1). CORDIC (Coordinate Rotation Digital Computer).

↳ To calculate trignometric functions, hyperbolic functions, square roots, multiplications, divisions, exponentials and logarithms with arbitrary base, typically converting one digit (or bit) per iteration.

→ CORDIC is an example of digit by digit algorithms.
→ CORDIC is used when no hardware multiplier is available in simple microcontroller and FPGA because of space and cost reasons.

Mapping Signal Processing Algorithms to architecture - NPTEL (Nitin Chandrachoodan)

Lec 99): CORDIC Algorithm.

→ CORDIC Algorithm has a lot of applications in signal processing.

→ This algorithm is very h/w friendly.

Motivation:

↳ How to compute $\cos\theta$ and $\sin\theta$ [θ given].
↳ Assume we are working with fixed point representation.

Method 1) Taylor's Series

$$\cos x = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \dots$$

$$\sin x = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \dots$$

x expressed in radians. These 2 are infinite series.

→ Here we need to compute higher powers of x
→ And we also need to compute factorial terms.

Method 2: Look Up Table

↳ Let $x = 0.25$ radians.
→ Look up table is created as follows:

x	$\cos x$
0.25	0.968
0.5	0.877
0.75	0.731
⋮	⋮

→ Given x, we can find $\cos x$ from the look up tables.

→ If we needed to find $\cos 0.3$, from the look up table we can get the values of $\cos 0.25$ and $\cos 0.5$ and using complex interpolation formula, we can get the value of $\cos 0.3$

→ But the process of using interpolation can turn out to be very computation intensive ⇒ Not worth all the effort when we want h/w friendly approach.

→ Hence we go for CORDIC Algorithm

CORDIC Algorithm [Coordinate Rotation Digital Computer]

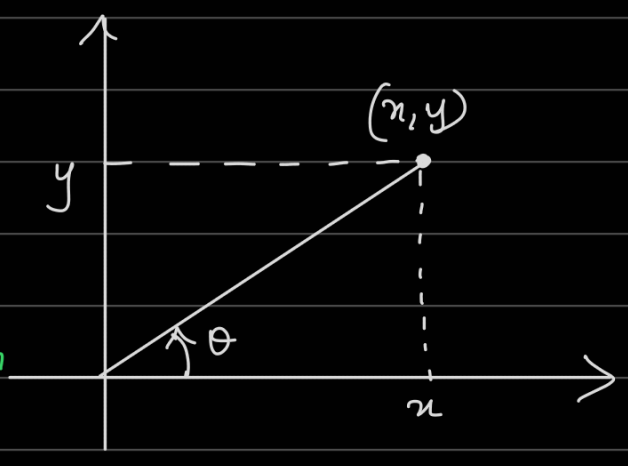
↳ Developed by Volder
↳ Replaced Analog Computation.
↳ Straightforward and Easy to Compute.

Ideas used in CORDIC Algorithm

- 1). Binary Search
- 2). Strength Reduction [Replace multiplication by bit shifting operation]

Binary Search:

- Q). Given (x, y) how do you find the angle made by vector of (x, y) with the x-axis?



→ Let $\theta = 70^\circ$ [We do not know this ahead of time]

→ First step is to rotate the vector clockwise by 45°

→ Given that the angle lies somewhere between 0° and 90°, it makes sense to rotate the vector by 45°.

if $\theta > 45^\circ \Rightarrow$ Rotated vector would still lie in Quadrant I

if $\theta < 45^\circ \Rightarrow$ Rotated vector would have moved below Quadrant I
- y-coordinated in the rotated vector would have become -ve.

How to rotate?

- I/p coordinate (x, y); Rotated Coordinates $= (x', y')$

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$

- θ: Angle of rotation.

for eg:
→ $\theta = -45^\circ \Rightarrow$ clockwise rotation of 45°.

Step1) : Rotate by -45° [45° in clockwise dir'n] and check whether $y>0$ or $y<0$?

