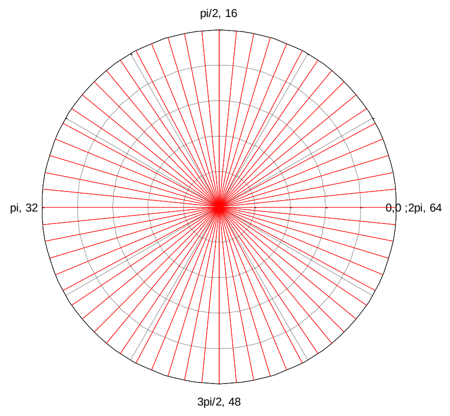
Generating a sine wave using a lookup table.

First things first. The unit circle is a good place to start when discussing sine or cosine waves. The unit circle is popular because the sine and cosine values can be directly determined from the x and y values of a unity radius circle given the angle argument. Usually the angle is in radians or sometimes degrees.

Great now what? A sine wave generator typically has an angle argument that is changed by a constant increment for each new output. The angle argument is sent to a sine wave routine to get the value that must be put out so the signal generated looks like a sine. For example if you are using radians then you would change an angle by a fixed increment until you get equal to or greater than 2π then you wrap the angle argument around by removing 2π to keep the argument between 0 and 2π. This argument is sent to the sine routine to yield the actual value you need to put out. Sometimes the argument goes from 0 to 360° if you have a sine function that uses degrees instead of radians.

Another way to get a sine value is to look the value up in a table. Given the angle you go to the table and read the sine value for that angle. When the sine function is actually a lookup table then the value of the phase argument is arbitrary because you are no longer calculating the sine value. You are just looking it up from a table. It makes sense to make a full path around the circle equal to the number of elements in the lookup table. This makes it easy to know which element of the lookup table to get the sine value from.



Look at the unit circle on the left. Notice the radian argument also has an integer argument with it. If you are calculating the sine wave value you use the 0 to 2π value as your argument as you increment around the circle. With a 64 element look-up table you will use the integer value as the argument as you increment around the circle. This makes it easy because the sine wave table elements are the sine value for the angle (0 to 2π) that the element represents.

Recap with some equations. Once around the unit circle, loop, is 1 period of a sine wave. What is the frequency of the sine wave? That is determined by the number of output samples it takes to go around the loop 1 time and how much time it takes to put out each sample. Time between samples is simply the sample period, T_S, ($1/F_S$). Time around the circle is determined by the phase increment added to the angle argument each time. Example: How many licks does it take to get to the center of a tootsie roll? I mean how many increments does it take to get around the circle (loop).

$$IncrementsAround = \frac{\text{Distance around the circle}}{\text{incremental distance}}$$

Say the phase increment is π/12 rads, and the distance around is 2π the formula gives 2π/(π/12) = 24 increments exactly to make it around the circle. How long does this take? If the sample rate is

48KHz then the sample period, TS, is 1/48000. Now $P_{sin} = Increments * P_{sin} = \frac{24}{48000} = 0.5mS$ and this frequency is $f_{sin} = \frac{F_S}{Increments} = 2KHz$. One more collection of terms.

$$f_{sin} = \frac{F_S}{Increments} = \frac{F_S}{\frac{\text{Distance around the circle}}{\text{incremental distance}}} = \frac{\text{incremental distance} * \mathrm{F}_S}{\text{Distance around the circle}}$$

We usually know the Distance around the circle, FS and the desired sine wave frequency fSIN and we want to know the incremental distance. So re-write the equation solving for incremental distance and we get:

$$\text{incremental distance} = \frac{f_{sin} * \text{Distance around the circle}}{\mathrm{F}_S}$$

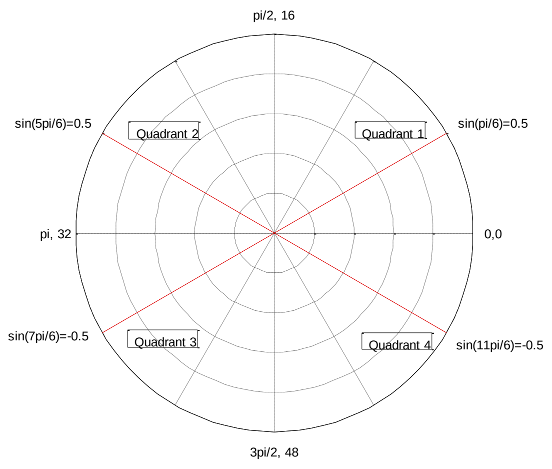
So in radians we know the incremental distance is 2K*2π/48K = π/12 which is what we started with. In this case the increment represents a phase increment. OK so maybe the equations are correct.

So what is the incremental distance if we are using a 64 element look-up table instead of a sine function based on 2π? incremental distance = 2K*64/48K = 8/3. In this case increment represents a virtual phase increment or more directly usable a table index increment.

In both cases when generating the sine wave. The distance around the circle is the term we use for wrapping around and determining the incremental distance.

Generate the table: To generate the sine look-up table you need the 0 to 2π Distance around the circle and then divide this by the number of values you want in the table. This is the incremental phase value for each value in the table. You need to start with the sine wave value for the angle starting at 0 rads. This is the first element in the table. Then add the phase increment and get the new sine wave value for each element until you get to the angle 2π which is the number of elements in the table. These should be the same number. You can have the processor generate the table as the sine function exists in the math library. You can also have MATLAB generate the values of the table. The processor generated table is more versatile because you can change the number of elements in a table on the fly. This can't be done with the MATLAB generation of the table values.

¼ size look-up table: This method of look up is used to conserve memory. It is great for a processor with limited memory. The idea is to take advantage of the symmetry of a sine wave function versus the angle of the sine. The following figure shows four angles which have the same magnitude of sine value. The angles are π/6, 5π/6, 7π/6 and 11π/6. The sine values of these 4 angles is 0.5 with a sign. We need to determine how to map the angles in quadrant 2, 3 and 4 back to an angle in quadrant 1 which yields the same sine magnitude and then set the correct sign for the quadrant it came from. The sign is easy Q3



and Q4 sine waves have the – sign. The angle mapping is a little trickier. Give it a try. Hint use the boundary values to compare and translate.