1. Simulate over all pulses

function y = simulateRadar()

%#codegen

% Simulink-compatible version using default isotropic antenna

% === Antenna Setup ===

antenna = phased.ULA('NumElements', 8); % Use default isotropic elements

% === Waveform Setup ===

fs = 1e6; % Sample rate

prf = 1000; % PRF

pw = 1e-4; % Pulse width

waveform = phased.LinearFMWaveform('SampleRate', fs, ...

'PRF', prf, ...

'PulseWidth', pw);

nSamples = fs / prf;

% === Transmitter, Target, Platform ===

TX = phased.Transmitter('Gain', 20);

TgtModel = phased.RadarTarget;

tgtPos = [10e3 \* sqrt(3); 10e3; 0];

tgtVel = [75 \* sqrt(3); 75; 0];

PlatformModel = phased.Platform('InitialPosition', tgtPos, ...

'Velocity', tgtVel);

ChannelModel = phased.FreeSpace('TwoWayPropagation', true);

txArray = phased.Radiator('Sensor', antenna, ...

'OperatingFrequency', 300e6);

rxArray = phased.Collector('Sensor', antenna, ...

'OperatingFrequency', 300e6);

rxPreamp = phased.ReceiverPreamp('Gain', 10, ...

'NoiseFigure', 5, ...

'SampleRate', fs);

% === Simulation Settings ===

radarPos = [0;0;0];

radarVel = [0;0;0];

nPulses = 32;

% === Output Buffer ===

magResponse = zeros(nSamples \* nPulses, 1);

% === Radar Pulse Loop ===

for ii = 1:nPulses

wf = waveform(); % Generate waveform

[tgtPos, tgtVel] = PlatformModel(1/prf); % Update target motion

[~, tgtAng] = rangeangle(tgtPos, radarPos); % Find target angle

s0 = TX(wf); % Transmit

s1 = txArray(s0, tgtAng); % Radiate

s2 = ChannelModel(s1, radarPos, tgtPos, radarVel, tgtVel); % Propagate

s3 = TgtModel(s2); % Reflect

s4 = rxArray(s3, tgtAng); % Receive

s5 = rxPreamp(s4); % Amplify and noise

magResponse((ii-1)\*nSamples + (1:nSamples)) = abs(s5(:,1));

end

% === Output: Vector magnitude of first element over time ===

y = magResponse;

end

2. simulate for one pulse only

function y = simulateRadar()

%#codegen

% Simulink-compatible version that returns 1 pulse (magnitude)

% === Antenna Setup ===

antenna = phased.ULA('NumElements', 8); % Default isotropic elements

% === Waveform Setup ===

fs = 1e6; % Sample rate

prf = 1000; % PRF

pw = 1e-4; % Pulse width

waveform = phased.LinearFMWaveform('SampleRate', fs, ...

'PRF', prf, ...

'PulseWidth', pw);

nSamples = fs / prf;

% === Transmitter, Target, Platform ===

TX = phased.Transmitter('Gain', 20);

TgtModel = phased.RadarTarget;

tgtPos = [10e3 \* sqrt(3); 10e3; 0];

tgtVel = [75 \* sqrt(3); 75; 0];

PlatformModel = phased.Platform('InitialPosition', tgtPos, ...

'Velocity', tgtVel);

ChannelModel = phased.FreeSpace('TwoWayPropagation', true);

txArray = phased.Radiator('Sensor', antenna, ...

'OperatingFrequency', 300e6);

rxArray = phased.Collector('Sensor', antenna, ...

'OperatingFrequency', 300e6);

rxPreamp = phased.ReceiverPreamp('Gain', 10, ...

'NoiseFigure', 5, ...

