```
% Ross Penniman
% EEN 502, Project 1, Part 1

clear all;
close all;

% At 20 deg C
c = 343.2;  % m/s

f0 = 500;    % Frequency in Hz
U = 20;  % Velociy of source (m/s)(about 45 mph)
p0= 1;  % Pressure amplitude, Pa (surface of sound source)

w = 2*pi*f0;
k = w/c;

Fs = 22050;
% Ts = 1/Fs;
Ts = 0.01;  % for plotting

t=-2:Ts:2;
Dt = U * t; % Distance wrt time

% Velocity of source, relative to observer wrt time
Ut = zeros(1, length(t));

% Pressure envelope ("+ 1" term to prevent divide by 0)
Et = p0 * (1./(abs(Dt)+1));

ft = zeros(1, length(t)); % Observed frequency wrt time
sig = zeros(1, length(t)); % Calculated signal

for m = 1:length(t)

    if t(m) < 0
        Ut(m) = U;
        ft(m) = f0 * (c/(c - U));
    else
        Ut(m) = -U;
        ft(m) = f0 * (c/(c + U));
    end
end

% sig = Et .* sin(2*pi.*ft.*t);
% sig = sig./(max(sig)*1.1);
% wavwrite(sig, Fs, 'part1.wav');
% wavplay(sig, Fs);

subplot(3,1,1);
plot(t,Ut);
axis([-2 2 -35 35]);
ylabel('Velocity, m/s');
title('Properties of Source as Seen by Observer');
```

```
subplot(3,1,2);
plot(t, Et);
axis([-2 2 0 1.2]);
ylabel('Pressure Envelope, Pa');

subplot(3,1,3);
plot(t, ft);
axis([-2 2 450 550]);
ylabel('Frequency, Hz')
xlabel('Time, seconds');

% print(gcf, 'part1_plot', '-dpng', '-r100')


**********************************************************

% Ross Penniman
% EEN 502, Project 1, Part 2

clear all;
close all;

% At 20 deg C
c = 343.2;  % m/s

f0 = 500;    % Frequency in Hz
U = 20;  % Velociy of source (m/s)(about 45 mph)
p0= 1;  % Pressure amplitude, Pa (surface of sound source)
yD = 2; % Distance in meters that observer is offset from path of source
        % (should be at least 1 m)

w = 2*pi*f0;
k = w/c;

Fs = 22050;
Ts = 1/Fs;
% Ts = 0.01;  % for plotting

t=-2:Ts:2;
Dx = U * t; % Distance wrt time along x axis

Dt = zeros(1, length(t)); % Distance to observer wrt time
Ut = zeros(1, length(t)); % Velocity of source, relative to observer
ft = zeros(1, length(t)); % Observed frequency wrt time
sig = zeros(1, length(t)); % Calculated signal

for m = 1:length(t)

    if Dx(m) == 0
        Dx(m) = 1e-10; % Prevent divide by 0
    end

    Dt(m) = sqrt(yD^2 + Dx(m)^2); % Distance wrt time
    theta = atan(yD/Dx(m));
```

```
    if t(m) < 0
        Ut(m) = U * cos(theta);
        ft(m) = f0 * (c/(c - Ut(m)));
    else
        Ut(m) = -U * cos(theta);
        ft(m) = f0 * (c/(c - Ut(m))); % Subtract Ut(m) because it is negative
    end
end

Et = p0 * (1./(abs(Dt))); % Pressure envelope

sig = Et .* sin(2*pi.*ft.*t);
sig = sig./(max(sig)*1.1); % Normalize the signal for good playback level
wavwrite(sig, Fs, 'part2.wav');
wavplay(sig, Fs);

% subplot(3,1,1);
% plot(t,Ut);
% axis([-2 2 -35 35]);
% ylabel('Velocity, m/s');
% title('Properties of Source as Seen by Observer');
%
% subplot(3,1,2);
% plot(t, Et);
% axis([-2 2 0 0.8]);
% ylabel('Pressure Envelope, Pa');
%
% subplot(3,1,3);
% plot(t, ft);
% axis([-2 2 450 550]);
% ylabel('Frequency, Hz')
% xlabel('Time, seconds');

% print(gcf, 'part2_plot', '-dpng', '-r100')



**********************************************************




% Ross Penniman
% EEN 502, Project 1, Part 3

clear all;
close all;

% At 20 deg C
c = 343.2;  % m/s

U = 20;  % Velociy of source (m/s)(about 45 mph)
p0= 1;  % Pressure amplitude, Pa (surface of sound source)
yD = 2; % Distance in meters that observer is offset from path of source
        % (should be at least 1 m)
```

```
fOsc = 5; % Oscillating frequency of siren (Hz)

Fs = 22050;
% Ts = 1/Fs; % for generating signal
Ts = 0.01;  % for plotting

t=-2:Ts:2;
Dx = U * t; % Distance wrt time along x axis
fSiren = 450 + 150 * sin(2*pi*fOsc.*t); % Frequency of siren in Hz

Dt = zeros(1, length(t)); % Distance to observer wrt time
Ut = zeros(1, length(t)); % Velocity of source, relative to observer
ft = zeros(1, length(t)); % Observed frequency wrt time
sig = zeros(1, length(t)); % Calculated signal

for m = 1:length(t)

    if Dx(m) == 0