If $y>0 \Rightarrow \theta > 45^\circ \rightarrow$ Search θ of original vector in $45^\circ - 90^\circ$ (OR)
↳ Search θ of rotated vector in $0^\circ - 45^\circ$.

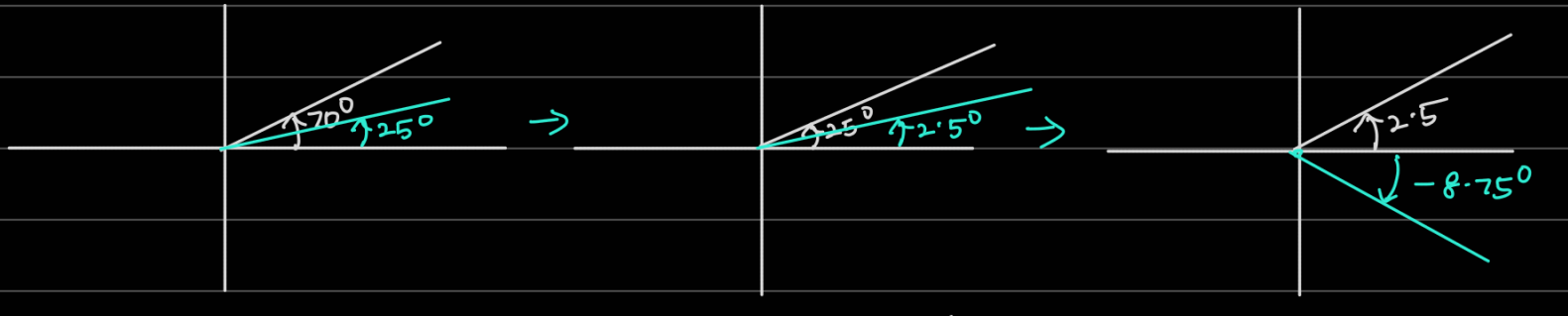
Eg). → Let $\theta = 70^\circ$. After rotation: $\theta = 70 - 45 = 25^\circ$.
→

Step2) Rotate by $-45/2 = -22.5^\circ$

$25 - 22.5 = 2.5$ (y of rotated vector would still be +ve in this case).

Step3) Rotate by $-22.5/2 = -11.25^\circ$

$2.5 - 11.25 = -8.75 \angle 0$ (y of rotated vector would be -ve)



→ Here note that we have switched quadrants
→ In the final step 'y' would have been -ve.
→ If we continued rotating in clockwise dir'n, we would go towards the third quadrant ⇒ not what we want.
→ Hence we need to change the direction of rotation.

→ Changing direction of rotation.

Step4) : Rotate by $(+)11.25/2 = 5.625^\circ$

↳ $-8.75^\circ + 5.625^\circ = -3.125 \angle 0$

(y of rotated would still be -ve ⇒ continue rotating in

counter clockwise direction)

Step5) : Rotate by $+5.625/2 = 2.8125^0$

↳ $-3.125 + 2.8125 = -0.3125^0$

↳ y of rotated vector would still be < 0

↳ Continue rotating in counter clockwise dir'n.

Step 6: Rotate by $2.8125/2 = 1.40625$

↳ $-0.3125 + 1.40625 = 1.09375$. Now y of the rotated vector would be > 0. So we can shift in clockwise dir'n

Step 7 : Rotate by $-1.40625/2 = -0.703125$

↳ $1.09375 - 0.703125 = 0.390625$

Step 8: Rotate by $-0.703125/2 = -0.3515625$

↳ $0.390625 - 0.3515625 = 0.0390625$

Step 9: Rotate by $-0.3515625/2 = -0.17578125$

↳ $0.0390625 - 0.17578125 = -0.13671875$

Step 10: Rotate by $+0.17578125/2 = 0.0878906$

↳ $-0.13671875 + 0.0878906 = -0.04882281$

→ Stopping after 10 iterations. → Choice of designer

→ Essentially, we keep doing this until our result gets as close to zero as possible.