'SampleRate', fs);

% === Simulation Setup ===

radarPos = [0;0;0];

radarVel = [0;0;0];

% === Simulate 1 pulse ===

wf = waveform(); % Generate waveform

[tgtPos, tgtVel] = PlatformModel(1/prf); % Update target

[~, tgtAng] = rangeangle(tgtPos, radarPos); % Angle to target

s0 = TX(wf); % Transmit

s1 = txArray(s0, tgtAng); % Radiate

s2 = ChannelModel(s1, radarPos, tgtPos, radarVel, tgtVel); % Propagate

s3 = TgtModel(s2); % Reflect

s4 = rxArray(s3, tgtAng); % Receive

s5 = rxPreamp(s4); % Amplify & noise

% === Return magnitude of first element over time ===

y = abs(s5(:,1)); % Size: [1000 x 1]

end

1. Simulink-compatible function: returns 1 pulse from 1 element

function y = simulateConformalRadar()

%#codegen

% Simulink-compatible function: returns 1 pulse from 1 element

% === Target Definitions ===

nTgt = 20;

tgtPos = [15e3 + abs(randn(1,nTgt))\*30e3;

abs(randn(1,nTgt))\*30e3;

zeros(1,nTgt)];

tgtVel = [abs(randn(1,nTgt))\*1e3;

abs(randn(1,nTgt))\*1e3;

zeros(1,nTgt)];

RCS = abs(randn(20,1))';

TgtModel = phased.RadarTarget('MeanRCS', RCS);

Platform = phased.Platform('InitialPosition', tgtPos, 'Velocity', tgtVel);

% === Conformal Array Setup ===

az = -10:2:10;

el = -10:2:10;

[ele\_az, ele\_el] = meshgrid(az,el);

ele\_az = ele\_az(:).';

ele\_el = ele\_el(:).';

ele\_normal = [ele\_az; ele\_el];

N\_ele = size(ele\_normal,2);

[ele\_x, ele\_y, ele\_z] = sph2cart(degtorad(ele\_az), degtorad(ele\_el), ones(1,N\_ele));

ele\_pos = [ele\_x; ele\_y; ele\_z]\*10;

antenna = phased.ConformalArray( ...

'ElementPosition', ele\_pos, ...

'ElementNormal', ele\_normal);

nElements = getNumElements(antenna);

% === Waveform Setup ===

fs = 20e6; % Sample rate = 20 MHz

prf = 1000; % Pulse Repetition Frequency

pw = 1e-4; % Pulse Width = 100 μs

waveform = phased.LinearFMWaveform('SampleRate', fs, ...

'PRF', prf, ...

'PulseWidth', pw);

nSamples = fs / prf;

% === System Blocks ===

TX = phased.Transmitter('Gain', 20);

txArray = phased.Radiator('Sensor', antenna);

rxArray = phased.Collector('Sensor', antenna);

rxPreamp = phased.ReceiverPreamp('Gain', 10, ...

'NoiseFigure', 5, ...

'SampleRate', fs);

ChannelModel = phased.FreeSpace( ...

'SampleRate', fs, ...

'TwoWayPropagation', true);

% === Radar Platform ===

radarPos = [0;0;0];

radarVel = [0;0;0];

% === Generate One Pulse Only ===

wf = waveform(); % Generate waveform

[tgtPos, tgtVel] = Platform(1/prf); % Update target positions

[~, tgtAng] = rangeangle(tgtPos, radarPos); % Target angles

s0 = TX(wf); % Transmit signal

s1 = txArray(s0, tgtAng); % Radiate from array

s2 = ChannelModel(s1, radarPos, tgtPos, radarVel, tgtVel); % Propagation

s3 = TgtModel(s2); % Target reflections

s4 = rxArray(s3, tgtAng); % Collect at Rx

s5 = rxPreamp(s4); % Add gain + noise

% === Output only magnitude of 1st element ===

y = abs(s5(:, 1)); % size = [20000 x 1]

end

1. Simulate for all pulses

function y = simulateConformalRadarCube()

%#codegen

% Simulates a radar return for all pulses and elements using a conformal array

% === Target Definitions ===

nTgt = 20;

tgtPos = [15e3 + abs(randn(1,nTgt))\*30e3;

abs(randn(1,nTgt))\*30e3;

zeros(1,nTgt)];

tgtVel = [abs(randn(1,nTgt))\*1e3;

abs(randn(1,nTgt))\*1e3;

zeros(1,nTgt)];

RCS = abs(randn(nTgt,1))';

TgtModel = phased.RadarTarget('MeanRCS', RCS);

Platform = phased.Platform('InitialPosition', tgtPos, 'Velocity', tgtVel);

% === Conformal Array Setup ===

az = -10:2:10;

el = -10:2:10;

[ele\_az, ele\_el] = meshgrid(az, el);

ele\_az = ele\_az(:).';

ele\_el = ele\_el(:).';

ele\_normal = [ele\_az; ele\_el];

N\_ele = length(ele\_az);

[ele\_x, ele\_y, ele\_z] = sph2cart(deg2rad(ele\_az), deg2rad(ele\_el), ones(1,N\_ele));

ele\_pos = [ele\_x; ele\_y; ele\_z] \* 10;

antenna = phased.ConformalArray( ...

