Modulo Sequences in *Mathematica*

Craig Stuart Sapp <craig@ccrma.stanford.edu>

14 January 1996

Getting Started

To use functions used in this notebook, type the pathname to the SCMTheory package in the cell below:

```
<< "SCMTheory.m"
```

To see a list of the functions defined in this package:

In[160]:=
```
? SCMTheory`*
```

```
Amplitude           IP                  Repeat
BreakStyle          LeftOptions         RightOptions
CartesianForm       Lines               RIPlot
Causal              LineStyle           SampleFunction
Continuous          LogScale            SeqPlot
Convolution         MagnitudePhasePlot  ShiftToCausal
Correlation         MagnitudePlot       ShiftToNonCausal
Decimate            NumericDFT          SignalFunction
DFTAnalyze          NumericIDFT         SignalOptions
EOPlot              OddPart             SignalPlot
EOPlotEven          OddStyle            Sinc
EOPlotOdd           PadFactor           Spectral
EOSPlot             ParabolaFit         SpectrumOptions
EvenOddPlot         ParabolaFitPlot     SpectrumPlot
EvenOddSinusoidPlot ParabolaPeak        Stretch
EvenPart            Phase               SymbolicDFT
EvenStyle           PhasePlot           SymbolicIDFT
Flip                Points              TickPeriod
HermitianQ          PointStyle          Wrap
ImaginaryPlot       PolarForm           xGreen
InnerProduct        RealImaginaryPlot   xRed
InterpGraphic       RealPlot            xxx
Interpolate         RectForm            yyy
InterpStyle         ReImPlot            ZeroPad
```

To get a brief description of a certain function:

```
?SeqPlot
```

```
SeqPlot[modSequence, options...] plots a modulo sequence as
   a discrete set of lines.  SeqPlot[function, range,
   options...] samples the function, with the possibility of
   adding the rule Continuous->True to overlay the sampled
   function.  See Options[SeqPlot] for a list of the default
   options.
```

Double-click to the right of the headers below to open/close a section of this notebook:

Definition of a Modulo Sequence

A Modulo Seuqence is a list of numbers that represents one period of an infinitely sampled sequence. The first number in the list is the 0-index value of the seuqence. The period of the sequence is equal to the length of the sequence. If the length of the modulo sequence is N, then the –1 index value (the sample before the 0-index value which is the first sample in the list) is equivalent to the element at index N-1.

Plotting Sequences

The function **SeqPlot** can be used to plot modulo sequences. There are many options that can be passed to **SeqPlot** for different plotting styles. The basic options are the same as *Mathematica*'s built-in function **Plot**. To see the additional options specific to the **SeqPlot** function, try the command **Options[SeqPlot]**.

```
Information@"SeqPlot", LongForm → FalseD
```

```
SeqPlot[modSequence, options...] plots a modulo sequence as
   a discrete set of lines.  SeqPlot[function, range,
   options...] samples the function, with the possibility of
   adding the rule Continuous->True to overlay the sampled
   function.  See Options[SeqPlot] for a list of the default
   options.
```

```
Options@SeqPlotD
```

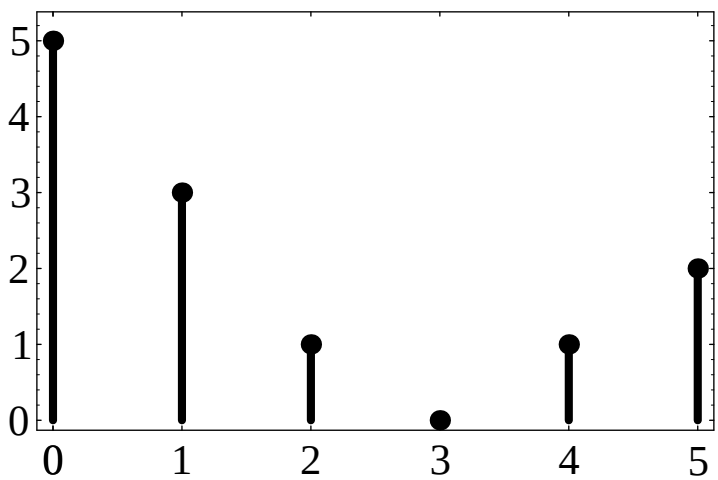
```
8Points Æ True, Lines Æ True, PointStyle Æ PointSize@0.03D,
 LineStyle Æ Thickness@0.012D, Repeat Æ 1, Causal Æ True,
 TickPeriod Æ Automatic, Continuous Æ False, PlotRange Æ All, Stretch Æ 1,
 PadFactor Æ 12, InterpStyle Æ Thickness@0.02D, Interpolate Æ False,
 InterpGraphic Æ Line<
```

Here is an concrete example of a mod sequence:

```
seq = 85, 3, 1, 0, 1, 2<;
```

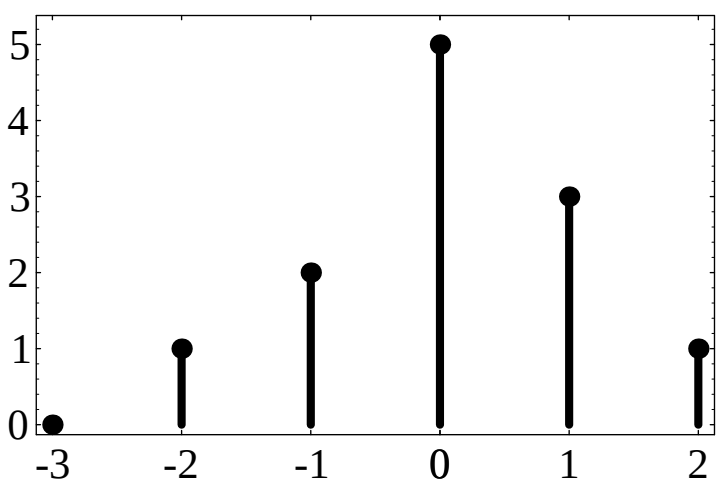
The time 0 element of the sequence is 5, and the length of the sequence is 6. Note that element 5, which has a value of 2, is also the element –1, as well as the element 11, and so on.

```
SeqPlot@seq, Frame → True, Axes → FalseD;
```