Final Angle Estimate

- (Sum of all the angles that we used for rotation)
- $-(-45 - 22.5 - 11.25 + 5.625 + 2.8125 - 0.703125 - 0.3515625 - 0.17578125 + 0.0878906)$

$= -(-71.45) = 71.45^{\circ}$.

→ Our original angle was 70°. We did not know this beforehand. But we have ended up with 71.45° after 10 iterations, which is very close to 70°.

This method works for any angle in the first quadrant. How do we extend it to work for x and y coordinates in other quadrants?

- Look at the signs of x & y. The signs of x and y will tell us in which quadrant the value lies in.
- Switch the vector over so that the vector lies in the 1st quadrant
- Estimate the angle after switching the vector to quadrant 1.
- Apply correction so that we get the angle made by the vector in its original quadrant.

Making the rotations hardware friendly

- Rotation Vector: $\begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$
- We need to store the value of the rotation matrix for different values of θ and store them in some form of look up table. ($\theta = 45^{\circ}, 22.5^{\circ}, 11.25^{\circ}, \ldots$ => Assuming (x,y) lies in quadrant 1)

Rotation Vector = $\begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} = \cos\theta \begin{pmatrix} 1 & -\tan\theta \\ \tan\theta & 1 \end{pmatrix}$ → Need to store $\cos\theta$ & $\tan\theta$ values and store in look up table.

→ Now this where Strength Reduction comes into picture.

Strength Reduction

↳ Aim is to replace multiplication with $\tan\theta$ by a simpler operation.

Insight

↳ 1). $\tan 45^\circ = 1 = 1/2^0$

$\tan 22.5^\circ \simeq 0.5 = 1/2^1$

$\tan 11.25^\circ \simeq 0.25 = 1/2^2$

→ Based on this, we change the way in which we are going about these computations.

→ Instead of storing Rotation matrices of 45°, 22.5°, 11.25°, ..., let us find out θ values for which $\tan\theta$ is of the form $\frac{1}{2^0}, \frac{1}{2^1}, \frac{1}{2^2}, \frac{1}{2^3}, \ldots\ldots$

$\tan^{-1}(1) = 45^\circ$

$\tan^{-1}(0.5) = 26.565^\circ$

$\tan^{-1}(0.25) = 14.036^\circ$

⋮

↳ Based on this, our search algorithm will now be modified as:

↳ 1). Rotate by 45°

↳ 2). Rotate by $\pm 26.565^\circ$ (+ or − depending on y value of rotated vector)

↳ 3). Rotate by $\pm 14.036^\circ$ (+ or − depending on y value)

→ And this process will be repeated for a number of steps.
→ The number of steps is left as a choice to the designer.

Would we still get the convergence that we got in the previous case?

↳ It would converge.
↳ Because the angle at each iteration is still greater than or equal to half of the angle considered in the previous iteration.

(But how does this make the angle converge?)

Why is the modified algorithm better?

↳ Elements of the Rotation Vector will now contain only $\pm \frac{1}{2^i}$; $i = 0, 1, 2, 3 \text{ - - - -}$

↳ Multiplication by $\frac{1}{2^i}$ = Right shift i times for a fixed point representation of numbers ⇒ No hardware multiplier required in this case.

Rotation Vector: $\cos\theta \begin{bmatrix} 1 & -\tan\theta \\ \tan\theta & 1 \end{bmatrix}$ → Multiplication with this is just right shift.

This part is still remaining.

→ Multiplication by $\pm\cos\theta$ is still remaining. We will come back to this in a moment.

