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:

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

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

```mathematica
? SCMTheory`*
```

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 Sequence 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 sequence. 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].
Here is an concrete example of a mod sequence:

```
seq = 8, 5, 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 -> False];
```

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 -> False];
```

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`. 
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:

```math
seq = 8\{0, 1, 2, 3, 4, 5\};
```

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

```math
SampleFunction[x, 8x, 0, 5\];
```

Here is one period of a sampled sinewave:

```math
len = 12;
seq = SampleFunction[\text{Sin}[2\pi x/len], 9x, 0, 1 - \frac{1}{len}, 1 - \frac{1}{len}];
```

The `SeqPlot` function can also take a continuous function and sample it internally:
In the case where SeqPlot is doing the sampling of a continuous function, you can also have it plot the original continuous function:

```
SeqPlot[A Sin@2 π x &, 9 x, 0, 1 - \(1/\text{len}\), \(1/\text{len}\) E;]
```

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@seqD
```

```
\{0, 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 = 8 0, 1, 2, 3, 4<;

Flip@seq

8 0, 4, 3, 2, 1<
\textbf{Repeat}

\begin{verbatim}
Information@"Repeat", LongForm \rightarrow False\end{verbatim}

\texttt{Repeat[modSequence, repetition]} Will repeat \texttt{modSequence} by \texttt{repetition}. Example: \texttt{Repeat\{(1,2,3), 2\} == (1,2,3,1,2,3)}.

\texttt{seq = \{0, 2, 1, 4, 3, 1\};}
Repeat@seq, 2D

\( 80, 2, 1, 4, 3, 1, 0, 2, 1, 4, 3, 1 < \)

SeqPlot@seq0;

\[ \text{SeqPlot}\{\text{seq}, \text{Repeat} \to 2, \text{AspectRatio} \to \frac{1}{3} \}; \]

\[ \text{Shift} \]

Information@"Shift", LongForm \to \text{False}0

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

\[ \text{seq} = \{80, 1, 2, 3, 4<;} \]
```
Shift@seq, 1D

84, 0, 1, 2, 3<
```
BlockA8 $DisplayFunction = \text{Identity}$,

\text{options := SequenceAAspectRatio} \rightarrow \frac{1}{6}, \text{Frame} \rightarrow \text{True}, \\
\text{FrameLabel} \rightarrow \text{Evaluate}@8 ", "Shift " <> \text{ToString}@\text{shiftD}, "", "", \text{RotateLabel} \rightarrow \text{False}, \text{FrameTicks} \rightarrow \text{None};

\text{plots} = \text{Table}@8 \text{SeqPlot}@\text{Shift}@\text{seq}, \text{shiftD}, \text{Repeat} \rightarrow 4, \text{optionsD}<, \\
\text{8shift}, 0, \text{Length}@\text{seqD}<D;E; \\
\text{Show}@\text{GraphicsArray}@\text{plotsD};
**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}`.

```plaintext
seq = 8 4, 1, 2;

Stretch@seq, 3D

8 0, 0, 1, 0, 0, 2, 0, 0
```

```
Block$DisplayFunction = Identity<,
plot1 = SeqPlot@seq, PointStyle → GrayLevel@1.0D, Stretch → 3D; plot2 =
    SeqPlot@Stretch@seq, 3D, Stretch → 1, RotateLabel → False, Frame → True,
    FrameLabel → 8""", FontForm@"Stretch 3 " , 8"Times-Roman", 18<0, "", "",<,
    AspectRatio → 1/2.5, TickPeriod → 3, PointStyle → PointSize@0.035D;
plot3 = SeqPlot@seq, PointStyle → GrayLevel@0.0D,
    Stretch → 1, RotateLabel → False, Frame → True, AspectRatio → 1/2.5,
    FrameLabel → 8""", FontForm@"Stretch 1 " , 8"Times-Roman", 18<0, "", "",<,
    TickPeriod → 1, PointStyle → PointSize@0.035D;E;
Show@plot3, Graphics@9PointSize@0.0D, GrayLevel@1.0D, Point@9 8 3 , 0.1=E=EE; Show@plot2, plot1D;
```
Decimate

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\}.

\[ \text{seq} = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}; \]

\[ \text{Decimate@seq, 2D} \]

\[ \{0, 2, 4, 6, 8\}; \]
**DisplayFunction** = Identity, plot1 = SeqPlot[seq, Stretch \( \frac{1}{2} \), PointStyle \( \to \) 8 PointSize@0.038D, GrayLevel@0.0D<];

plot2 = SeqPlot[Decimate@seq, 2D, RotateLabel \( \to \) False, Frame \( \to \) True, FrameLabel \( \to \) 8 "", FontForm@"Times-Roman", 18D, "", ""<, AspectRatio \( \to \) \( \frac{1}{2.5} \), TickPeriod \( \to \) 1, PointStyle \( \to \) 8 PointSize@0.03D, GrayLevel@1.0D<];

Show@plot2, plot1, plot2;

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

**Interpolate**

\textbf{Information@"Interpolate", LongForm \( \to \) FalseD}

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

\textbf{Interpolate@81, 0, 0, 0<, 2D}

81., 0.647213595499958, 0, -0.247213595499958, 0, 0.2, 0, -0.247213595499958, 0, 0.647213595499958
plot1 = SeqPlot[Interpolate @@ {1, 2, 3, 4, 5}, Lines -> False, PointStyle -> RGBColor[1, 0, 0], PointSize -> 0.04];

plot2 = SeqPlot[Interpolate @@ {1, 2, 3, 4, 5}, Lines -> False, Stretch -> 1/3, TickPeriod -> 3, PointStyle -> PointSize -> 0.04];
Show@plot2, plot1D;

SeqPlot can also interpolate between points internally:

SeqPlot@1, 2, 3, 4, 5<, Interpolate \rightarrow True, Lines \rightarrow False, InterpStyle \rightarrow \text{PointSize}@0.04D, PointStyle \rightarrow \text{RGBColor}@1, 0, 0D, PointSize@0.04D<, InterpGraphic \rightarrow \text{Point}, PadFactor \rightarrow 3D;
Alias

Not working yet

Convolution

*Information@"Convolution", LongForm -> False*

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, 0, 0, 0<, 80, 0, 0, 0, 1, 1<

Out[161]= 84., 3., 2., 1., 0, 1., 2., 3.<
SeqPlot@Convolution@81, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0<,
81, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0<DD;

Correlation

Correlation@81, 1, 1<, 81, 1, 1 DD

83.000000000000001, 3.000000000000001, 3.000000000000001<

ZeroPad

Information@"ZeroPad", LongForm→False DD

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, 6 DD

83, 2, 1, 0, 0, 0, 0, 0, 0, 0, 1, 2<
ZeroPad actually is adding zeros outside of the period of the sequence, as is more clear in the following plot.
Show[GraphicsArray[%, %<<DO;]
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.