Infinite impulse response (IIR) digital filters have both a recursive and a non-recursive section:

- An IIR filter can be viewed as two FIR filters, one of which is connected in a feedback loop.
- The key design concern for an adaptive IIR filter is ensuring that the recursive section is stable.
Although IIR filters have some favourable properties, they also have some “unfavourable” properties. In general IIR filters are not linear phase and therefore introduce phase distortion. Therefore the use of IIR filter in phase sensitive applications such as data communications, and hi-fidelity audio should be carefully considered. Although by careful design IIR filters can often be made approximately linear phase in the passband.

(Non-adaptive) IIR filters are designed based on bilinear transformation techniques whereby from an analog prototype (Butterworth, Chebychev etc) in the s-domain (Laplace) a useful discrete approximation can be produced.
**IIR Digital Filter Signal Flow Graph**

\[ y(k) = a_0 x(k) + a_1 x(k-1) + a_2 x(k-2) + a_3 x(k-3) + b_1 y(k-1) + b_2 y(k-2) + b_3 y(k-3) \]

\[ = \sum_{n=0}^{3} a_n x(k-n) + \sum_{m=1}^{3} b_m y(k-m) \]
Notes:
The feedforward weights of an IIR filter are often denoted by the symbol $a_n$ and the feedback coefficients by the symbol $b_m$. However this is not universal and other authors may use exactly the reverse notation. The notation used often depends on the actual problem being addressed.

Note of course that there is **no** $b_0$ filter weight. If there was, then a delayless feedback loop would exist and the filter would not be implementable:

Note also that the filters do not require to have the same number of weights.
General IIR Filter

- The general input output equation for an IIR filter with $N$ feedforward weights and $M - 1$ feedback weights is:

$$y(k) = \sum_{n=0}^{N-1} a_n x(k - n) + \sum_{m=1}^{M-1} b_m y(k - m)$$

- In vector notation $y(k) = a^T x_k + b^T y_{k-1} = w^T u_k$,

where the weight and data vectors are respectively:

$$w^T = [a^T, b^T] = [a_0, a_1, a_2, \ldots, a_{N-1}, b_1, b_2, \ldots, b_{M-1}]$$

and

$$u_k^T = [x_k^T, y_{k-1}^T]$$

$$= [x(k), x(k-1), x(k-2), \ldots, x(k-N+1), y(k-1), y(k-2), \ldots, y(k-M+1)]$$
Notes:
The use of vector notation allows for a more compact representation, and will actually simplify the mathematics at a later stage.

Using the “transpose” notation allows a row vector to be expressed as a column vector and vice versa:

\[
\mathbf{a} = \begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix} \quad \text{then} \quad \mathbf{a}^T = \begin{bmatrix} a_0 & a_1 & a_2 \end{bmatrix}
\]

The transpose of a matrix is obtained by writing the \(n\)-th column (top to bottom) of the matrix as the \(n\)-th row (left to right). The transpose of a matrix, \(\mathbf{A}\), is denoted as \(\mathbf{A}^T\). For example, if:

\[
\mathbf{A} = \begin{bmatrix}
 a_{11} & a_{12} & a_{13} \\
 a_{21} & a_{22} & a_{23} \\
 a_{31} & a_{32} & a_{33} \\
 a_{41} & a_{42} & a_{43}
\end{bmatrix}
\]

\[
\Rightarrow \quad \mathbf{A}^T = \begin{bmatrix}
 a_{11} & a_{12} & a_{13} & a_{14} \\
 a_{21} & a_{22} & a_{23} & a_{24} \\
 a_{31} & a_{32} & a_{33} & a_{34} \\
 a_{41} & a_{42} & a_{43} & a_{44}
\end{bmatrix}
\]

Therefore if \(\mathbf{B} = \mathbf{A}^T\), then for every element of \(\mathbf{A}\) and \(\mathbf{B}\), \(a_{ij} = b_{ji}\). Note also the identity:

\[
(\mathbf{AB})^T = \mathbf{B}^T\mathbf{A}^T \quad \text{and} \quad (\mathbf{A}^T)^T = \mathbf{A}
\]

The product of \(\mathbf{A}^T\mathbf{A}\) is frequently found in DSP particularly in least squares derived algorithms.
IIR Filter Stability

- To intuitively reason why IIR filters can provide sharper cut-off for fewer weights consider a simple one weight IIR filter:

\[
y(k) = x(k) + b_1 y(k - 1)
\]

- For \( b_1 < 1 \) the impulse response can be very long in duration. E.g. if \( b_1 = -0.9 \):

- A 1 weight FIR has an impulse response of length 1 only
Therefore an IIR filter with just one weight can have an impulse response that is infinite in duration. Hence the name, infinite impulse response filter. Of course when using finite precision arithmetic (i.e. a fixed wordlength of say 16 bits) the response does eventually die away to zero when the output becomes smaller than the smallest number that can be represented.