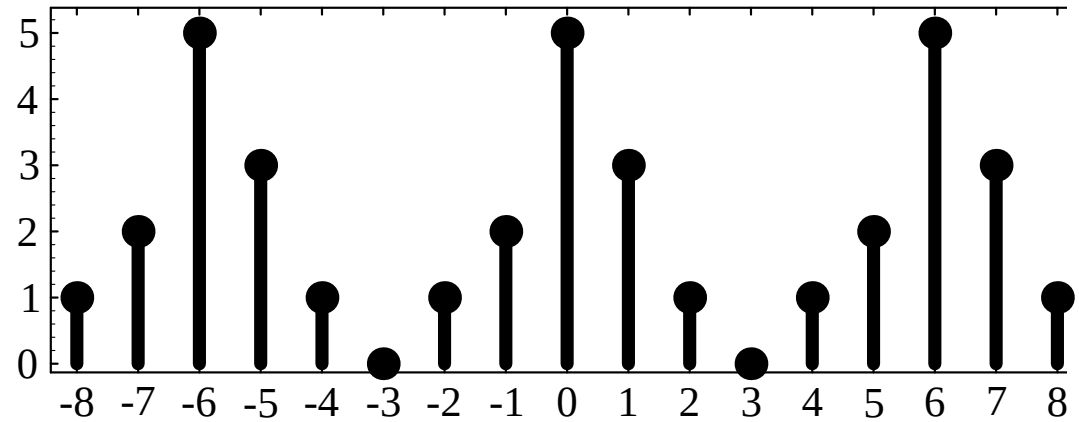
SeqPlot can also show the sequence centered about the zero element. If the length N of the sequence is even, than the extra element will be placed on the negative side instead of the positive side (as the –3 element is done in the plot below):

```
SeqPlot@seq, Causal → False, Frame → True, Axes → FalseD;
```



The plot below is an example of one of the many possible changes to the **SeqPlot** that can be done. See the Notebook `Examples/SeqPlot.ma` for an indepth demonstration of the options for **SeqPlot**.

```
SeqPlot[seq, Repeat → 3, Causal → False, Frame → True, AspectRatio → 1/3,
 Axes → False, TickFreq → 1];
```



Caution when indexing lists in *Mathematica*

Mathematica accesses lists starting with index 1. This is different from the C language which accesses arrays starting from index 0.

```
seq = {11, 22, 33, 44, 55, 66, 77, 88};
```

The first element in the seq list:

```
seq[[1]]
```

```
11
```

The 0th location of a list, store the fact that the list is a List:

```
FullForm[seq]
```

```
List[11, 22, 33, 44, 55, 66, 77, 88]
```

```
seq[[0]]
```

```
List
```

Creating Sequences

The easiest way to create a modulo sequence is to just type it in:

```
seq = 80, 1, 2, 3, 4, 5<;
```

A slightly more sophistacated way to create a sequence is to sample a continuous function:

```
SampleFunction@x, 8x, 0, 5<D
```

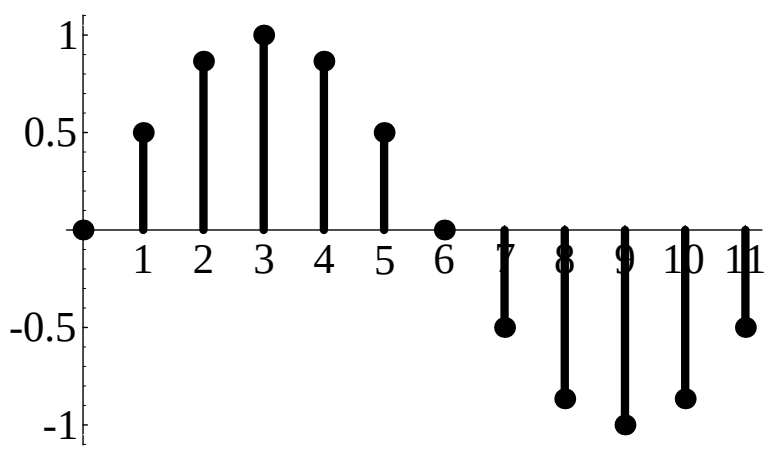
```
80, 1, 2, 3, 4, 5<
```

Here is one period of a sampled sinewave:

```
len = 12;
seq = SampleFunctionASin@2 π xD, 9x, 0, 1 - 1/len, 1/len=E
```

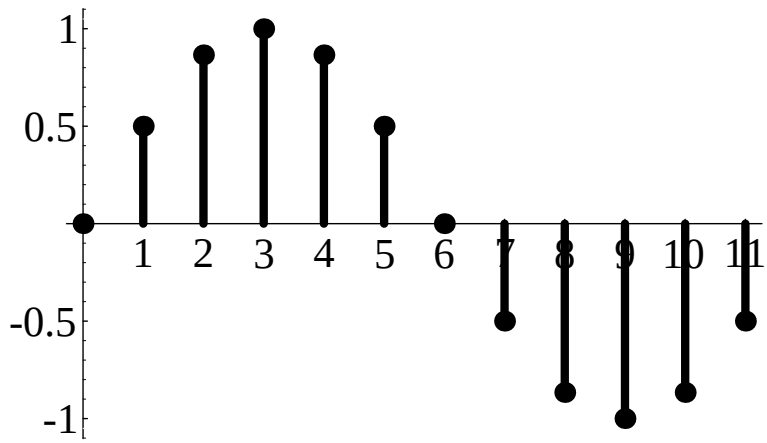
```
90, 1/2, è!!!3/2, 1, è!!!3/2, 1/2, 0, -1/2, -è!!!3/2, -1, -è!!!3/2, -1/2=
```

```
SeqPlot@seqD;
```



