and thus appear in exponents in $h[i]$.

Low-level image processing filters

Deriche (MV&MI, Tokyo, 1987 and IJCV, 1987) gave a procedure for solving the above (see also Canny's MSc thesis).

- In a later paper (IEEE PAMI, 1990), Deriche then showed how a family of second order recursive filters could be used to arrive at computationally efficient implementations of good approximations to Gaussian smoothing filters, and first and second derivative of a Guassian filters (or satisfying Canny's criteria).

In each case, since the spatial filters used in image processing do **not** have to be causal (unless we tie ourselves to a particular scanning strategy), the filters are implemented as *separable* and

- split into two parts, one $h_-[n]$ defined for $n \geq 0$ running left to right along the image rows (say), the other $h_+[n]$ defined for $n < 0$, right to left, with $h[n] = h_+[n] + h_-[n]$ for $n = -\infty, \ldots, \infty$.
- Deriche's filters are of Lorentzian (or exponential) type, with decay constant α chosen to satisfy $\alpha\sigma = 5/\left(2\sqrt{\pi}\right) \approx 1.41$ so that the smoothing filter has the same energy as a Gaussian of width σ.

Deriche's filters

The filters are:

- the smoothing filter $s[n] = k_s(1+\alpha|n|)e^{-\alpha|n|}$ with k_s chosen to give the area under the filter as one,
- the smoothed derivative: $d[n] = k_d n e^{-\alpha|n|}$ with k_d set to produce no response to a uniform signal,
- and the smoothed second derivative: $l[n] = -a\left(1 - k_l \alpha|n|\right)e^{-\alpha|n|}$ with k_l and α chosen to give no response to a uniform signal.
- For details of the constants see Deriche (1990).
- Since the filters are all of Lorentzian or exponential type, they may be implemented exactly as recursive filters and their coefficients obtained from the z-transforms.
- For example, for the first derivative, d_-,

$$D_-(z) = \sum_{i=0}^{\infty} k_d i e^{-\alpha i} z^{-i} = k_d \frac{e^{-\alpha} z^{-1}}{\left(1 - e^{-\alpha} z^{-1}\right)^2}$$

- which defines a second order recursive filter.

- Deriche’s paper
- Deriche’s pictures
- Check the shifting property:should a $\theta(k\text{-}m)$ be inserted?
- Need to consider also relationship of z-transform to DFT (Ifeachor page 134) and to Laplace transform (page 128)
- Also consider what above an be deleted; regard kernel factorisation and recursive filter as the objectives.