An array in a standard language has a **name** (array0 in this case) and a number of **values**. We can reference a particular value using an **index**.

A wave also has a **name** (wave0 in this case) and **data values**. It differs from the array in that it has *two* indices. The first is called the **point number** and is identical to an array index or row number. The second is called the **X value** and is in the natural X units of the data (e.g., seconds, meters). Like point numbers, X values are not stored in memory but rather are *computed*.

The X value is related to the point number by the wave's X scaling, which is a property of the wave that you can set. The X scaling of a wave specifies how to compute an X value for a given point number using the formula:

```
x[p] = x0 + p*dx
```

where x[p] is the X value for point p. The two numbers x0 and dx constitute the wave's X scaling property. x0 is the starting X value. dx is the difference in X value from one point to the next. X values are uniformly spaced along the data's X dimension.

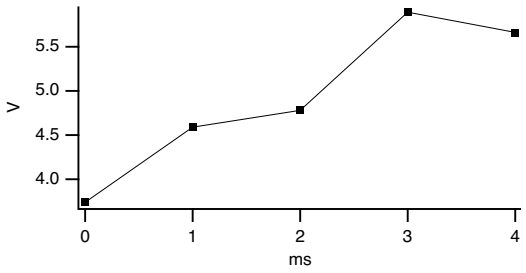
The **SetScale** operation (see page V-853) sets a wave's X scaling. You can use the Change Wave Scaling dialog to generate SetScale commands.

Why does Igor use this model for representing data? We chose this model because it provides all of the information that needed to properly display, analyze and transform waveform data.

By setting your data's X scaling, and X and data units in addition to its data values, you can make a proper graph in one step. You can execute the command

```
Display wave0
```

to produce a graph like this:



If your data is uniformly spaced on the X axis, it is *critical* that you understand and use X scaling.

The X scaling information is essential for operations such as integration, differentiation and Fourier transforms and for functions such as the **area** function (see page V-40). It also simplifies waveform assignment by allowing you to reference a single value or range of values using natural units.

Igor waves can have up to four dimensions. We call these dimensions X, Y, Z and T. X scaling extends to dimension scaling. For each dimension, there is a starting index value (x0, y0, z0, t0) and a delta index value (dx, dy, dz, dt). See Chapter II-6, **Multidimensional Waves**, for more about multidimensional waves.

XY Model of Data

If your data is not uniformly spaced along its X dimension then it can not be represented with a single wave. You need to use two waves as an XY pair.

In an XY pair, the data values of one wave provide X values and the data values of the other wave provide Y values. The X scaling of both waves is irrelevant so we leave it in its default state in which the x0 and dx components are 0 and 1. This gives us

```
x[p] = 0 + 1·p
```

This says that a given point's X value is the same as its point number. We call this "point scaling". Here is some sample data that has point scaling.

	X wave			Y wave		
	Point Number	X value ()	data value (V)	Point Number	X value ()	data value (V)
xWave / yWave	0	0	0.0	0	0	3.74
	1	1	.0013	1	1	4.59
	2	2	.0021	2	2	4.78
	3	3	.0029	3	3	5.89
	4	4	.0042	4	4	5.66

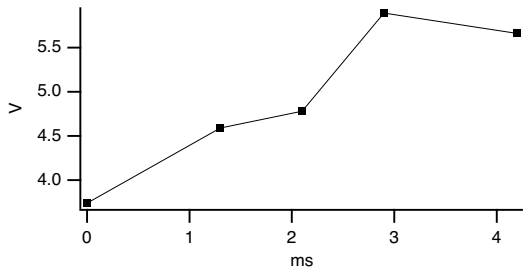
The X values serve no purpose in the XY model. Therefore, we change our thinking and look at an XY pair this way.

	X wave		Y wave	
	Point Number	Value (s)	Point Number	Value (V)
xWave / yWave	0	0.0	0	3.74
	1	.0013	1	4.59
	2	.0021	2	4.78
	3	.0029	3	5.89
	4	.0042	4	5.66

We can execute

```
Display yWave vs xWave
```

and it produces a graph like this.



Some operations, such as Fast Fourier Transforms and convolution, require equally spaced data. In these cases, it may be desirable for you to create a uniformly spaced version of your data by interpolation. See **Converting XY Data to a Waveform** on page III-109.

Some people who have uniformly spaced data still use the XY model because it is what they are accustomed to. **This is a mistake**. If your data is uniformly spaced, it will be well worth your while to learn and use the waveform model. It greatly simplifies graphing and analysis and makes it easier to write Igor procedures.