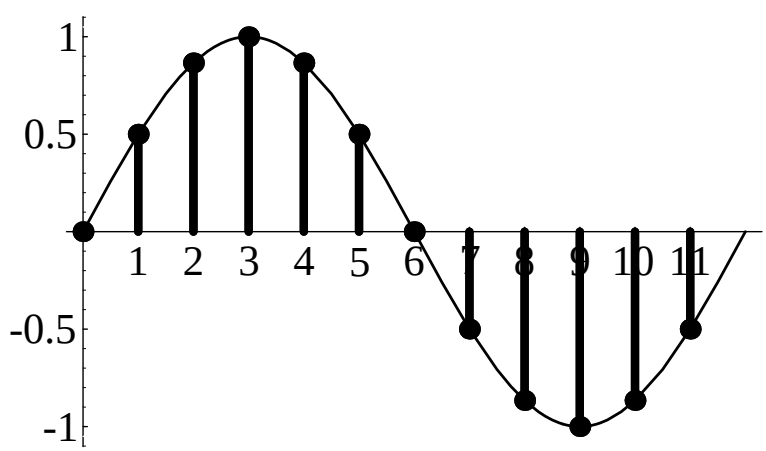
The SeqPlot function can also take a continuous function and sample it internally:

```
SeqPlot[Sin@2 π x], {x, 0, 1 - 1/len, 1/len}];
```



In the case where SeqPlot is doing the sampling of a continuous function, you can also have it plot the original continuous function:

```
SeqPlot[Sin@2 π x], {x, 0, 1 - 1/len, 1/len}, Continuous → True];
```



Perhaps you do not care for exact numbers and want to work with floating point numbers. In that case, use the build-in *Mathematica* function N[], which creates numeric approximations:

```
N@seq]
```

```
80, 0.5, 0.866025403784439, 1., 0.866025403784439, 0.5, 0, -0.5,
 -0.866025403784439, -1., -0.866025403784439, -0.5<
```

Sequences Manipulators

Flip

```
Information@"Flip", LongForm → False
```

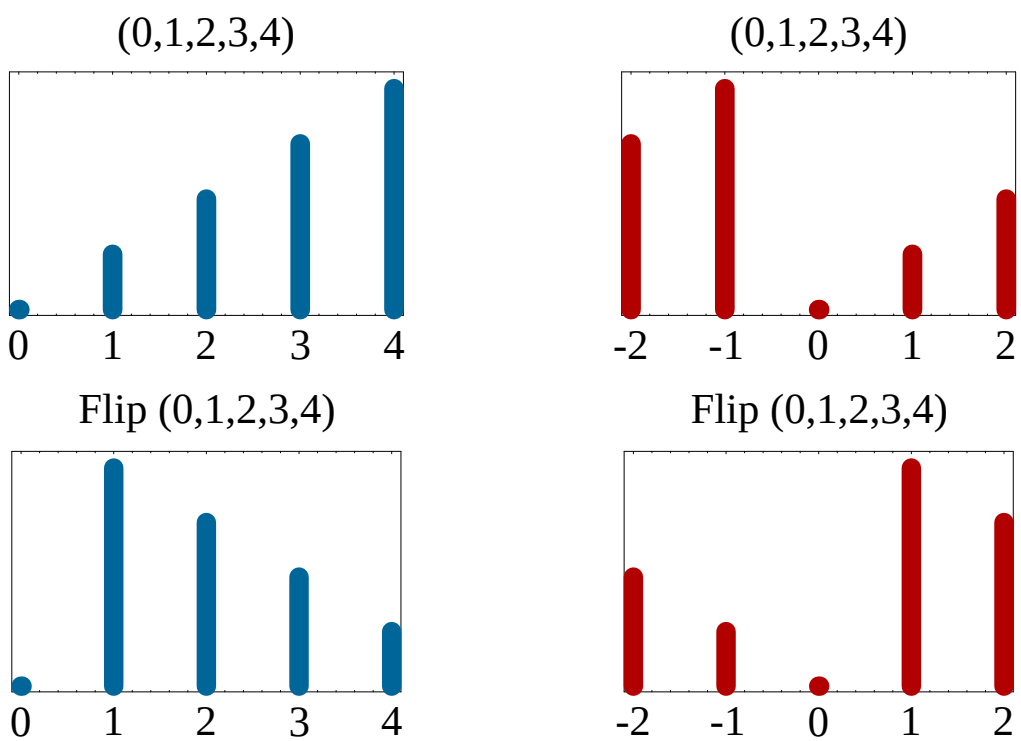
```
Flip[modSequence] Gives the flip, or reverse, of the
   modSequence. Example: Flip[{0,1,2,3,4}] = {0,4,3,2,1}
```

```
seq = 80, 1, 2, 3, 4<;
```

```
Flip@seq
```

```
80, 4, 3, 2, 1<
```

```
Block@8$DisplayFunction = Identity<, options1 = Sequence@Frame → True,
   Points → False, LineStyle → 8Thickness@0.05D, RGBColor@0, .4, .6D<,
   Axes → False, FrameTicks → 8Automatic, None<D;
 options2 = Sequence@Frame → True,
   Points → False, LineStyle → 8Thickness@0.05D, RGBColor@0.7, 0, 0D<,
   Axes → False, FrameTicks → 8Automatic, None<D;
 plot1 = SeqPlot@seq, options1, PlotLabel → "H0,1,2,3,4L"D;
 plot2 = SeqPlot@Flip@seqD, options1, PlotLabel → "Flip H0,1,2,3,4L"D;
 plot3 = SeqPlot@Flip@seqD, options2,
   Causal → False, PlotLabel → "Flip H0,1,2,3,4L"D; plot4 =
  SeqPlot@seq, options2, Causal → False, PlotLabel → "H0,1,2,3,4L"D;D;
Show@GraphicsArray@88plot1, plot4<, 8plot2, plot3<<DD;
```



