limit

`limit(`*`num`*`, `*`low`*`, `*`high`*`)`

The limit function returns *num*, limited to the range from *low* to *high*:

num if *low* <= *num* <= *high*.

low if *num* < *low*.

high if *num* > *high*.

Since all comparisons with NaN return false, limit will not work as expected with NaNs. If a parameter may be NaN, use **numtype** to test it before calling limit.

See Also

SelectNumber, **min**, **max**

LinearFeedbackShiftRegister

`LinearFeedbackShiftRegister` `[`***`flags`***`]`

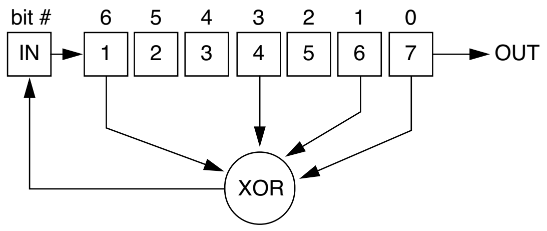
The LinearFeedbackShiftRegister operation implements a, well, linear feedback shift register, or LFSR. A LFSR is a way to produce a sequence of very bad pseudorandom numbers, or a random bit stream (that is, a random sequence of zeroes of ones that over time are nearly equal in number).

If it produces bad random numbers, why would I want to use a LFSR? A properly-configured LFSR will create a "maximal-length sequence": a LFSR of N bits will produce 2^N-1 numbers in a quasi-random sequence without repeating. That is, it will produce all the N-bit numbers except zero. This gives the sequence good spectral properties for certain applications, and, taking the least-significant bit as the output, it creates a pseudorandom bit stream with nearly equal numbers of zeroes and ones (*nearly* means one more one than zeroes).

The LinearFeedbackShiftRegister operation generates either a wave full of the sequential states of the shift register or a wave full of ones and zeroes representing the least significant bit of the shift register.

Linear Feedback Shift Registers

A LFSR is a shift register with taps. The tap bits are XOR'ed together and the result, after the register is shifted, becomes the new most significant bit. Here is a diagram of a 7-bit LFSR:



Each successive number is generated by shifting the contents of the register (boxes 1-7) to the right, while shifting in the output of the XOR node. The XOR node samples specified bits of the register contents, generating its output ready to be shifted in. Thus, the inputs of the XOR are bits sampled *before* a shift; the output of the XOR becomes the leading bit in the register *after* a shift.

In many applications the output of interest is the stream of bits that appear in the last position. This stream of bits is a pseudorandom sequence of ones and zeroes (or ones and minus ones, or whatever other binary sequence you need).

The bits fed into the XOR node are referred to as *taps*. The taps illustrated here would be specified with the tap list 7,6,4,1. As implemented in Igor Pro, the output tap (tap 7 in the illustration) is the least significant bit, so an alternate way to express the tap list is as the binary number 1001011_2 (77_{10}).

With the right taps, a LFSR produces a maximal-length sequence. The list of sequential states in a maximal-length sequence has length 2^N-1 without repeating a state. That means that every possible N-bit nonzero number appears exactly once in the maximal-length sequence.

Maximal-length tap lists always have an even number of taps.

If you have a tap list that gives a maximal-length sequence, you can generate another tap list from it. If your tap list is (n, A, B, C) the new tap list is (n, n-C, n-B, n-A). This new tap list will generate a bit stream that is the mirror image in time of the bit stream produced by the first tap list.

Flags

/DEST=*wavename*	Specifies a wave to receive the generated sequence. With /MODE=0, the number type of the wave must have at least *nbits* bits for an integer wave, or at least an *nbit* mantissa if it is a floating-point wave. That is, /N=25 requires a double-precision wave or a 32-bit integer wave. /N=18 requires any floating-point wave or a 16- or 32-bit integer wave. If you use an integer wave, we recommend an unsigned integer wave for /N=8, 16, or 32. If /MODE=1 is used, any number type is acceptable. See the Details for what happens if you don't use /DEST. If *wavename* doesn't exist, a suitable integer wave will be made. If *wavename* already exists, LinearFeedbackShiftRegister will use it as-is. The sequence length will be taken from the wave. If the number type of the wave is not suitable, an error is issued. If the sequence length is less than the number of points in your wave, it will be truncated to match.
/FREE	In a user-defined function, makes a free wave. See **Free Waves** on page IV-91 for details.
/INIT=*initialValue*	Sets the initial value of the shift register to *initialValue*. This will also be the first value in the output for /MODE=0, or the least-significant bit of *initialValue* will be the first output for /MODE=1. You can use this initial value to restart a very long sequence from the last state of a previous run. Default is a single 1 bit in the first position (bit *nbits*-1 for /N=*nbits*).
/LEN=*length*	Sets the length of sequence to generate. If the sequence repeats before *length* states are generated, the sequence is terminated early. If *length* is larger than the number of states in a maximal-length sequence, you will get a maximal-length sequence, or a shorter sequence if the initial value is seen again (that is, your sequence is not a maximal-length sequence). You can specify *length* greater than the maximal-length sequence length, but it will be truncated to the maximal length.
/MAX=*index*	An internal table of tap lists gives maximal-length sequences. This table has up to 32 tap lists for each value of *nbits*. You select a tap list by setting index to a number from 0 to 31. For values of *nbits* that do not have 32 maximal-length tap lists, the table repeats. Most *nbits* values have many more than 32 possible maximal-length sequences. For each tap list in the table, another tap list can be accessed using the /MROR flag.
/MODE=*doBitStream*	Sets the output stream format. *doBitStream*=0: Succession of bit register states (default). *doBitStream*=1: Stream of ones and zeroes.
/MROR [=*doMirror*]	Transforms the tap list into its complementary tap list, creating a mirror-image bit stream, when you use /MROR or *doMirror*=1. Specify the tap list using /TAPS, /TAPB, or /MAX.
/N=*nbits*	Determines the number of bits in the shift register. A maximal-length sequence will have 2^{nbits}-1 states. *nbits* must be in the range of 1-32. Note that *nbits* = 1 or 2 is not very interesting.
/STOP=*stopValue*	Terminates the sequence when *stopValue* is the next shift register value. You can use this flag to generate long sequences using multiple calls to LinearFeedbackShiftRegister by storing the initial value of the first call, and setting *stopValue* to that initial value in subsequent calls.
/TAPB=*tapbits*	An alternate way to express the tap list. *tapbits* is a number in which each bit represents a tap, with bit 0 representing the tap with tap number *nbits*.
/TAPS={*t1*, *t2*, …}	Specifies the tap list. Tap numbers are in the range from 1 to *nbits*.