'ElementPosition', ele\_pos, ...

'ElementNormal', ele\_normal);

nElements = getNumElements(antenna);

% === Waveform Setup ===

fs = 20e6; % Sample rate

prf = 1000; % PRF

pw = 1e-4; % Pulse width

waveform = phased.LinearFMWaveform('SampleRate', fs, ...

'PRF', prf, ...

'PulseWidth', pw);

nSamples = fs / prf;

nPulses = 32;

% === System Setup ===

TX = phased.Transmitter('Gain', 20);

txArray = phased.Radiator('Sensor', antenna);

rxArray = phased.Collector('Sensor', antenna);

rxPreamp = phased.ReceiverPreamp( ...

'Gain', 10, ...

'NoiseFigure', 5, ...

'SampleRate', fs);

ChannelModel = phased.FreeSpace( ...

'SampleRate', fs, ...

'TwoWayPropagation', true);

% === Radar Platform ===

radarPos = [0;0;0];

radarVel = [0;0;0];

% === Preallocate Output ===

dataCube = complex(zeros(nSamples, nElements, nPulses));

% === Loop Through Pulses ===

for ii = 1:nPulses

wf = waveform(); % Generate waveform

[tgtPos, tgtVel] = Platform(1/prf); % Update target position

[~, tgtAng] = rangeangle(tgtPos, radarPos); % Target angles

s0 = TX(wf);

s1 = txArray(s0, tgtAng);

s2 = ChannelModel(s1, radarPos, tgtPos, radarVel, tgtVel);

s3 = TgtModel(s2);

s4 = rxArray(s3, tgtAng);

s5 = rxPreamp(s4);

dataCube(:,:,ii) = s5; % Store full sample × element slice

end

% === Output: reshape to 2D for Simulink compatibility ===

% Flatten to [nSamples\*nPulses, nElements]

y = abs(reshape(dataCube, nSamples \* nPulses, nElements));

end

1. Simulate one pulse

function [reflected\_signal, data\_cube] = radar\_data\_cube\_generator(target\_range, target\_velocity, target\_angle, target\_rcs)

% Radar parameters

c = 3e8; % Speed of light (m/s)

fc = 10e9; % Carrier frequency (Hz)

lambda = c / fc; % Wavelength (m)

Tp = 10e-6; % Pulse width (s)

B = 50e6; % Bandwidth (Hz)

fs = B; % Sampling frequency (Hz)

PRF = 10000; % Pulse repetition frequency (Hz)

PRI = 1 / PRF; % Pulse repetition interval (s)

num\_pulses = 64; % Number of pulses

num\_elements = 4; % Number of antenna elements

d = lambda / 2; % Element spacing (m)

T\_max = PRI; % Maximum receive time (s)

num\_samples = floor(T\_max \* fs); % Number of fast-time samples per pulse

% Transmitted signal (LFM chirp)

t\_fast = (0:1/fs:Tp-1/fs)';

chirp\_slope = B / Tp;

tx\_pulse = exp(1i \* pi \* chirp\_slope \* t\_fast.^2);

% Initialize data cube

data\_cube = zeros(num\_samples, num\_elements, num\_pulses);

% Pre-calculate Doppler and angle parameters

theta\_rad = deg2rad(target\_angle);

element\_phase\_shift = exp(-1i \* 2 \* pi \* (0:num\_elements-1) \* d \* sin(theta\_rad) / lambda);

% Generate radar data cube

for m = 0:num\_pulses-1

% Calculate current range and delay

R\_m = target\_range + target\_velocity \* m \* PRI;

tau = 2 \* R\_m / c; % Round-trip delay (s)

delay\_samples = round(tau \* fs); % Delay in samples

% Skip if target is beyond receive window

if delay\_samples >= num\_samples

rx\_signal\_ref = zeros(num\_samples, 1);

else

% Create baseband received signal for reference element

rx\_signal\_ref = zeros(num\_samples, 1);

start\_idx = delay\_samples + 1;

end\_idx = min(start\_idx + length(tx\_pulse) - 1, num\_samples);

valid\_length = end\_idx - start\_idx + 1;