Repeat

```
Information@"Repeat", LongForm → FalseD
```

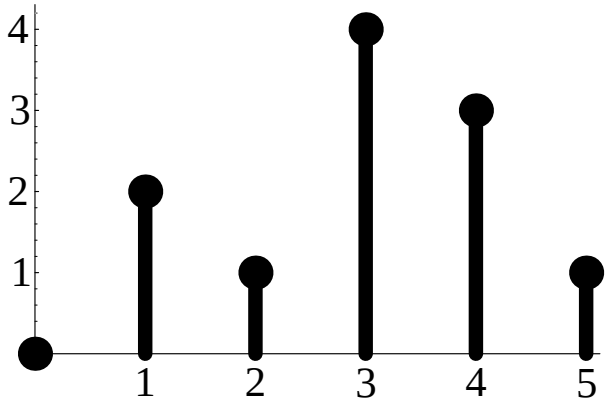
```
Repeat[modSequence, repetition] Will repeat modSequence by
   repetition.  Example: Repeat[{1,2,3}, 2] ==
   {1,2,3,1,2,3}.
```

```
seq = 80, 2, 1, 4, 3, 1<;
```

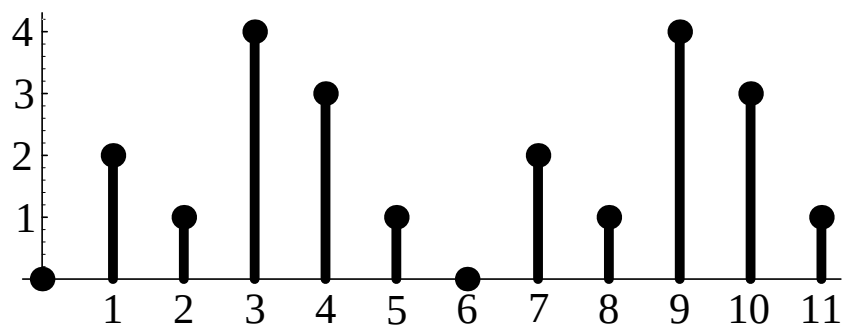
```
Repeat@seq, 2D
```

```
80, 2, 1, 4, 3, 1, 0, 2, 1, 4, 3, 1<
```

```
SeqPlot@seqD;
```




```
SeqPlotAseq, Repeat → 2, AspectRatio → 1/3 E;
```



Shift

```
Information@"Shift", LongForm → FalseD
```

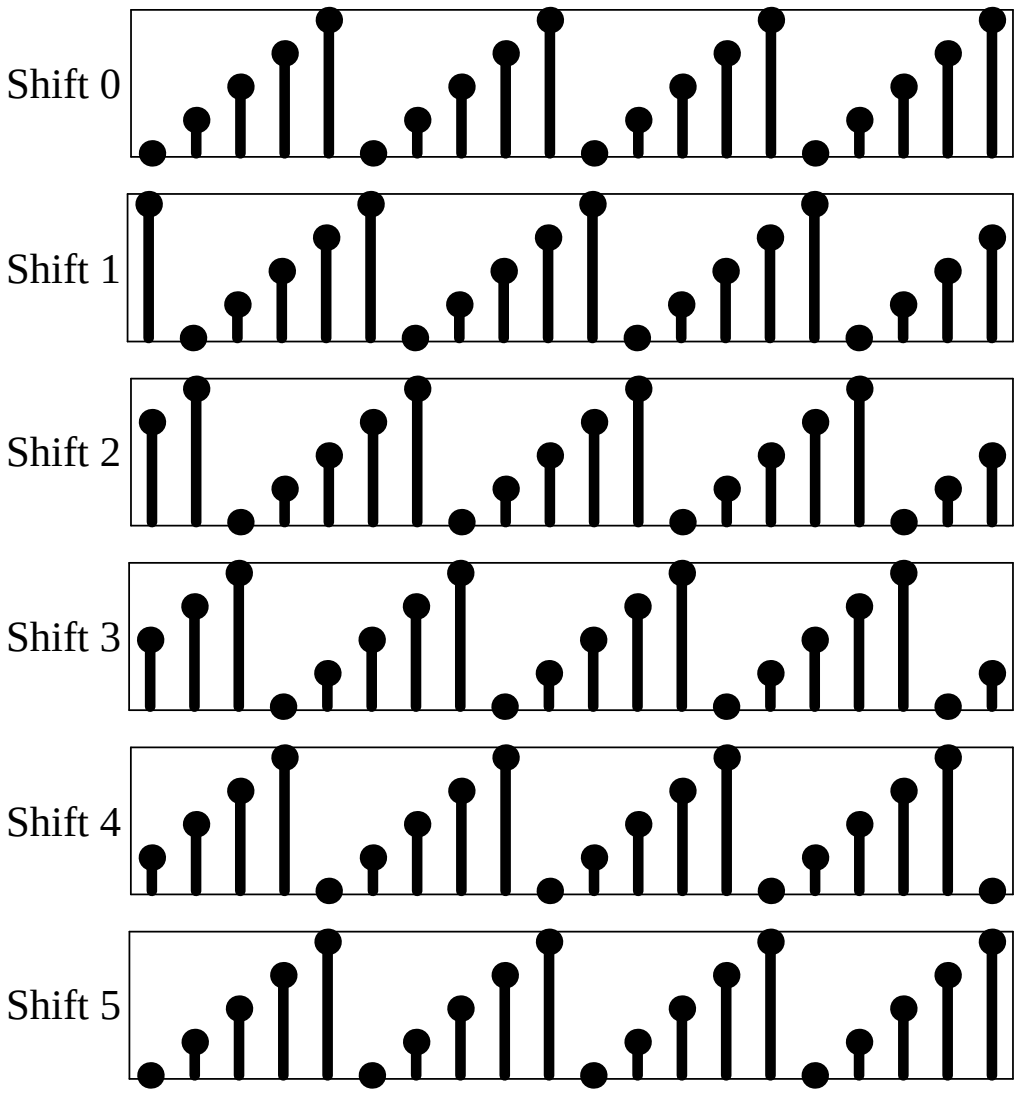
```
Shift[modSequence, amount] Same as RotateRight[].  Delays a
   sequence of samples by amount.  Example:
   Shift[{0,1,2,3}, 1] == {3,0,1,2}.
```

```
seq = 80, 1, 2, 3, 4<;
```

```
Shift@seq, 1D
```

```
84, 0, 1, 2, 3<
```

```
Block[$DisplayFunction = Identity,
 options := {AspectRatio → 1/6, Frame → True,
   FrameLabel → {"", "Shift " <> ToString[shift], "", ""},
   RotateLabel → False, FrameTicks → None};
 plots = Table[SeqPlot[Shift[seq, shift], Repeat → 4, options],
   {shift, 0, Length[seq]-1}];
Show[GraphicsArray[plots]];
```



