Table of Contents


```
% Team 20 - Avalanche Detection
% Nov 12th, Algorithim demo
% Louis Rosenblum, Cayden Seiler, Khristian Jones
```

Initialization

```
close all
```

Sensor placement

```
s0 = [0 0];
s1 = [100 0];
s2 = [0 100];
s3 = [100 100];
```

Grid design

```
% data structure of all x,y locations for grid points
grid = cell(100,100);

for i = 1:100
    for j = 1:100
    grid{i,j} = [ (10*i-5) (10*j+995)];


    end
end
```

Distance function usage example

```
 dist1 = distance(s0,s1);
```

```
 dist1 = distance(s0, grid{30,80});
```

Avalanche condition generation

```
% Two random intergers from 1-100 for grid indexes
randx = randi(100,1,1);
randy = randi(100,1,1);

% Generate random signal to noise ratio (1 to 100, with 1 being the
 most noise)
signal_to_noise_ratio = randi(100,1,1)

origin_point = {randx,randy};
origin = grid{randx, randy};

% Temp in celsius, -40 C to 10 C
tempc = randi([-40 10],1,1)
% tempc = tempk-273

% Universal gas constant
% r = 8.314;

% Adiabatic constant
% y = 1.4;

% Molecular mass for dry air
% m = .02895;

% Speed of sound in m/s
speed_of_sound = 331.3 * sqrt(1 + (tempc / 273.15))
% speed_of_sound = sqrt(y*r*tempk/m)
```

```
signal_to_noise_ratio =

    17


tempc =

   -27


speed_of_sound =

  314.5001
```

Calculate distance to sensors

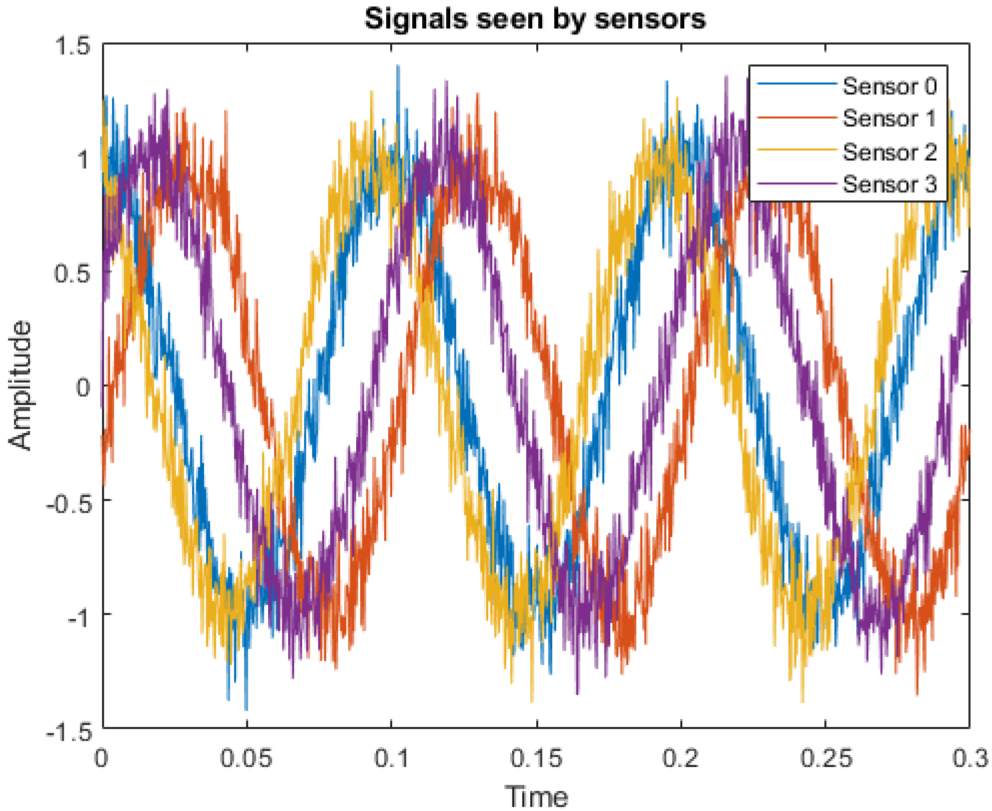
```
d0 = distance(s0,origin);
d1 = distance(s1,origin);
d2 = distance(s2,origin);
```

```
d3 = distance(s3,origin);

% Calculate difference in distance from sensors 1-3 to reference
 sensor 0
delta1 = d1 - d0;
delta2 = d2 - d0;
delta3 = d3 - d0;
```