% Apply phase shift due to range and Doppler

phase\_term = exp(-1i \* 4 \* pi \* R\_m / lambda);

amplitude = sqrt(target\_rcs) / R\_m^2; % Simplified radar equation

rx\_signal\_ref(start\_idx:end\_idx) = amplitude \* phase\_term \* tx\_pulse(1:valid\_length);

end

% Apply array phase shifts

for n = 1:num\_elements

data\_cube(:, n, m+1) = rx\_signal\_ref \* element\_phase\_shift(n);

end

end

% Extract reflected signal (first element, first pulse)

reflected\_signal = data\_cube(:, 1, 1);

end

1. All pulses

function [reflected\_signal, data\_cube] = radar\_data\_cube\_generator1(target\_range, target\_velocity, target\_angle, target\_rcs)

% Fixed radar parameters (for deterministic output sizes)

c = 3e8; % Speed of light (m/s)

fc = 3e9; % Carrier frequency (Hz)

fs = 3e9; % Sampling frequency (Hz)

PRF = 1e5; % Pulse repetition frequency (Hz)

PRI = 1/PRF; % Pulse repetition interval (s)

num\_pulses = 6; % Number of pulses

num\_elements = 4; % Number of antenna elements

Tp = 1e-6; % Pulse width (s)

B = 300e6; % Bandwidth (Hz)

lambda = c/fc; % Wavelength (m)

% Precomputed fixed sizes (critical for Simulink)

num\_samples\_pri = 30000; % PRI \* fs = (1/1e5)\*3e9 = 30000

num\_chirp\_samples = 3000; % Tp \* fs = 1e-6 \* 3e9 = 3000

% Create transmitted signal (LFM chirp)

t\_chirp = (0:num\_chirp\_samples-1)'/fs;

chirp\_slope = B / Tp;

tx\_pulse = exp(1i \* pi \* chirp\_slope \* t\_chirp.^2);

% Array configuration

d = lambda/2; % Element spacing

element\_positions = (0:num\_elements-1) \* d;

% Initialize COMPLEX data cube with fixed size

data\_cube = complex(zeros(num\_samples\_pri, num\_elements, num\_pulses));

% Convert angle to radians

theta\_rad = deg2rad(target\_angle);

% Steering vector calculation for all elements

steering\_vector = exp(-1i \* 2 \* pi \* element\_positions \* sin(theta\_rad) / lambda);

% Simulate all pulses

for pulse\_idx = 1:num\_pulses

% Calculate current target range

current\_range = target\_range + target\_velocity \* (pulse\_idx-1) \* PRI;

% Calculate time delay (round trip)

tau = 2 \* current\_range / c;

delay\_samples = round(tau \* fs);

% Initialize COMPLEX received signal

rx\_signal = complex(zeros(num\_samples\_pri, 1));

% Create received signal if within detection window

if (delay\_samples >= 0) && (delay\_samples < num\_samples\_pri)

start\_idx = delay\_samples + 1;

end\_idx = min(start\_idx + num\_chirp\_samples - 1, num\_samples\_pri);

valid\_length = end\_idx - start\_idx + 1;

% Radar equation factors

amplitude = sqrt(target\_rcs) / current\_range^2;

phase\_shift = exp(-1i \* 4 \* pi \* current\_range / lambda);

doppler\_shift = exp(1i \* 2 \* pi \* 2 \* target\_velocity \* (pulse\_idx-1) \* PRI / lambda);

% Create received pulse segment

rx\_signal(start\_idx:end\_idx) = amplitude \* phase\_shift \* doppler\_shift \* ...

tx\_pulse(1:valid\_length);

end

% Apply array steering to all elements

for elem\_idx = 1:num\_elements

data\_cube(:, elem\_idx, pulse\_idx) = rx\_signal \* steering\_vector(elem\_idx);

end

end

% Extract reflected signal (first element, first pulse)

reflected\_signal = data\_cube(:, 1, 1);

end