Stretch

```
Information@"Stretch", LongForm → False
```

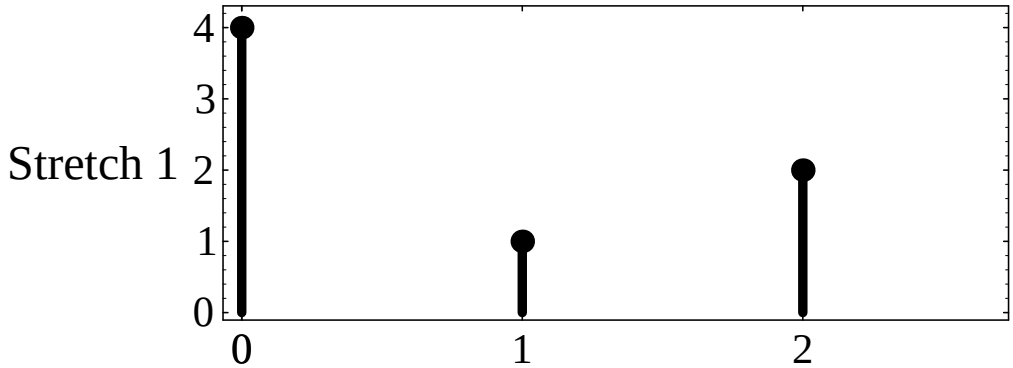
```
Stretch[signal, amount] Will stretch a signal by amount.
   Example: {1,1,1} stretched by 2 = {1,0,1,0,1,0}.
```

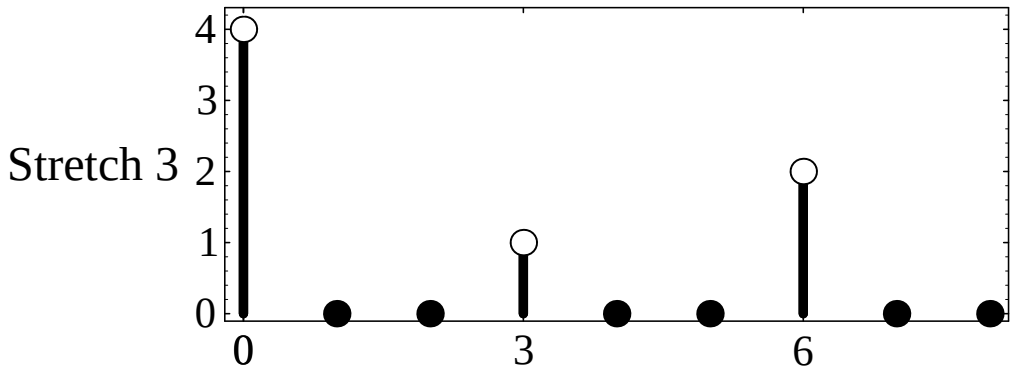
```
seq = 84, 1, 2<;
```

```
Stretch@seq, 3
```

```
84, 0, 0, 1, 0, 0, 2, 0, 0<
```

```
Block8$DisplayFunction = Identity<,
 plot1 = SeqPlot@seq, PointStyle → GrayLevel@1.0, Stretch → 3; plot2 =
  SeqPlot8Stretch@seq, 3, Stretch → 1, RotateLabel → False, Frame → True,
   FrameLabel → 8"", FontForm@"Stretch 3   ", 8"Times-Roman", 18<, "", ""<,
   AspectRatio → 1/2.5, TickPeriod → 3, PointStyle → PointSize@0.035;
 plot3 = SeqPlot8seq, PointStyle → GrayLevel@0.0,
   Stretch → 1, RotateLabel → False, Frame → True, AspectRatio → 1/2.5,
   FrameLabel → 8"", FontForm@"Stretch 1   ", 8"Times-Roman", 18<, "", ""<,
   TickPeriod → 1, PointStyle → PointSize@0.035;E;
Show8plot3, Graphics89PointSize@0.0, GrayLevel@1.0, Point89 8/3, 0.1=E=EE;
Show@plot2, plot1;
```





Decimate

```
Information@"Decimate", LongForm → False
```

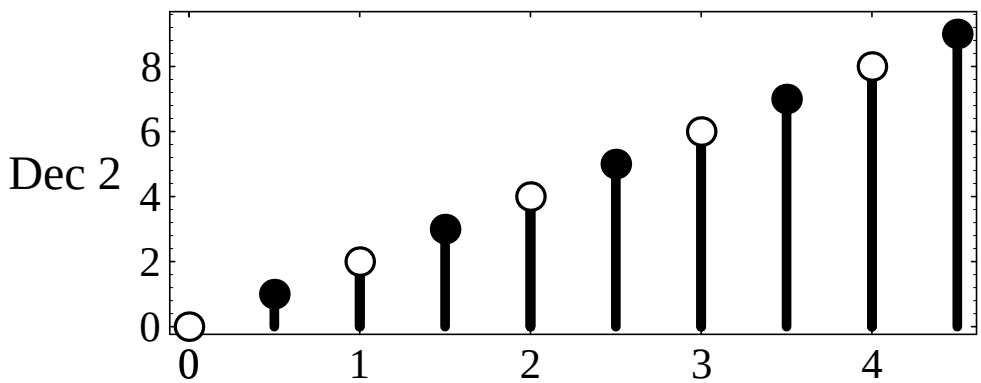
```
Decimate[modSequence, amount] will sample every amount'th
   value from the modSequence.  modSequence should be
   divisible by amount+1 or last incomplete multiple of
   amount in signal will be dropped.  Example: {0,1,2,3}
   decimated by 1 gives {0,2}.
```

```
seq = 80, 1, 2, 3, 4, 5, 6, 7, 8, 9<;
```

```
Decimate@seq, 2
```

```
80, 2, 4, 6, 8<
```

```
Block[{$DisplayFunction = Identity}, plot1 = SeqPlot[seq,
   Stretch → 1/2, PointStyle → {PointSize[0.038], GrayLevel[0.0]}];
 plot2 = SeqPlot[Decimate[seq, 2], RotateLabel → False, Frame → True,
   FrameLabel → {"", FontForm["Dec 2   ", {"Times-Roman", 18}], "", ""},
   AspectRatio → 1/2.5, TickPeriod → 1,
   PointStyle → {PointSize[0.03], GrayLevel[1.0]}];]
Show[plot2, plot1, plot2];
```