Signal Generation

```
figure();
t = 0:1/3413:0.3;

% Generate original avalanche signal
signal0 = cos(10*2*pi.*t);

% Shift each signal to match distance travelled to each sensor
wavelength = speed_of_sound/10;
shift1 = delta1/wavelength;
shift2 = delta2/wavelength;
shift3 = delta3/wavelength;

% Generate signals received by each sensor
signal1 = cos(10*2*pi.*(t-shift1/10));
signal2 = cos(10*2*pi.*(t-shift2/10));
signal3 = cos(10*2*pi.*(t-shift3/10));

signal0_orig = signal0;
signal1_orig = signal1;
signal2_orig = signal2;
signal3_orig = signal3;

% Add gaussian noise
signal0 = awgn(signal0,signal_to_noise_ratio);
signal1 = awgn(signal1,signal_to_noise_ratio);
signal2 = awgn(signal2,signal_to_noise_ratio);
signal3 = awgn(signal3,signal_to_noise_ratio);

% Plot signals received by sensors
plot(t,signal0), hold on
plot(t,signal1);
plot(t,signal2);
plot(t,signal3);
legend('Sensor 0', 'Sensor 1', 'Sensor 2', 'Sensor 3');
title("Signals seen by sensors");
xlabel("Time");
ylabel("Amplitude"); hold off;

amplitude = max(signal0(:));

noise0 = signal0 - signal0_orig;
noise1 = signal1 - signal1_orig;
noise2 = signal2 - signal2_orig;
```

```
noise3 = signal3 - signal3_orig;
```



Noise analysis

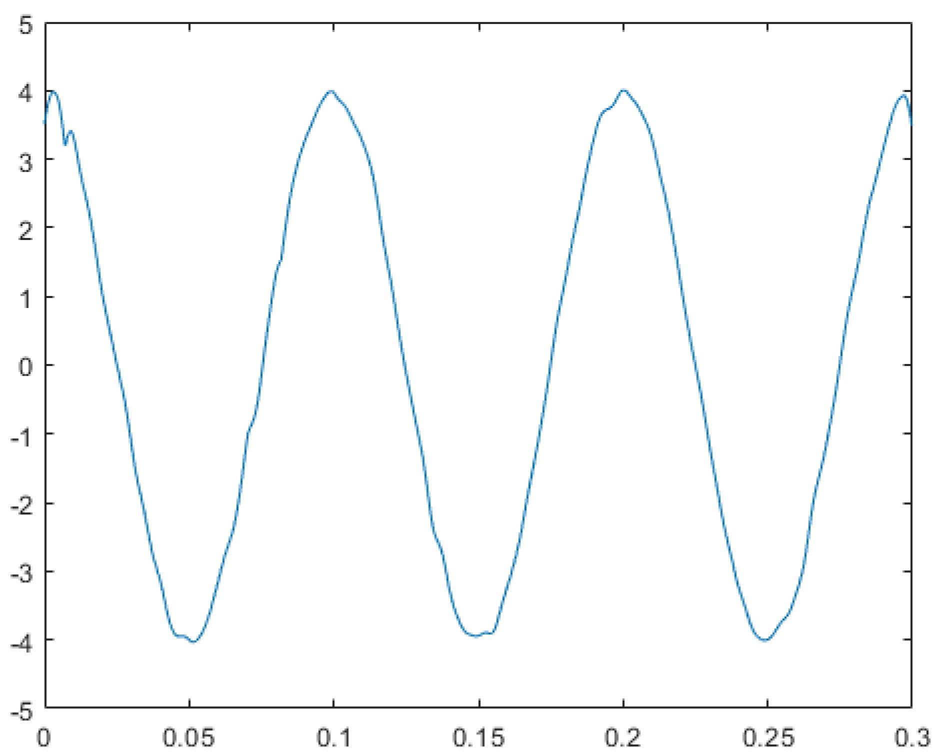
```
zero = zeros(1,1024);
noise_avg = [ ];
for k = 1:100
    noise = awgn(zero,signal_to_noise_ratio);
    val = mean(sqrt(noise.^2));
    noise_avg = [noise_avg val];
end

% Convert from dB to decimal
deviation = std(noise_avg)
average = mean(noise_avg)
```