        Dx(m) = 1e-10; % Prevent divide by 0
    end

    Dt(m) = sqrt(yD^2 + Dx(m)^2); % Distance wrt time
    theta = atan(yD/Dx(m));

    if t(m) < 0
        Ut(m) = U * cos(theta);
        ft(m) = fSiren(m) * (c/(c - Ut(m)));
    else
        Ut(m) = -U * cos(theta);
        % Subtract Ut(m) because it is negative
        ft(m) = fSiren(m) * (c/(c - Ut(m)));
    end
end

Et = p0 * (1./(abs(Dt))); % Pressure envelope

% fst = ft * Ts;  % Calculate cycles per sample
% fstI = zeros(1, length(fst));
% % Integrate fst to get total radians for each sample
% for m = 2:length(fst)
%     fstI(m) = fst(m - 1) + fstI(m - 1);
% end
%
% sig = Et .* sin(2*pi.*fstI);
% sig = sig./(max(sig)*1.1); % Normalize the signal for good playback level
% wavwrite(sig, Fs, 'part3.wav');
% wavplay(sig, Fs);

subplot(3,1,1);
plot(t,Ut);
axis([-2 2 -35 35]);
ylabel('Velocity, m/s');
title('Properties of Source as Seen by Observer');

subplot(3,1,2);
```

```
plot(t, Et);
axis([-2 2 0 0.8]);
ylabel('Pressure Envelope, Pa');

subplot(3,1,3);
plot(t, ft);
axis([-2 2 200 700]);
ylabel('Frequency, Hz')
xlabel('Time, seconds');

print(gcf, 'part3_plot', '-dpng', '-r100')


**********************************************************



% Ross Penniman
% EEN 502, Project 1, Part 4

clear all;
close all;

% At 20 deg C
c = 343.2;  % m/s

U = 20;  % Velociy of source (m/s)(about 45 mph)
p0= 1;  % Pressure amplitude, Pa (surface of sound source)
yD = 2; % Distance in meters that observer is offset from path of source
        % (should be at least 1 m)
fOsc = 5; % Oscillating frequency of siren (Hz)

Fs = 44100;
Ts = 1/Fs;

t=-2:Ts:2;
Dx = U * t; % Distance wrt time along x axis

t2=0:Ts:5;
fSiren = ones(1, length(t2)) * 500;
fst = fSiren * Ts;  % Calculate cycles per sample
fstI = zeros(1, length(fst));
% Integrate fst to get total radians for each sample
for m = 2:length(fst)
    fstI(m) = fst(m - 1) + fstI(m - 1);
end
sigSrc = sin(2*pi.*fstI); % Sound as emmited from siren

sig = zeros(length(t), 2); % Calculated signal

% Using a variable delay buffer to simulate the propagation
% time from the signal source to each ear (separate taps from
% the buffer for left and right signals). As the delay is
% shortened (for an approaching source), the pitch of the sound
```

```
% rises appropriately. In the same way, as the delay is lengthened
% (for a receding source), the pitch of the sound drops. This
% method also produces the correct time relationship between the
% left and right ears. Linear interpolation is used when reading
% samples out of the buffer to reduce noise artifacts.

chunk = round(0.2 * Fs);  % Size of delay buffer (0.2 seconds)
delayBuf = sigSrc(1:chunk);  % Initialize delay buffer
wrtCur = chunk; % Indicates the last location written to
readCurLH = 0;  readCurLL = 0;  % Left chan read cursors
readCurRH = 0;  readCurRL = 0;  % Right chan read cursors
weightL = 0;    weightR = 0;    % Weighting factors

for m = 1:length(t)
    % Calculate propagation delay
    DtL = (sqrt(yD^2 + (Dx(m) + 0.085)^2))/c;
    DtR = (sqrt(yD^2 + (Dx(m) - 0.085)^2))/c;

    % Calculate number of samples of delay
    delSampL = DtL * Fs;
    delSampR = DtR * Fs;

    % "weight" multiplies by the larger delay
    weightL = delSampL - floor(delSampL);
    weightR = delSampR - floor(delSampR);
    delSampL = floor(delSampL);
    delSampR = floor(delSampR);

    % Calculate read cursor indices
    readCurLL = wrtCur - delSampL;
    readCurLH = readCurLL - 1;
    readCurRL = wrtCur - delSampR;
    readCurRH = readCurRL - 1;

    % Wrap locations of read cursors as needed
    if readCurLL < 1
        readCurLL = readCurLL + chunk;
    end
    if readCurLH < 1
        readCurLH = readCurLH + chunk;
    end
    if readCurRL < 1
        readCurRL = readCurRL + chunk;
    end
    if readCurRH < 1
        readCurRH = readCurRH + chunk;
    end

    % Read output from delay buffer, apply linear interpolation
    sig(m,1) = (weightL * delayBuf(readCurLH)) + ...