Decimate[seq, 2] leaves only the white points in the sequence above.

Interpolate

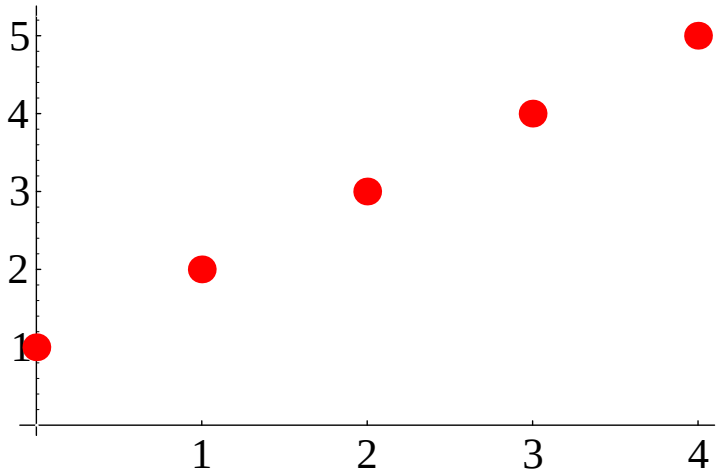
```
Information["Interpolate", LongForm → False]
```

```
Interpolate[modSequence, amount, options...] Will stretch
   the sequence with amount-1 extra samples between the
   original samples with band-limted values.
```

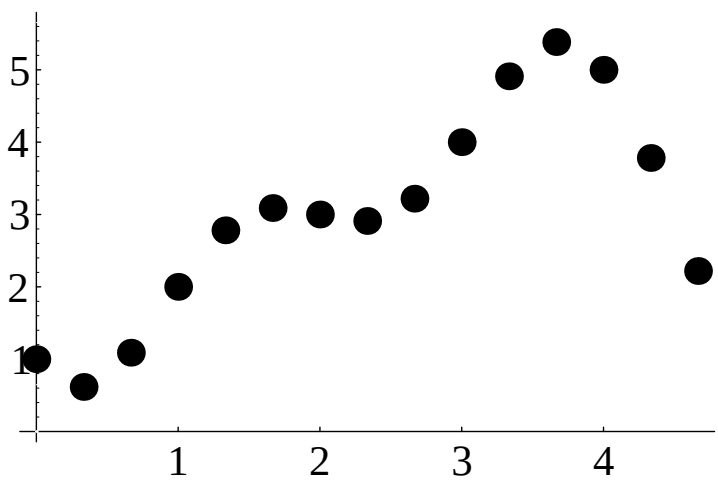
```
Interpolate[{1, 0, 0, 0, 0}, 2]
```

```
{1., 0.647213595499958, 0, -0.247213595499958, 0, 0.2, 0,
 -0.247213595499958, 0, 0.647213595499958}
```

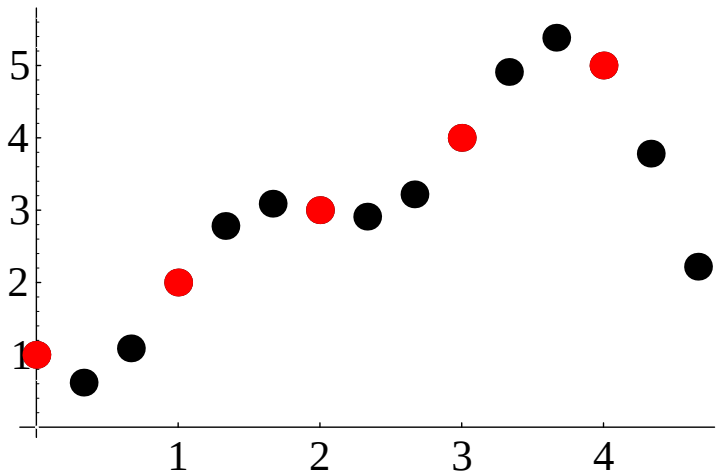
```
plot1 = SeqPlot@Interpolate@81, 2, 3, 4, 5<, 1D, Lines → False,
  PointStyle → 8RGBColor@1, 0, 0D, PointSize@0.04D<D;
```




```
plot2 = SeqPlotAInterpolate@81, 2, 3, 4, 5<, 3D, Lines → False,
  Stretch → 1/3, TickPeriod → 3, PointStyle → PointSize@0.04DE;
```