Iteration Steps:

1). Start with (x, y)

2). Rotate by $\theta_1 = 45^0$

↳ $\begin{pmatrix} x_1 \\ y_1 \end{pmatrix} = \cos\theta_1 \begin{pmatrix} 1 & -\tan 45^0 \\ \tan 45^0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \cos\theta_1 \begin{pmatrix} 1 & -1 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$

3). Rotate by $\theta_2 = \pm 26.565^\circ$

↳ $\begin{pmatrix} x_2 \\ y_2 \end{pmatrix} = \cos\theta_2 \begin{pmatrix} 1 & -\tan(\pm\theta_2) \\ \tan(\pm\theta_2) & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ y_1 \end{pmatrix}$

→ After 'n' iterations, we will be left with

$$\begin{pmatrix} x_n \\ y_n \end{pmatrix} = \cos\theta_n \begin{pmatrix} 1 & -\tan\pm(\theta_n) \\ \tan(\pm\theta_n) & 1 \end{pmatrix} \begin{pmatrix} x_{n-1} \\ y_{n-1} \end{pmatrix}$$

Expanding, we get

$$\begin{pmatrix} x_n \\ y_n \end{pmatrix} = (\cos\theta_1 \cos\theta_2 \dots \cos\theta_n) \begin{bmatrix} 1 & -\tan\theta_1 \\ \tan\theta_1 & 1 \end{bmatrix} \begin{bmatrix} 1 & -\tan(\pm\theta_2) \\ \tan(\pm\theta_2) & 1 \end{bmatrix} \dots \begin{bmatrix} 1 & -\tan(\pm\theta_n) \\ \tan(\pm\theta_n) & 1 \end{bmatrix}$$

↳ Once we fix the number of iterations ⇒ 'n' becomes fixed.

↳ Once we fix 'n', $[\cos\theta_1 \cdot \cos\theta_2 \cdot \cos\theta_3 \dots \cos\theta_n]$ becomes a constant value which can be pre-computed

↳ Note that $\cos\theta$ is an Even fn ⇒ $\cos\theta = \cos(-\theta)$

(So irrespective of whether θ values is +ve or -ve ⇒ "$\cos\theta_1 \cdot \cos\theta_2 \dots \cos\theta_n$" will always be a fixed number that we can pre-compute).

→ So in this manner, multiplication of cos terms is also taken care of.

→ So given (x, y), we can use this algorithm to find the angle made by the vector corresponding to (x, y) with x axis.

28/2/24 :

CORDIC Algorithm [All about Circuits]

Applications of CORDIC Algorithm:

1). Calculating sin and cosine of an arbitrary angle through rotation

→ Let us say we have an angle θ to begin with and we need to compute $\sin\theta$ and $\cos\theta$.

→ Assume that we have a hardware friendly method (Avoiding usage of multipliers) to rotate a given vector about an angle θ.

→ We specifically start with $(x_0, y_0) = (1, 0)$

Rotation:
$$\begin{pmatrix} x_1 \\ y_1 \end{pmatrix} = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} x_0 \\ y_0 \end{pmatrix}$$

Let $x_0 = 1$; $y_0 = 0$ $\Rightarrow$
$$\begin{pmatrix} x_1 \\ y_1 \end{pmatrix} = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{matrix} \cos\theta \\ \sin\theta \end{matrix}$$

$\Rightarrow$ $\boxed{x_1 = \cos\theta \quad ; \quad y_1 = \sin\theta}$ → if $(x_0, y_0) = (1, 0)$

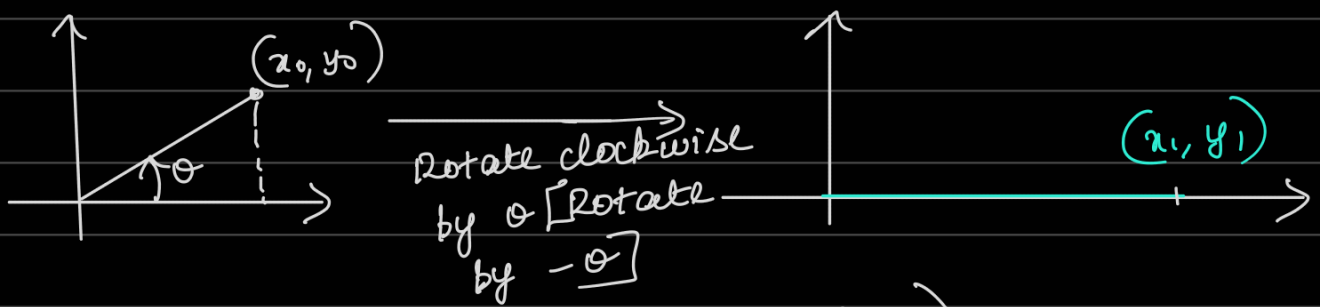
↳ Assuming we perform the rotations in a hardware friendly way (without multipliers), we end up with $x_1 = \cos\theta$; $y_1 = \sin\theta$