By careful selection of a few recursive filter weights we can therefore produced IIR filters that have very long impulse responses. Most filter design packages allow up to around 10 recursive weights to be specified.
IIR Filter Instability

• Note that if \( b > 1 \) then the filter output will diverge, i.e. it is unstable. E.g. if \( b_1 = 1.1 \):

\[
y(k) = x(k) + 1.1y(k - 1)
\]

![Diagram of IIR filter with diverging output](image)
Notes:

The impulse response of this filter (i.e. applying a discrete unit impulse, $\delta(k)$, following the principles of convolution described in Section A.5) is:

$$h(k) = b^k$$

If $|b| < 1$ then the filter is converging (stable) and if $|b| > 1$ the filter impulse response is diverging (unstable). Care is therefore required to ensure that the IIR filter is stable.

For a one weight filter this stability assurance trivial and the weight just requires to be less than 1. However for more than a one weight recursive filter it is no longer such a simple calculation. For example consider the following two weight IIR filter:

![Diagram](image)

Although both weights are less than 1 in magnitude, their cumulative effect will cause the filter to be unstable. For example the output sequence for an input of a unit impulse \{1,0,0,0 .....\} will be \{1, 0.9, 1.71, 2.349 .......\} leading to the output being unbounded and hence unstable.

Therefore some careful strategies to ensure that the output is bounded are required. By representing the IIR filter in the z-domain and identifying the poles of the filter we can do exactly this.
IIR Filters in the z-Domain

- For the 4 feedforward weight, and 3 feedback weight filter in Slide 6.24, the z-domain transfer function is given as:

\[
Y(z) = a_0X(z) + a_1X(z)z^{-1} + a_2X(z)z^{-2} + a_3X(z)z^{-3} + \ldots \\
+ b_1 Y(z)z^{-1} + b_2 Y(z)z^{-2} + b_3 Y(z)z^{-3}
\]

\[
\frac{Y(z)}{X(z)} = \frac{a_0 + a_1z^{-1} + a_2z^{-2} + a_3z^{-3}}{1 - b_1z^{-1} - b_2z^{-2} - b_3z^{-3}}
\]

\[
= \frac{a_0(1 - \alpha_1z^{-1})(1 - \alpha_2z^{-1})(1 - \alpha_3z^{-1})}{(1 - \beta_1z^{-1})(1 - \beta_2z^{-1})(1 - \beta_3z^{-1})} = \frac{A(z)}{B(z)}
\]

- The roots of the numerator (i.e. \(A(z) = 0\)) give the zeroes of the filter and the roots of the denominator (i.e. \(B(z) = 0\)) give the poles of filter.

- For stability of the IIR filter the magnitude of all poles are < 1.
Alternatively we can talk about the poles being within the “unit circle” of the “z-plane”; this means the same as being of a magnitude < 1.

Recalling from Slide 6.27 that it was straightforward to calculate the stability of a simple first order recursive filter, then we can determine the overall stability by looking at the stability of each first order section.

Consider an all pole (recursive, or IIR) filter:

\[
\frac{Y(z)}{X(z)} = \frac{1}{1 - b_1 z^{-1} - b_2 z^{-2} - \ldots - b_{M-1} z^{-M+1}}
\]

\[
= \frac{1}{(1 - \beta_1 z^{-1})(1 - \beta_2 z^{-1}) \ldots (1 - \beta_{M-1} z^{-1})}
\]

\[
= \left(\frac{1}{1 - \beta_1 z^{-1}}\right)\left(\frac{1}{1 - \beta_2 z^{-1}}\right) \ldots \left(\frac{1}{1 - \beta_{M-1} z^{-1}}\right)
\]

then this can be implemented as the cascade of first order sections:

\[X(z) \rightarrow \frac{1}{1 - \beta_1 z^{-1}} \rightarrow \frac{1}{1 - \beta_2 z^{-1}} \rightarrow \ldots \rightarrow \frac{1}{1 - \beta_{M-1} z^{-1}} \rightarrow Y(z)\]

