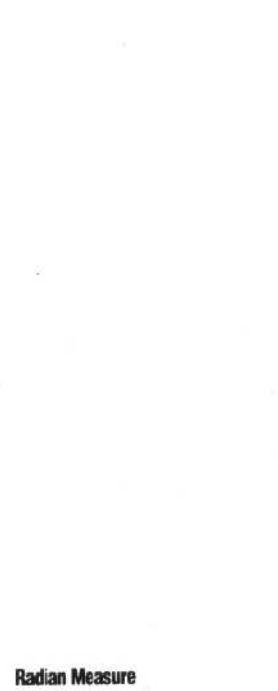
DEGREES OF PRECISION

In this second instalment of a series on mathematics and BASIC programming, we continue our look at trigonometrical functions (begun on page 154). Here we look at how the sine and cosine functions can be used in BASIC programs, and also provide ways of testing these two functions to check for any possible sources of error.

Because BASIC is provided with both COS and SIN functions, calculating the position of a point on a line after rotating it through a certain number of degrees should be an easy task. The COS of θ will give the position on the x-axis (the x co-ordinate) and the SIN of θ will give the position on the y-axis (the y co-ordinate). However, when using these two functions, it is important to remember that most versions of BASIC work in radians and not degrees. Another thing that should be checked is that the values returned for θ may not be reliable as θ approaches 0 or 1. The first thing we will do is deal with the vital difference between degrees and radians.

Radian Measure



If a portion of a circle (called an *arc*) is drawn so that its length is exactly equal to the radius of the circle, the angle at the centre is defined as one radian (see the illustration). If the radius of the circle is one unit, this portion of the circumference will also have a length of one unit. The formula for finding the circumference of a circle is $2\pi r$, so there must be 2π radians in one complete revolution. One complete revolution — the turn needed to make a full circle — expressed in a more familiar notation is 360 degrees. Therefore, 360° is equal to 2π radians. This gives us an easy way of relating degrees to radians:

$$360^\circ = 2\pi \text{ radians}$$
$$180^\circ = \pi \text{ radians}$$
$$90^\circ = \pi/2 \text{ radians}$$
$$1^\circ = \pi/180 = 0.0174 \text{ radians}$$

A BASIC program that needed to find the cosine of an angle measured in degrees would first have to convert the angle measure from degrees into radians, and then use the COS function. Try this:

```
10 INPUT "INPUT ANGLE IN DEGREES";A
20 LET B# = A * 0.0174
30 LET C# = COS(B#)
40 PRINT "THE COSINE OF ";A;" DEGREES IS ";C#
50 END
```

The hash symbols indicate that the variables in the program are double precision (which we'll look at later in this article). A simple modification of this program using the sine function, will input all values of θ from 0° to 360° and produce the sine of these values as a table. If these values are plotted against the y-axis of a graph (where the x-axis represents values of θ in radians), the sine wave graph familiar to hi-fi buffs and electrical engineers will result (see the diagram on page 155). This familiar curve is nothing more than the plot of positions of the intersection of the hypotenuse with the unit circle on the y-axis for all angles of rotation. In other words, it is an alternative way of describing a circle mathematically.

A few versions of BASIC allow the SIN and COS functions to work on either degrees or radians by using a 'software switch', but most do not. If you prefer to work in degrees all the time, it is possible to define a 'user defined function' to make the conversions for you. Here is one possibility:

```
10 REM A USER DEFINED FUNCTION FOR WORKING
   IN DEGREES
20 DEF FNDSIN (D#) = SIN(D#*0.017453293)
30 INPUT "INPUT ANGLE IN DEGREES";D#
40 PRINT "THE SINE OF#";D#;" DEGREES IS";
   FNDSIN(D#)
50 END
```

Line 20 defines a function called DSIN (standing for 'degrees/sine') that uses as its only parameter the double precision variable D#. The right hand half of the definition simply shows how the value to be returned by the function (the sine of an angle in degrees) is to be derived. To call a user defined function, you simply use the name of the function (with the value to be operated on in parenthesis) as usual. Note, however, that the line containing the definition must be executed before any calls to the function can be made.

One of the problems of using the sine function in BASIC is that not all BASICs handle it correctly as the value of θ approaches 0. It should be obvious that, as θ approaches zero, the value of SIN θ will also approach zero, since SIN θ is zero when θ is zero. In other words, as the angle gets nearer and nearer to zero, so the arc on the circumference that defines θ comes closer and closer to zero, and the point at which the hypotenuse intersects the circle gets closer and closer to 0 on the y-axis. Unfortunately, the precision of BASIC is limited. In other words, BASIC can only handle very large values up to a certain value and very small values down to a certain value. If θ is very small (say 1.0E-36, i.e. 1 × 10 to the power of minus 36), then BASIC may not be able to cope and will simply return a value of 0 for the sine of such numbers. Before using the sine function, try testing your BASIC using the following small program: