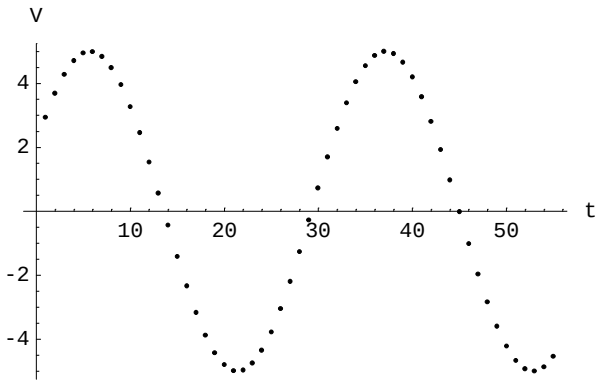
Introduction to Time Series Analysis

This is the first of a series of notes on Time Series Analysis, based on lecture notes in the course PHY308S/408S given by David Harrison. The course is based in *Mathematica* software, and some information in *Mathematica* appears; this is a very minor component of the notes.

We begin by imagining we have some process producing, say, a voltage that is varying in time:

$$V(t) = V_0 \sin(\omega t + \phi)$$

In the laboratory, we seldom if ever actually measure such a voltage. Instead we measure a succession of the values of the voltage at particular times:



Such a succession of values is a *time series*.

Almost always, we measure the values at equally spaced sampling intervals, so the values of the voltages can be written as:

$$\{V(0), V(\Delta), V(2\Delta), V(3\Delta), \ldots\}$$

where Δ is called the sampling interval.

We can also write the time series as:

$$\{V_0, V_1, V_2, V_3, \ldots\}$$

where:

$$V_n = V(n\Delta)$$

Often, we will choose units of time so that Δ is equal to one. This can in principle always be done.

At some point the apparatus was switched on and began producing the voltages. We will assume that this is at time $t = 0$. We also switched off the apparatus at some time $t = m\Delta$. The length of the time series, then, is $m + 1$. We will assume that

all voltages before we turn on the apparatus and after we turn it off are zero. We can represent this in a *Mathematica*-esque fashion as:

```
V_n = 0 /; n < 0 || n > m
```

You should be aware that although we will usually be thinking of the variable **t** to be the time, virtually nothing in what follows requires that assumption. It could just as well be position, in which case we would say that these notes are discussing *position series analysis.*

In *Mathematica*, we represent a time series as a **List**:

```
timeSeries = List[ v0, v1, v2,  ..., vm]
```

or equivalently:

```
timeSeries = {v0, v1, v2,  ..., vm}
```

All built-in commands in *Mathematica* begin with an upper-case letter. Thus if you use names for your own symbols that begin with a lower-case letter, as above, you will never collide with a built-in.

We can access parts of the series:

```
In[1] := timeSeries[[2]]
Out[1] = v1
```

In the above, **In[1]:=** is the prompt from *Mathematica*; you will not type that in. The line that begins with **Out[1]=** is what *Mathematica* returned. An equivalent command is:

```
In[2] := Part[timeSeries, 2]
Out[2] = v1
```

Note that lists in *Mathematica* begin with 1. This is similar to FORTRAN. Arrays in C and Java begin with zero, which is somewhat more natural in this context.

It is possible to trick *Mathematica* to follow the convention of C and Java, although we will not bother here. In this context, it is interesting to note that a classic book on scientific computing in the C language, **Numerical Recipes in C** by Press, Flannery, Teukolsky and Vetterling, goes to some trouble to trick C into having arrays begin with one, which they feel is more natural for the context of that book.

■ Author

 It was prepared using *Mathematica*, which produced a PostScript file; *Adobe Distiller* was then used to prepare the PDF file. This is version 1.1 of the document, date (m/d/y) 01/07/99.