As it is easy to assess the stability of a first order section, then this is how the overall filter stability is considered, i.e. by finding the roots of the denominator polynomial, and ensuring they are all less than 1.
First Order IIR Filter Instability

- The z-domain representation of the simple first order filter is:

\[ y(k) = x(k) + \beta_1 y(k-1) \]

\[ \Rightarrow H(z) = \frac{Y(z)}{X(z)} = \frac{1}{1 - \beta_1 z^{-1}} \]

- Clearly for stability \(|\beta_1| < 1\).

Expressed in a different way, for stability the root of the polynomial \((1 - \beta_1 z^{-1}) = 0\) must be less than 1.

The root of this (first order) polynomial is \(z = \beta_1\) which is also referred to as a pole of the filter.
The terminology “pole” is used for both discrete and continuous time domain systems. In the continuous world a pole is found from the roots of the denominator of an s-domain (Laplace) polynomial.

The higher the order of the filter, the more difficult it is to factorise the polynomial to check for instability.

It is sometimes desirable to design an IIR filter that is marginally stable and will therefore oscillate when an impulse is input. Therefore a simple IIR filter may be used to produce sine waves of a certain frequency.

A two pole “marginally stable” IIR filter. For an input of an impulse the filter begins to oscillate. The frequency of oscillation is controlled by the value of $b_1$. Note that the magnitude of the poles is exactly 1 for all values of $b_1$.

For more details see A-Z, *Dual Tone Multifrequency, Tone Generation*. 

**Developed by:** www.steepestascent.com
More General IIR Stability

• An (all-pole) IIR filter can be shown as two equivalent SFGs:

\[
H(z) = \frac{1}{1 - b_1 z^{-1} - b_2 z^{-2} - b_3 z^{-3}} = \frac{1}{(1 - \beta_1 z^{-1})(1 - \beta_2 z^{-1})(1 - \beta_3 z^{-1})}
\]

• Stability requires that all \(|\beta_1|, |\beta_2|, |\beta_3| < 1\), i.e. all roots (poles) of the polynomial are less than 1.
Therefore to identify stability of a recursive filter requires that all poles of the filter are less than 1 and therefore none of the cascade of first order sections will diverge. The poles may be complex numbers, however these complex numbers will always occur in conjugate pairs given that the filter weights are real numbers.

It is common practice to plot the poles (and zeroes) of a filter on the z-plane. If all poles are within the unit magnitude circle, then the filter is stable:

Consider the poles and zeroes of a simple 2nd order all-pass filter transfer function (found by simply using the quadratic formula):

\[
H(z) = \frac{1 + 2z^{-1} + 3z^{-2}}{3 + 2z^{-1} + z^{-2}}
\]

\[
= \frac{(1 - (1 + j\sqrt{2})z^{-1})(1 - (1 - j\sqrt{2})z^{-1})}{3(1 - (1/3 + j\sqrt{2}/3)z^{-1})(1 - (1/3 - j\sqrt{2}/3)z^{-1})}
\]

and obviously \( p_1 = 1/3 - j\sqrt{2}/3 \) and \( p_2 = 1/3 + j\sqrt{2}/3 \) and \( p_1^{-1} = 1 - j\sqrt{2} \) and \( p_2^{-1} = 1 + j\sqrt{2} \). This example demonstrates that given that the poles must be inside the unit circle for a stable filter, the zeroes will always be outside of the unit circle, i.e. maximum phase.

Of course factorising a polynomial of greater than order 2 is not trivial and the assistance of a computer is useful! (of order 2 can be easily done using the quadratic formula). It is also important to recall that the non-recursive filter is unconditionally stable, regardless of the values of the zeroes.