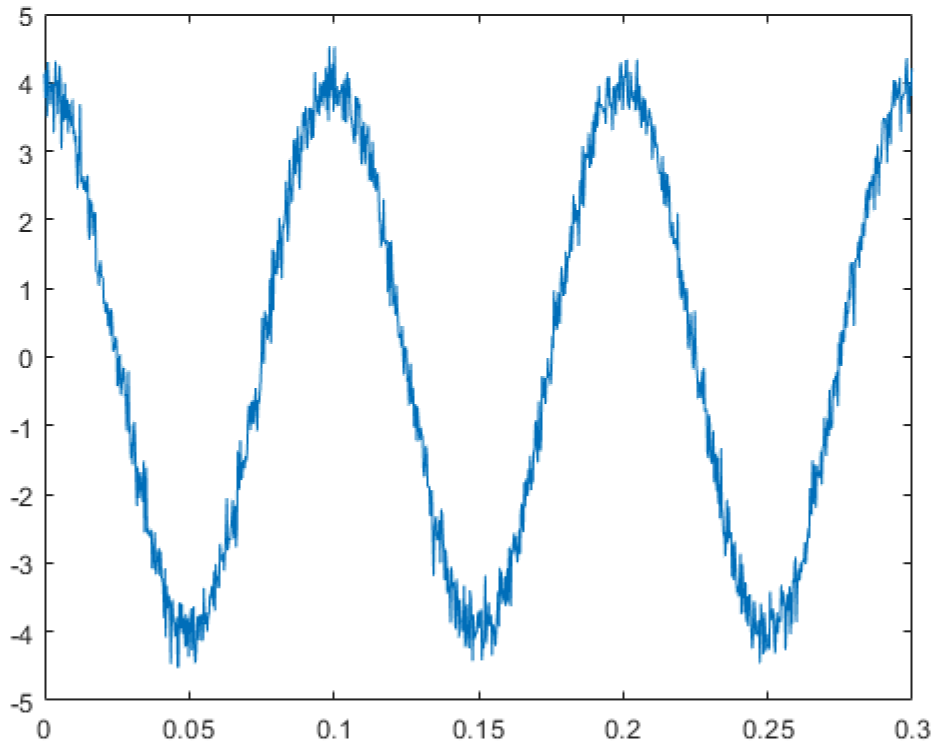
```
deviation =

    0.0025


average =

    0.1123
```

Algorithim execution

```
% Pass sensor locations, filtered sensor data, grid layout, and speed
 of
% sound into the geolocation algorithim

[guess, height] =
 algorithm(s0,s1,s2,s3,signal0,signal1,signal2,signal3,grid,speed_of_sound,deviati
```

```
T_score_of_detection =

   7.9663e+03

The system is    100

percent confident an avalanche infrasound signal is present

T_score_of_geolocation =

    1.7408

The system is    95.9126

percent confident it has correctly predicted the origin location
```