Details

If the /TAPS, /TAPB, or /MAX flags are absent, the maximal-length sequence corresponding to /MAX=0 is generated.

In you omit the /DEST flag, a wave named W_LFSR will be generated for you. W_LFSR is an unsigned integer wave with number type set to the minimum size for the shift register size and /MODE setting. Thus, if you set `/N=10/MODE=0`, W_LFSR will be an unsigned 16-bit integer wave.

Because W_LFSR is an unsigned integer wave, you will need to redimension the wave to a floating-point wave for many purposes. Use the **Redimension** operation or the Redimension Waves item in the Data menu.

Up to /N=18, W_LFSR will initially be created large enough to hold a maximal-length sequence, unless you request a shorter sequence using the /LEN flag. If the initial value is seen again before a maximal-length sequence is generated, it means that the tap list specified was not one that generates a maximal-length sequence, and generation is terminated. The wave is shortened to the generated sequence length.

If you set a register size greater than /N=18 and you do not use /LEN, the generated sequence will stop after 2^{18}-1 (262143) states. Note that beyond some N, it will be impossible to create a wave large enough to hold a maximal-length sequence.

Some tap lists do not generate maximal-length sequences but also do not repeat the initial value. In that case, the generated sequence will be of maximal length but will contain repeated subsequences. The V_flag variable will be set to 0 if the sequence was not a maximal-length sequence, or 1 if it was. If /LEN=*length* values were generated, V_flag is set to 2.

If you specify your own wave using /DEST, the sequence length will be the same as the length of your wave. Your wave will be resized if a shorter sequence is generated.

Generating Long Sequences in Smaller Segments

Very long maximal-length sequences will not fit in the largest wave you can make. It may also be more convenient to make multiple, small fragments of a longer sequence. You can do this using the /INIT, /STOP, and /LEN flags, along with the V_nextValue variable. Here is an example of making 1000-point subsequences from a 16-bit maximal-length sequence:

```
// Start with the first 1000 states, with initial value of 1
LinearFeedbackShiftRegister/N=16/LEN=1000/INIT=1
// Restart using the V_nextValue variable to continue the sequence
// /STOP=1 sets the stopping value to the first initial value
LinearFeedbackShiftRegister/N=16/LEN=1000/INIT=(V_nextValue)/STOP=1
// Continue…
LinearFeedbackShiftRegister/N=16/LEN=1000/INIT=(V_nextValue)/STOP=1
```

Variables

The LinearFeedbackShiftRegister operation returns information in the following variables:

`V_flag`	Set to zero when a nonmaximal-length sequence was detected. This occurs if the initial value is seen again before a maximal-length sequence was generated, or if a maximal-length sequence was generated but the final state was not the same as the initial state. Set to 1 when a maximal-length sequence was generated. Set to 2 when the sequence was limited by /LEN=*length* or by the default limit of 2^{18}-1 (262143) states.
`V_tapValue`	Set to the binary representation of the tap sequence used. That is, you can generate the same sequence using `/TAPB=V_tapValue`. If you use /MROR=1, V_tapValue reflects that setting. It will also give the actual tap value used when you specify the a maximal-length sequence using the /MAX flag.
`V_nextValue`	Set to the next value beyond the last generated register state. This can be used to restart a truncated sequence.

Examples

Generate a 16-bit maximal-length sequence and reprocess the output values to be centered on zero and normalized to a maximum value of 1:

```
LinearFeedbackShiftRegister/N=16
Redimension/D W_LFSR
```