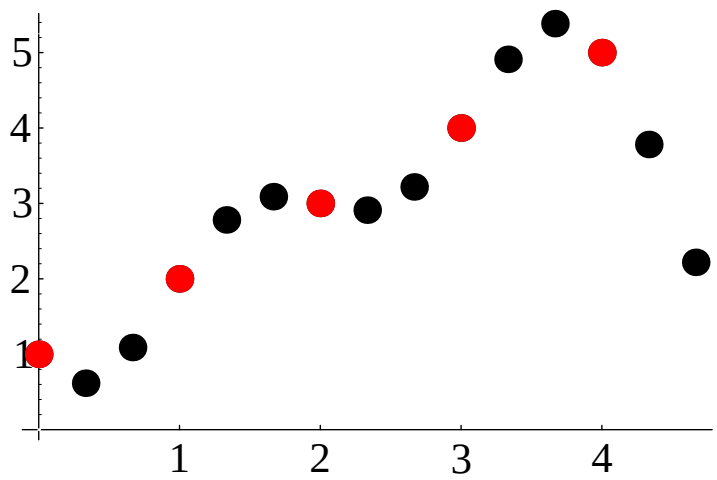


```
Show@plot2, plot1D;
```

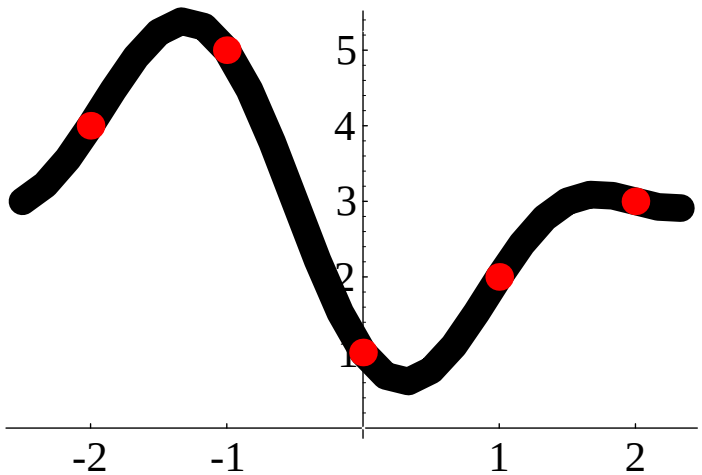


SeqPlot can also interpolate between points internally:

```
SeqPlot@81, 2, 3, 4, 5<,
 Interpolate → True, Lines → False, InterpStyle → PointSize@0.04D,
 PointStyle → 8RGBColor@1, 0, 0D, PointSize@0.04D<, InterpGraphic → Point,
 PadFactor → 3D;
```



```
SeqPlot@81, 2, 3, 4, 5<, Interpolate → True,
 Causal → False, Lines → False, InterpStyle → Thickness@0.04D,
 PointStyle → 8RGBColor@1, 0, 0D, PointSize@0.04D<, InterpGraphic → Line,
 PadFactor → 6D;
```



Alias

Not working yet

Convolution

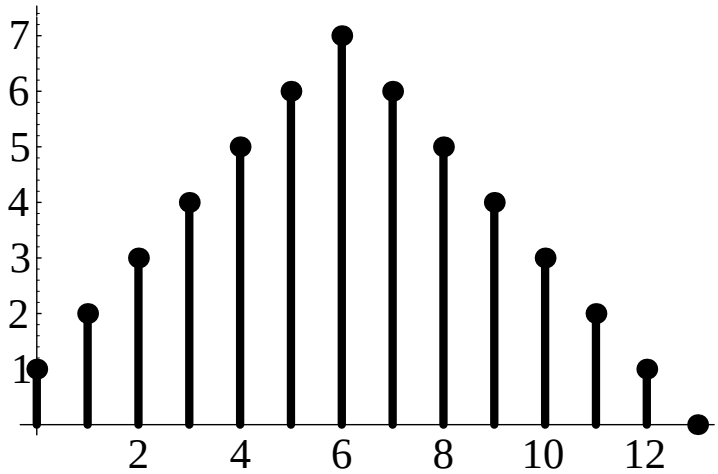
```
Information@"Convolution", LongForm → FalseD
```

```
Convolution[modSequence1, modSequence2, Spectral->True]
   Will convolve the two signals.  If Spectral->True, then
   convolution will be done by multiplying the individual
   spectral components of the two signals and then taking
   the inverse DFT. If Spectral->False then convolution
   will be done in the time domain.  Note that the
   convolution is circular -- the length of the output
   sequence is the same as the input sequences.
```

In[161]:=
```
Convolution@81, 1, 1, 1, 0, 0, 0, 0<, 81, 0, 0, 0, 0, 1, 1, 1<D
```

Out[161]=
```
84., 3., 2., 1., 0, 1., 2., 3.<
```

```
SeqPlot@Convolution@81, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0<,
  81, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0<DD;
```



Correlation

```
Correlation@81, 1, 1<, 81, 1, 1<D
```

```
83.000000000000001, 3.000000000000001, 3.000000000000001<
```

ZeroPad

```
Information@"ZeroPad", LongForm → FalseD
```

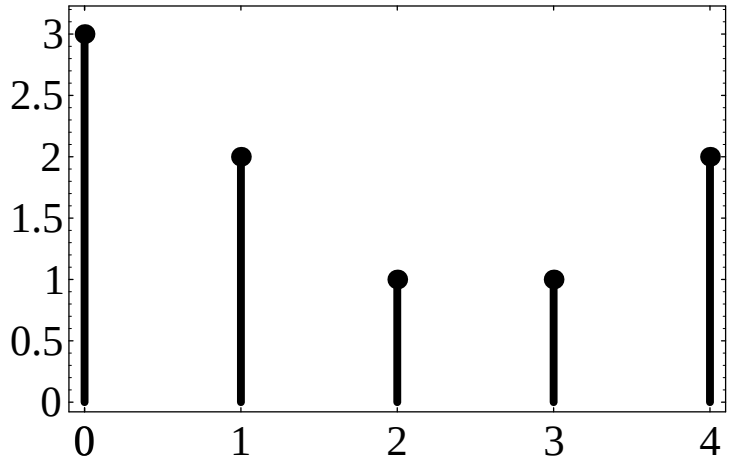
```
ZeroPad[modSequence, numZeros] will zero-pad a sequence
   with specified number of zeros. Start of signal is
   assumed to be time 0, and negative time is coming from
   the other direction, so zeros are added in the middle of
   the sequence.
```

