Fast & Accurate Cosine & Sine Approximation

First a little derivation:

$$e^{ix} = \cos(x) + i \cdot \sin(x) = c_x + i \cdot s_x, \quad e^{iy} = \cos(y) + i \cdot \sin(y) = c_y + i \cdot s_y, \quad i = \sqrt{-1}$$

$$\cos(x+y) + i \cdot sin(x+y) = e^{i(x+y)} = e^{ix} \cdot e^{iy} = (c_x + i \cdot s_x) \cdot (c_y + i \cdot s_y) = (c_x c_y - s_x s_y) + i \cdot (c_x s_y + s_x c_y)$$

$$\cos(x+y) = c_x c_y - s_x s_y = \cos(x)\cos(y) - \sin(x)\sin(y)$$

$$\sin(x+y) = c_x s_y + s_x c_y = \cos(x)\sin(y) + \sin(x)\cos(y)$$

Performing range reduction too reduce the input bounds the approximation will be over, let's say we wish to compute Cosine & Sine of an argument x, typical for computing rotation matrices. We first perform range reduction

$$x = X + k \cdot C^{(n)}, \quad where \quad C^{(n)} = \frac{2\pi}{2^n} \quad -C^{(n)}/2 < X < C^{(n)}/2 \quad k = \left\lfloor \frac{x}{C^{(n)}} + 0.5 \right\rfloor \quad X = x - k \cdot C^{(n)}$$

In our perfect mathematical world with infinitesimal precision this range reduction modulus would work to find X, however computers are limited to using finite floating point math and a problem arises when k & x are both large.

Since $C^{(n)}$ is a real non-rational number, representing it with a finite single precision FP number $C_{FP}^{(n)}$ will naturally introduce a truncation error ε due to rounding to the nearest finite FP number. The product of $C_{FP}^{(n)}$ & k for X will begin to propagate this error larger as x & consequently k get larger as seen below (assuming k and $k \cdot C_{FP}^{(n)}$ can **naïvely** be represented exactly by a FP number).

$$\varepsilon = C^{(n)} - C_{FP}^{(n)} \quad or \quad C_{FP}^{(n)} = C^{(n)} - \varepsilon; \quad k \cdot C_{FP}^{(n)} = k \cdot \left(C^{(n)} - \varepsilon\right)$$

$$X_{FP} = x - k \cdot C_{FP}^{(n)} = x - k \cdot C^{(n)} + k \cdot \varepsilon$$

To fix this error and naïve assumptions we try to represent $C^{(n)}$ with two finite FP numbers $C_{FP1}^{(n)}$ & $C_{FP2}^{(n)}$. The first value's goal is mask off or truncation off the lower precision such that the product with k does not suffer lower precision truncation which is stored in the FP mantissa portion. The second value's goal is to store the remaining finer precision such remaining after the first is computed. This effectively creates a pseudo double precision FP with only single precision FP multiplication as well as a more accurate range reduction.

$$C^{(n)} \approx C_{FP1}^{(n)} + C_{FP2}^{(n)}, \quad X_{FP} = \left(x - k \cdot C_{FP1}^{(n)}\right) - k \cdot C_{FP2}^{(n)}$$

Back to the mathematics:

$$\cos(x) = \cos\left(X + k \cdot C^{(n)}\right) = \cos(X) \cdot \cos\left(k \cdot C^{(n)}\right) - \sin(X) \cdot \sin\left(k \cdot C^{(n)}\right)$$

$$\sin(x) = \sin\left(X + k \cdot C^{(n)}\right) = \cos(X) \cdot \sin\left(k \cdot C^{(n)}\right) + \sin(X) \cdot \cos\left(k \cdot C^{(n)}\right)$$

Just a little linear algebra to see this is only a rotation matrix

$$\begin{bmatrix}\cos(x)\\ \sin(x)\end{bmatrix} = \begin{bmatrix}\cos\left(k \cdot C^{(n)}\right) & -\sin\left(k \cdot C^{(n)}\right)\\ \sin\left(k \cdot C^{(n)}\right) & \cos\left(k \cdot C^{(n)}\right)\end{bmatrix} \cdot \begin{bmatrix}\cos(X)\\ \sin(X)\end{bmatrix}$$

Now to efficient store this in a **single** lookup table i.e.:

$$c_k = \cos\left(\left(\frac{2\pi}{2^n}\right)k\right), \qquad 0 \le k \le 2^n - 1, \quad n \in \mathbb{Z} = \{\dots, -1, 0, 1, \dots\}$$

$$\sin(x) = \cos\left(x - \frac{\pi}{2}\right) = \cos\left(x - \frac{\pi}{2} + 2\pi \cdot m\right) = \cos\left(x + \frac{\pi \cdot (4m-1)}{2}\right), \qquad m \in \mathbb{Z} = \{\dots, -1, 0, 1, \dots\}$$

$$\sin(x) = \cos\left(x + \frac{3\pi}{2}\right)$$

$$\sin\left(\left(\frac{2\pi}{2^n}\right)k\right) = \cos\left(\left(\frac{2\pi}{2^n}\right) \cdot k + \frac{3\pi}{2}\right) = \cos\left(2\pi\left(\frac{k + 3 \cdot 2^{n-2}}{2^n}\right)\right), \qquad let\ m = k + 3 \cdot 2^{n-2}, \qquad n \ge 2$$

To make k & m positive integers we perform a simple modulus of 2^n which does not require division or the traditional modulus operator to compute, only a masked union of the lower n bits.

$$k_{2^n} = k - 2^n \cdot \left\lfloor \frac{k}{2^n} \right\rfloor = k[n:0] \quad \& \quad m_{2^n} = m - 2^n \cdot \left\lfloor \frac{m}{2^n} \right\rfloor = m[n:0]$$

$$c_{k[n:0]} = \cos\left(\left(\frac{2\pi}{2^n}\right)k\right), \qquad s_{m[n:0]} = \cos\left(\left(\frac{2\pi}{2^n}\right)m\right)$$

Now all that's left to apply a minimax algorithm for both $cos(X)$ & $\sin(X)$, over the interval $-C^{(n)}/2 < X < C^{(n)}/2$