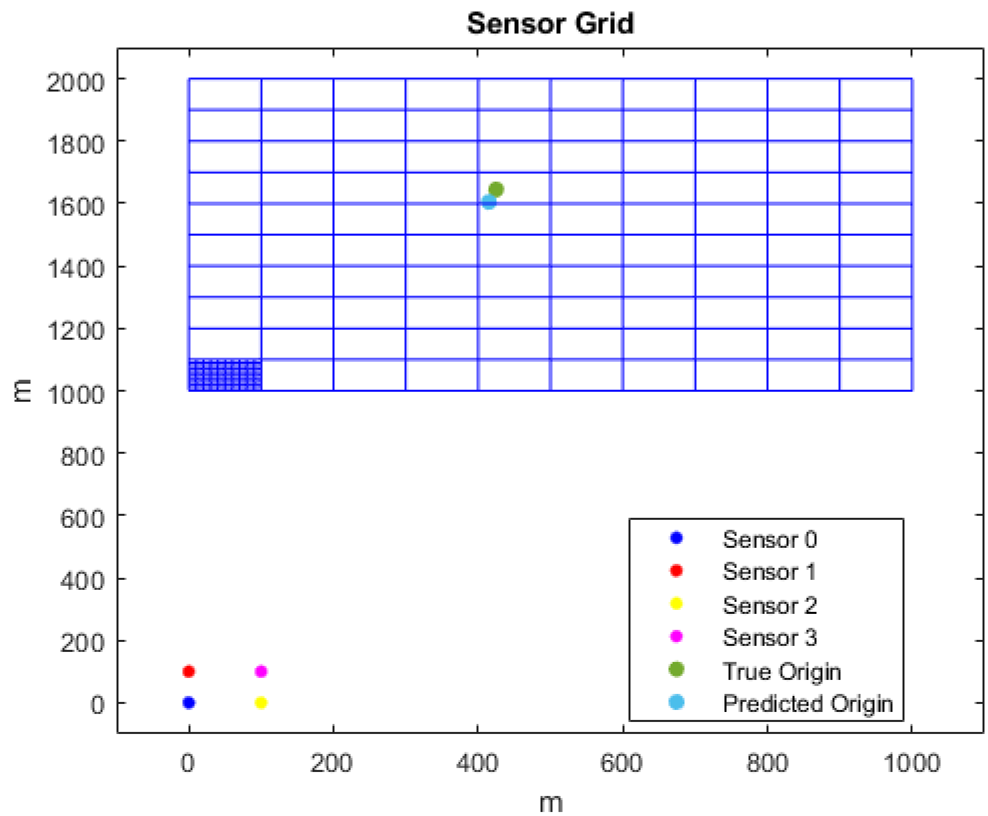




Plot

```
figure();
% Sensors
%gscatter([0 100 0 100],[0 0 100 100],[0;1;2;3]),
gscatter(0,0,'Sensor 0', 'b'),hold on
gscatter(0,100,'Sensor 1', 'r');
gscatter(100,0,'Sensor 2', 'y');
gscatter(100,100,'Sensor 3', 'm');
xlim([-100 1100]),ylim([-100 2100]);


% True origin
scatter([origin(1)],[origin(2)],'filled');
scatter([guess(1)],[guess(2)],'filled');
legend('Sensor 0', 'Sensor 1', 'Sensor 2', 'Sensor 3', 'True
 Origin','Predicted Origin');
title("Sensor Grid");
%xlabel("X (m)");
%ylabel("Y (m)");
% Grid border
%plot([0 0 1000 1000 0],[1000 2000 2000 1000 1000],'g','Linewidth',2)

% Grid points
x1 = [];
```

```
y1 = [];

% One square filled to 100x100 resolution
for x = 1:10
    for y = 1:10
        z = grid{x,y};
        k1 = [(z(1) - 5) (z(1) +5) (z(1) +5) (z(1) -5) (z(1) -5)];
        k2 = [(z(2) + 5) (z(2) +5) (z(2) -5) (z(2) -5) (z(2) +5)];
        x1 = [x1 k1];
        y1 = [y1 k2];
    end
    plot(x1,y1,'b','HandleVisibility', 'off'), hold on;
    x1 = [];
    y1 = [];
end

% 10x10 resolution
for x = 1:10
    for y = 1:10
        z = grid{x*10,y*10};
        k1 = [(z(1) - 50) (z(1) +50) (z(1) +50) (z(1) -50) (z(1) -50)]
 - 45;
        k2 = [(z(2) + 50) (z(2) +50) (z(2) -50) (z(2) -50) (z(2) +50)]
 - 45;
        x1 = [x1 k1];
        y1 = [y1 k2];
    end
    plot(x1,y1,'b','HandleVisibility','off'),xlabel("m"),ylabel("m")
    x1 = [];
    y1 = [];
end
hold off;
```



