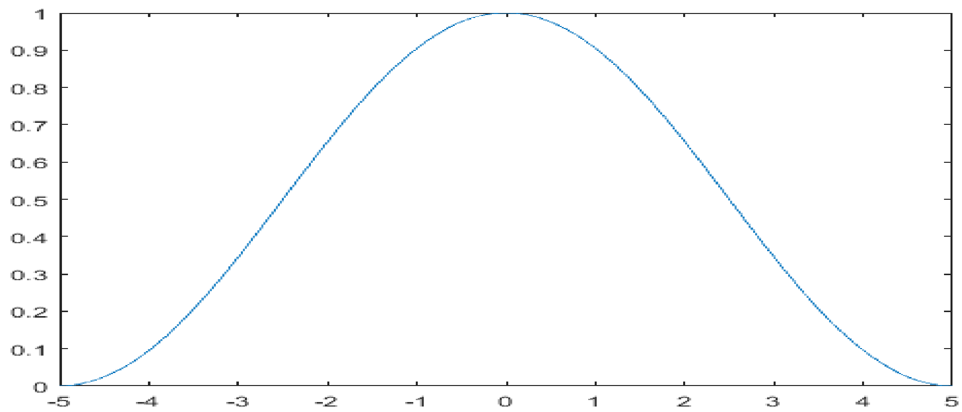

Figure 1: The Raised Cosine function is illustrated above.

Local Cosine Interpolation:

One of the problems with polynomial interpolation is their non-local nature. The Legendre and Hermite polynomials exist everywhere on $(-\infty, \infty)$ which is not ideal, since usually we want a smooth interpolation between points in a local manner. One option which has been suggested is the Raised Cosine interpolation. This starts with the simple function $\cos(x)$ on the interval $[-\pi, \pi]$. We now raise this function to $1 + cos(x)$. Furthermore we make its maximum one and we have $rc(x) = \frac{1}{2}(1 + cos(x))$ on $[-\pi, \pi]$, which can be seen in Figure 1.

The main idea of using this local interpolating function is that it is one at the center, zero at both ends, and has a zero derivative at the center and both endpoints. If we think of π and $-\pi$ as x_{k-1} and x_{k+1}, and 0 as x_k then we have a smooth function which will allow us to interpolate through the formula

$$I(x) = \sum_k f(x_k) rc_k(x),$$

where $rc_k(x)$ is the raised cosine scaled and centered at x_k.

The question now arises, "How do we implement the equation **??**". There are two answers. The first is to literally construct the raised cosine by hand, making sure each $rc_k(x)$ is centered at x_k and extends and ends at x_{k-1} and x_{k+1}. On the two ends of the approximation, you would have half of the interpolating functions, since half of them would be cut off.

The second approach is to make a matrix, with your functions embedded into the matrix. The final implementation would therefore be merely a matrix multiplication. Let's consider this option. Assume that you are interpolating on the interval $[-10, 10]$ with 5 interpolation points, $-10, -5, 0, 5, 10$. Lets assume that you decide to have interpolation points $[-10 : .01 : 10]$, which means you will have 2001 total points. For simplicity lets look at the interpolating function at the center. That function would have 1001 points, extending through the interval $[-5, 5]$. Thus we let $x_k = [-5 : .01 : 5]$ and $rc_0(x) = 1/2 * (1 + cos(x_k/5 * pi))$. This is your two sided interpolating function.

You should initialize the interpolation with the function $>> I(size([-10 : .01 : 10])) = 0$. You can now do the interpolation by hand, adding this times the function values $f(0)$ at the origin with the command

$$I(501 : 1501) = I(501 : 1501) + f(0) * rc_0(x).$$

This is easily altered to the point -5 with the function

$$I(1 : 1001) = I(1 : 1001) + f(-5) * rc_0(x).$$

Note that we merely shifted where we were putting the interpolating function. The interpolating point 5 would be handled with

$$I(1001 : 2001) = I(1001 : 2001) + f(5) * rc_0(x).$$

Now we need the endpoints properly interpolated. One the ends we will only need 1/2 of the interpolating function at each end. For the left endpoint, -10 we use the right portion of the interpolating function, and the command

$$I(1:501) = I(1:1501) + f(-10) * rc_0(501:1001).$$

The other endpoint, 10 will be included with the command

$$I(1501:2001) = I(1501:2001) + f(10) * rc_0(1:501).$$

Note that we never chance $rc_0(x)$, just its location.

This procedure can be repeated for a number of points, arbitrarily located as long as they are evenly distributed.