2). Computing Vector Magnitude through rotations:

→ Assume we want to compute $\sqrt{x^2 + y^2}$

→ Rotate the i/p vector so that it is aligned with x-axis

→ In this way $y_1 = 0$ [y of rotated component] and the x component of the rotated vector will give the magnitude of original vector.



$$\begin{pmatrix} x_1 \\ y_1 \end{pmatrix} = \begin{bmatrix} \cos(-\theta) & -\sin(-\theta) \\ \sin(-\theta) & \cos(-\theta) \end{bmatrix} \begin{pmatrix} x_0 \\ y_0 \end{pmatrix}$$

$$\begin{pmatrix} x_1 \\ y_1 \end{pmatrix} = \begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} x_0 \\ y_0 \end{pmatrix} = \begin{pmatrix} x_0\cos\theta + y_0\sin\theta \\ -x_0\sin\theta + y_0\cos\theta \end{pmatrix}$$

$$y_1 = 0 \Rightarrow y_0\cos\theta = x_0\sin\theta \Rightarrow \frac{y_0}{x_0} = \frac{\sin\theta}{\cos\theta} = \tan\theta$$

$$x_1 = x_0\cos\theta + y_0\sin\theta$$

Doubt: What to do after this to get the vector magnitude?

List of functions that can be computed through rotation:

- Arc tan, arc sin, arc cos, hyperbolic and logarithmic functions, Polar to rectangular form, Cartesian to polar form, multiplication and division.

Computing cos( ) and sin( ) of arbitrary angle through CORDIC

1). Rotating a vector by some angle θ_d once is the same as rotating the vector by several smaller angles $\theta_1, \theta_2, \theta_3 \ldots\ldots$ such that $\theta_d = \sum \theta_i$

2). For example rotating an angle by 57.535° is the same as rotating the angle by $45^\circ, 26.565^\circ, -14.036^\circ \ldots$ etc. [We can use -ve angles as well]

Example: Given $\theta = 34^\circ$, compute $\cos\theta$ and $\sin\theta$

↳ Start from $(x_0, y_0) = (1, 0)$ so that the rotated vector is of the form $\cos\theta$ and $\sin\theta$

→ Starting from $(1, 0)$ and rotating the $(1, 0)$ by an angle θ will give us a rotated vector whose x and y coordinates are $\cos\theta$ and $\sin\theta$ respectively.

i.e,

$$\begin{pmatrix} x_R \\ y_R \end{pmatrix} = \begin{pmatrix} \cos 34 & -\sin 34 \\ \sin 34 & \cos 34 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} \cos 34 \\ \sin 34 \end{pmatrix}$$

Only question that remains is how to perform this computation in a hardware friendly way [without using multipliers] ?

Target angle of rotation $= 34^\circ$

Step 1): Rotate by -45 $[\tan^{-1}(1/2^0) = 45^\circ]$:

↳ $34 - 45 = -11 < 0 \Rightarrow$ Change dir'n of rotation

Step 2): Rotate by $+26.565^\circ$ $(\tan^{-1}(1/2^1) = 26.565^\circ)$:

↳ $-11 + 26.565 = 15.565^\circ > 0 \Rightarrow$ Change dir'n of rotation.

Step 3): Rotate by -14.036° $(\tan^{-1}(1/2^2) = 14.036^\circ)$:

↳ $15.565 - 14.036 = 1.529 > 0 \Rightarrow$ Continue in the same dir'n of rotation

Step4): Rotate by -7.125° ($\tan^{-1}(1/23) = 7.125$)

→ $1.529 - 7.125 = -5.596 < 0 \Rightarrow$ Change dir'n of rotation.