Error calculation

```
d_1 = distance(s0,origin);
d_2 = distance(s0,guess);

geolocation_percent_error = sqrt((d_2 - d_1)^2)/d_1 * 100;

fprintf('\n');
fprintf('\n');
fprintf("The actual error of the origin prediction is")
disp(geolocation_percent_error);
fprintf("percent")

fprintf('\n');
fprintf('\n');
```

```
The actual error of the origin prediction is    2.4267

percent
```

Prediction algorithm

```
function [predict, amp] =
 algorithm(s0,s1,s2,s3,signal_0,signal_1,signal_2,signal_3,grid,speed,deviation1,a

    amp = 0;
    predict = {1,1};

    orig0 = signal_0;
    orig1 = signal_1;
    orig2 = signal_2;
    orig3 = signal_3;

    % Low-pass filter each sensor's data, cutoff of 20hz
    signal_0 = lowpass(signal_0,20,3413);
    signal_1 = lowpass(signal_1,20,3413);
    signal_2 = lowpass(signal_2,20,3413);
    signal_3 = lowpass(signal_3,20,3413);

    data = [];

    % Iterate through all grid points
    for i = 1:100
        for k = 1:100

            % Calculate distance from current grid point to each
 sensor
            distance0 = distance(s0,grid{i,k});
            distance1 = distance(s1,grid{i,k});
            distance2 = distance(s2,grid{i,k});
            distance3 = distance(s3,grid{i,k});

            % Determine difference in distance to reach sensor 1-3
 compared
            % to reference sensor 0
            delta_1 = distance1 - distance0;
            delta_2 = distance2 - distance0;
            delta_3 = distance3 - distance0;

            % Calculate wavelength from speed of sound
            wave_length = speed/10;

            % Calculate phase shifts from wavelength
            shift_1 = delta_1/wave_length;
            shift_2 = delta_2/wave_length;
            shift_3 = delta_3/wave_length;

            % Shift signals 1-3 accordingly, in attempt to match
 signal 0
            signal1_shift = circshift(signal_1,round(-
shift_1*1024/3));
            signal2_shift = circshift(signal_2,round(-
shift_2*1024/3));
```

```
            signal3_shift = circshift(signal_3,round(-
shift_3*1024/3));



            orig1_shift = circshift(orig1,round(-shift_1*1024/3));
            orig2_shift = circshift(orig2,round(-shift_2*1024/3));
            orig3_shift = circshift(orig3,round(-shift_3*1024/3));

            % Sum all four signals
            beamformed = signal_0 + signal1_shift + signal2_shift +
 signal3_shift;
            beamformed_plot = beamformed;
            beamformed_orig = orig0 + orig1_shift + orig2_shift +
 orig3_shift;


            % Calculate root mean square ampltitude
            beamformed = (beamformed).^2;
            beamformed = sqrt(beamformed);
            amplitude = mean(beamformed);

            data = [data amplitude];


            % Highest amplitude result survives as the prediction
 until
            % another point produces one higher
            if amplitude > amp
                amp = amplitude;
                predict = grid{i,k};
                beamformed_plot_final = beamformed_plot;
                beamformed_orig_final = beamformed_orig;
                orig_snr = snr(beamformed_orig);
            end

        end
    end

    % Plot the beamformed signal
    figure();
    t = 0:1/3413:0.3;
    plot(t,beamformed_plot_final);

    figure();

    plot(t,beamformed_orig_final);

    % Calculate probability of signal detection
        T_score_of_detection = (orig_snr - average1)/(deviation1)
        prob = tcdf(T_score_of_detection,99) * 100;
        fprintf('The system is ');
        disp(prob);
```

```
        disp('percent confident an avalanche infrasound signal is
 present');

    % Calculate geolocation accuracy probability
    data_mean = mean(data);
    data_std = std(data);
    T_score_of_geolocation = (amp - data_mean)/data_std
    prob = tcdf(T_score_of_geolocation,9999) * 100;
    fprintf('The system is ');
    disp(prob);
    disp('percent confident it has correctly predicted the origin
 location');

end
```

Distance function definition

```
function dist = distance(p1,p2)
    a = p2(1);
    b = p2(2);
    dist = sqrt(abs((p2(1) - p1(1))^2 + (p2(2)-p1(2))^2));
end
```

Published with MATLAB® R2019b