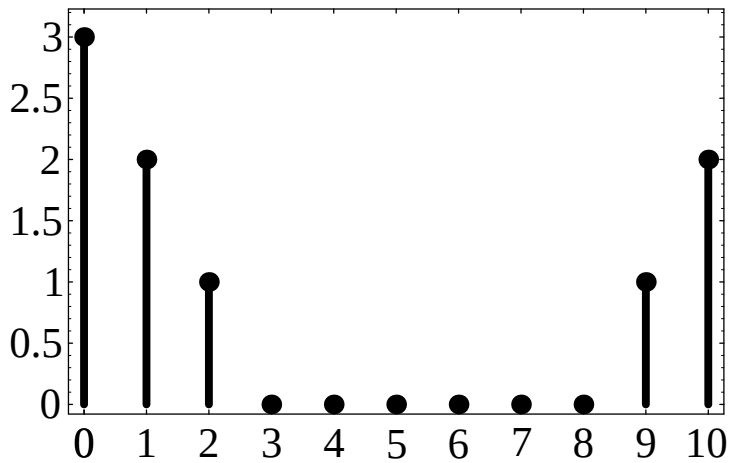
```
seq = 83, 2, 1, 1, 2<;
```

```
ZeroPad@seq, 6D
```

```
83, 2, 1, 0, 0, 0, 0, 0, 0, 1, 2<
```

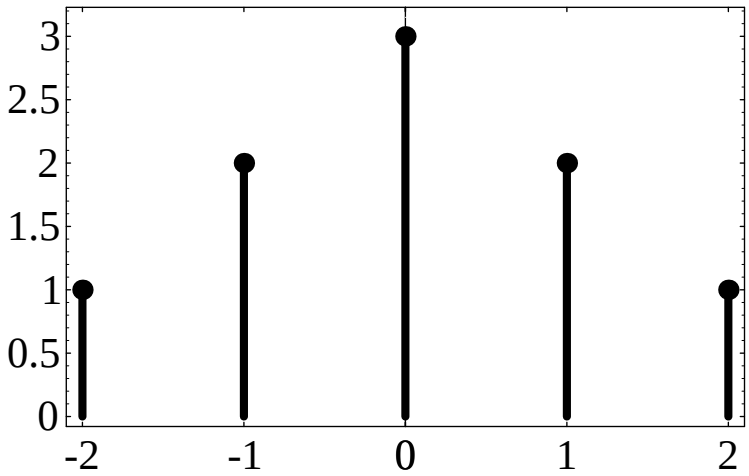
```
SeqPlot@seq, Causal → True, Frame → True];
SeqPlot@ZeroPad@seq, 6], Causal → True, Frame → True];
```

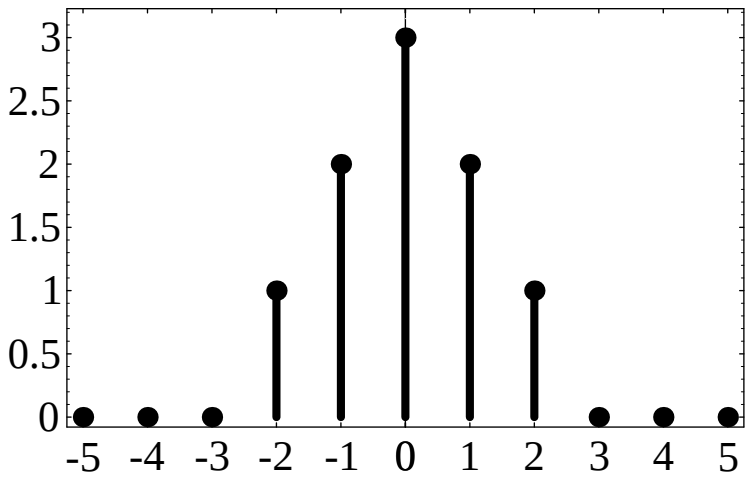




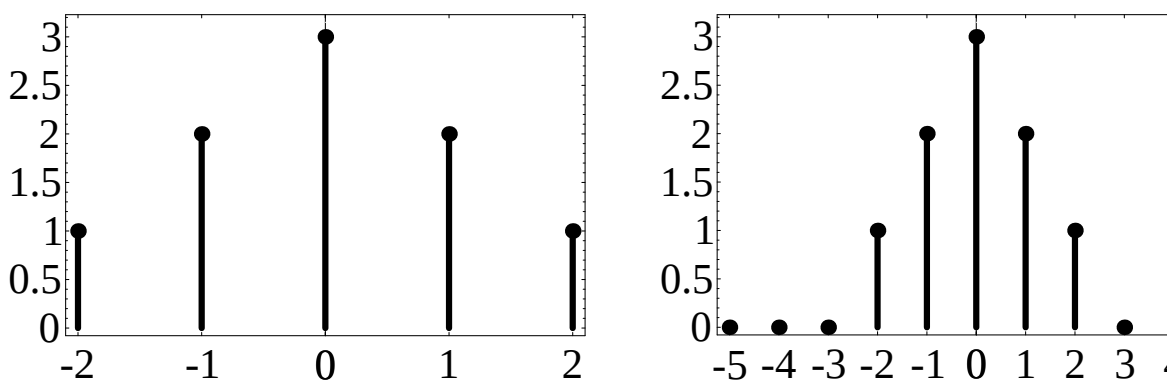
ZeroPad actually is adding zeros outside of the period of the sequence, as is more clear in the following plot.

```
SeqPlot@seq, Causal → False, Frame → True];
SeqPlot@ZeroPad@seq, 6], Causal → False, Frame → True];
```






```
Show@GraphicsArray@8%%, %<DD;
```



NumericDFT

```
? NumericDFT
```

```
NumericDFT[modSequence] Returns the numeric Discrete
   Fourier Transform of the sequence.
```

NumericIDFT

```
? NumericIDFT
```

```
NumericIDFT[modSequence] Returns the numeric Inverse DFT of
   the sequence.
```

SymbolicDFT

```
? SymbolicDFT
```

```
SymbolicDFT[modSequence, variable1:n, variable2:k] Returns
   a symbolic DFT on the sequence.
```

SymbolicIDFT

```
? SymbolicIDFT
```

```
SymbolicInverseDFT[modSequence, variable1:k, variable2:n]
   Returns a symbolic Inverse DFT of the sequence.
```