**MATLAB code for the CORDIC algorithm**

**function** v = cordic(theta,n)

*% This function computes v = [cos(theta), sin(theta)] (theta in radians)*

*% using n iterations. Increasing n will increase the precision.*

*% The input angle must be inside the interval [-pi/2, pi/2]*

if theta < -pi/2 || theta > pi/2

if theta < 0

v = cordic(theta + pi, n);

else

v = cordic(theta - pi, n);

end

v = -v; *% flip the sign for second or third quadrant*

return

end

*% Initialization of tables of constants used by CORDIC*

*% need a table of arctangents of negative powers of two, in radians:*

*% angles = atan(2.^(-(0:27)));*

angles = [ *...*

0.78539816339745 0.46364760900081 0.24497866312686 0.12435499454676 *...*

0.06241880999596 0.03123983343027 0.01562372862048 0.00781234106010 *...*

0.00390623013197 0.00195312251648 0.00097656218956 0.00048828121119 *...*

0.00024414062015 0.00012207031189 0.00006103515617 0.00003051757812 *...*

0.00001525878906 0.00000762939453 0.00000381469727 0.00000190734863 *...*

0.00000095367432 0.00000047683716 0.00000023841858 0.00000011920929 *...*

0.00000005960464 0.00000002980232 0.00000001490116 0.00000000745058 ];

*% and a table of products of reciprocal lengths of vectors [1, 2^(-2j)]:*

*% Kvalues = cumulative.product(1./abs(1 + 1j\*2.^(-(0:23))))*

Kvalues = [ *...*

0.70710678118655 0.63245553203368 0.61357199107790 0.60883391251775 *...*

0.60764825625617 0.60735177014130 0.60727764409353 0.60725911229889 *...*

0.60725447933256 0.60725332108988 0.60725303152913 0.60725295913894 *...*

0.60725294104140 0.60725293651701 0.60725293538591 0.60725293510314 *...*

0.60725293503245 0.60725293501477 0.60725293501035 0.60725293500925 *...*

0.60725293500897 0.60725293500890 0.60725293500889 0.60725293500888 ];

Kn = Kvalues(min(n, length(Kvalues))); *% if n is too high, the last value is considered.*

*% Initialize loop variables:*

v = [1;0]; *% start with 2-vector cosine and sine of zero*

poweroftwo = 1;

angle = angles(1);

*% Iterations*

**for** j = 0:n-1;

if theta < 0 *% direction of rotation*

sigma = -1;

else

sigma = 1;

end

factor = sigma \* poweroftwo;

*% Matrix multiplication*

x = v[0] – sigma\*(v[1]\*2^(-j));

y = sigma\*(v[1]\*2^(-j)) + v[1];

v = [x; y];

theta = theta - sigma \* angle; *% update the remaining angle*

poweroftwo = poweroftwo / 2;

*% update the angle from table, or eventually by just dividing by two for small angles*

if j+2 > length(angles)

angle = angle / 2;

else

angle = angles(j+2);

end

end

*% Adjust length of output vector to be [cos(theta), sin(theta)]:*

v = v \* Kn;

return

end