Step5): Rotate by $+3.576^\circ$ ($\tan^{-1}(1/24) = 3.576^\circ$)

→ $-5.596 + 3.576 = -2.019 < 0 \Rightarrow$ Continue in same dir'n of rotation.

Step6): Rotate by $+1.7899^\circ$ ($\tan^{-1}(1/25) = 1.7899$)

→ $-2.019 + 1.7899 = -0.229^\circ < 0 \Rightarrow$ Continue rotating in same dir'n

Step7): Rotate by $+0.895^\circ$ ($\tan^{-1}(1/26) = 0.895^\circ$)

↳ $-0.229 + 0.895 = 0.666^\circ > 0 \Rightarrow$ Change dir'n of rotation.

Step8): Rotate by -0.447 ($\tan^{-1}(1/27) = 0.447^\circ$)

↳ $0.660 - 0.447 = 0.218 > 0 \Rightarrow$ Continue rotating in same dir'n.

Step9): Rotate by -0.223 ($\tan^{-1}(1/28) = 0.223^\circ$)

↳ $0.218 - 0.223 = -0.005 < 0 \Rightarrow$ Change dir'n of rotation.

Step10): Rotate by $+0.111^\circ$ ($\tan^{-1}(1/29) = 0.111$)

↳ $-0.005 + 0.111 = 0.106 > 0 \Rightarrow$ Change dir'n of rotation.

→ Aim is to get the result as close to 0 as possible

→ Stopping with 10 iterations [This is a designer's choice].

Overall Angle of Rotation

↳ -(Sum of rotation angles used in each iteration)

$= -(-45 + 26.565^\circ - 14.036 - 7.125 + 3.576 + 1.7899 + 0.895 - 0.447 - 0.223 + 0.111)$

$= -(-33.8941) = 33.8941 \Rightarrow$ Close to target angle of 34°.

How are these rotations performed in hardware:

$$\begin{pmatrix} x_R \\ y_R \end{pmatrix} = \begin{pmatrix} \cos 34 & -\sin 34 \\ \sin 34 & \cos 34 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

$$= \cos 34 \begin{pmatrix} 1 & -\tan 34 \\ \tan 34 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

↳ Decompose rotation by 34° into multiple individual rotations

$$\begin{pmatrix} x_1 \\ y_1 \end{pmatrix} = \cos 45 \begin{pmatrix} 1 & -1 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} x_2 \\ y_2 \end{pmatrix}$$

$$\begin{pmatrix} x_3 \\ y_3 \end{pmatrix} = \cos(-26.565) \begin{pmatrix} 1 & -\frac{1}{2} \\ \frac{1}{2} & 1 \end{pmatrix} \begin{pmatrix} x_2 \\ y_2 \end{pmatrix} = \begin{pmatrix} x_4 \\ y_4 \end{pmatrix}$$

$$\begin{pmatrix} x_5 \\ y_5 \end{pmatrix} = \cos(14.036) \begin{pmatrix} 1 & -\frac{1}{8} \\ \frac{1}{8} & 1 \end{pmatrix} \begin{pmatrix} x_4 \\ y_4 \end{pmatrix}$$

$$\begin{pmatrix} x_5 \\ y_5 \end{pmatrix} = \cos(14.036) \begin{pmatrix} 1 & -\frac{1}{8} \\ \frac{1}{8} & 1 \end{pmatrix} \left(\cos(-26.565) \begin{pmatrix} 1 & -\frac{1}{2} \\ \frac{1}{2} & 1 \end{pmatrix} \cos 45 \begin{pmatrix} 1 & -1 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} \right)$$

$$= \cos(14.036) \cos(-26.565) \cos(45) \begin{bmatrix} 1 & -\frac{1}{8} \\ \frac{1}{8} & 1 \end{bmatrix} \begin{bmatrix} 1 & -\frac{1}{2} \\ \frac{1}{2} & 1 \end{bmatrix} \begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$