Finally note that representing the filter as a cascade of first order sections is only for analysis. An implementation (either in ASIC hardware or software) would use the standard unfactorised form, i.e. the filter weights \( b_1 \) to \( b_{M-1} \).
Analog to Digital - Differentiation Approx. 6.31

- IIR design is based on known *analog filters*. One of the simplest forms of analog to digital transform the *backward difference operator*:

\[ s \leftarrow \frac{1}{T_s}(1 - z^{-1}) \quad \text{where} \quad T_s = \frac{1}{f_s} \]

The justification is that the (Laplace) “s” represents differentiation.

\[ Y(s) = sX(s), \text{ means that } y(t) = \frac{dx(t)}{dt}, \text{ i.e. a differentiator (inductor)}. \]

- In the discrete domain the simplest form of “differentiation” (difference) is:

\[ y(k) = \frac{1}{T_s}[x(k) - x(k - 1)] \]

i.e. \[ Y(z) = \frac{1}{T_s}(1 - z^{-1})X(z) \]

- For low frequencies (\(<< f_s\)) the approximation is “reasonably” valid.
Similarly for integration \( Y(s) = \frac{X(s)}{s} \), we get \( Y(z) = \frac{T_sX(z)}{1-z^{-1}} \), i.e. \( y(k) = T_s x(k) + z^{-1} x(k) \).

One problem with this simple transform from “s” to “z” is that it does not guarantee a stable z-domain filter.

Then given, for example, a general low pass Butterworth characteristic: \( H(s) = \frac{1}{s^2 + \sqrt{2}s + 1} \)

we could produce a digital approximation to this analog circuit by substituting: \( s \leftarrow \frac{1}{T}(1-z^{-1}) \):

\[
H(z) = H(s) \bigg|_{s = \frac{1}{T}(1-z^{-1})} = \frac{1}{T^2 (1-z^{-1})^2 + \sqrt{2} \frac{1}{T} (1-z^{-1}) + 1} = \frac{T^2}{(1 - 2z^{-1} + z^{-2}) + \sqrt{2} T (1-z^{-1}) + T^2}
\]

\[
= \frac{T^2}{z^{-2} - (\sqrt{2} + 2)z^{-1} + (1 + \sqrt{2} + T^2)}
\]

Because of the stability concern a different (but similar mapping) called the Bilinear transform is used. This \textbf{does} guarantee that a stable s-domain filter will produce a stable z-domain filter by ensuring that the left hand side of the s-plane is mapped to inside the z-plane unit circle.
The Bilinear Transform

- The backward difference operator does not guarantee that the digital filter produced is stable (i.e. has all poles within in the unit circle) - even if the original analog prototype is stable.

- The **bilinear transform** is given by:

\[
s = \frac{2}{T_s} \left[ \frac{1 - z^{-1}}{1 + z^{-1}} \right]
\]

This transform is **guaranteed to produce a stable digital filter** from a stable analog filter prototype.

- The bilinear transform will always produce a filter with both (discrete) poles and zeroes in the z-plane.

- DSP design software typically uses the bilinear transform based on known prototypes of analog and digital filters such: Butterworth, Elliptic, Chebychev etc....
Consider the simple RC circuit above where the transfer function of the system is given as:

\[
H(f) = \frac{V_{\text{out}(\omega)}}{V_{\text{in}(\omega)}} = H(f) = \frac{1}{1 + j2\pi fRC} = \frac{1}{1 + j\omega RC} \quad \text{where} \quad \omega = 2\pi f.
\]

We can present this equation as a Laplace transfer function \(H(s) = \frac{V_{\text{out}(s)}}{V_{\text{in}(s)}} = \frac{1}{1 + sRC}\) where \(s = j\omega\).

It can be shown that the simple RC filter has a transfer function equivalent to a 1 pole Butterworth filter whose 3dB cutoff frequency is \(f_c = 1/(2\pi RC)\). For notational simplicity we let \(RC = 1\) and hence \(H(s) = \frac{1}{1 + s}\) and for the digital system set \(T = 1\), i.e. \(f_s = 1\text{Hz}\).

Using the bilinear transform gives:

\[
H(z) = \frac{1}{s + 1} = \frac{1}{2\left(\frac{1-z^{-1}}{1+z^{-1}}\right)+1} = \frac{1+z^{-1}}{(1-z^{-1})+1+z^{-1}} = \frac{1+z^{-1}}{3-z^{-1}} = \frac{\frac{1}{3} + \frac{1}{3}z^{-1}}{1 - \frac{1}{3}z^{-1}}
\]

Therefore the difference equation is \(y(k) = \frac{1}{3}x(k) + \frac{1}{3}x(k-1) + \frac{1}{3}y(k-1)\)