        ((1 - weightL) * delayBuf(readCurLL));
    sig(m,2) = (weightR * delayBuf(readCurRH)) + ...
        ((1 - weightR) * delayBuf(readCurRL));

    % Update write cursor location
```

```
    wrtCur = wrtCur + 1;
    if wrtCur > chunk
        wrtCur = 1;
    end

    % Load next sample into delay buffer
    delayBuf(wrtCur) = sigSrc(m + chunk);

end

envDist = sqrt(yD^2 + (Dx.^2)); % Average distance of source
Et = p0 * (1./envDist); % Pressure envelope
sig(:,1) = sig(:,1) .* Et';
sig(:,2) = sig(:,2) .* Et';

maxSig = max(max(sig)) * 1.1;
sig = sig./maxSig; % Normalize the signal for good playback level

wavwrite(sig, Fs, 'part4.wav');
wavplay(sig, Fs);
```

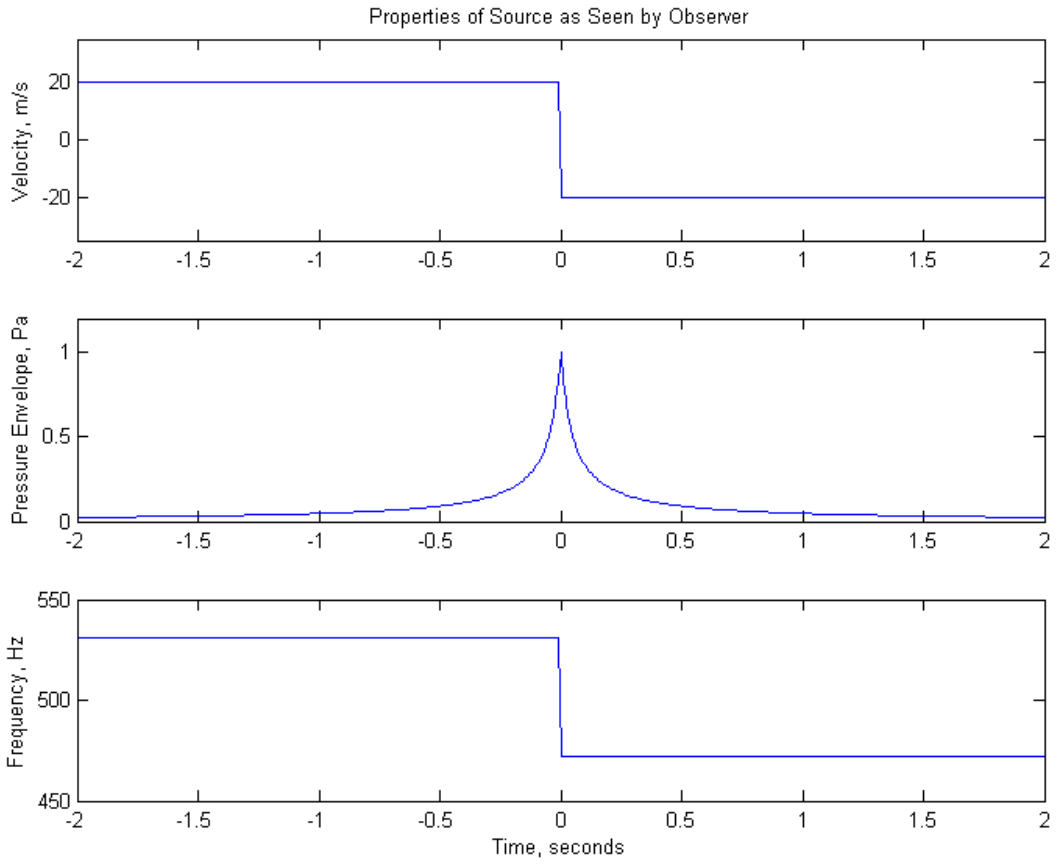
****

```
% Ross Penniman
% EEN 502, Project 1, Part 5

clear all;
close all;

% At 20 deg C
c = 343.2;  % m/s

p0= 1;  % Pressure amplitude, Pa (surface of sound source)
Fs = 4000;  % "sampling" frequency
Ts = 1/Fs;
f = 200;  % Frequency of source, Hz
w = 2*pi*f;
k = w/c;
L = c/f;  % Wavelength
U = 200;  % Velociy of source (m/s)
M = U/c;  % Mach number
tw = 0.04;  % time window, sec
l = linspace(-2.5*L, 2.5*L, 500); % Spatial dimension +/- 2.5 lambda
[x,y] = meshgrid(l);

video = VideoWriter('Proj1_5.avi', 'Motion JPEG AVI');

open(video);
figure;
```

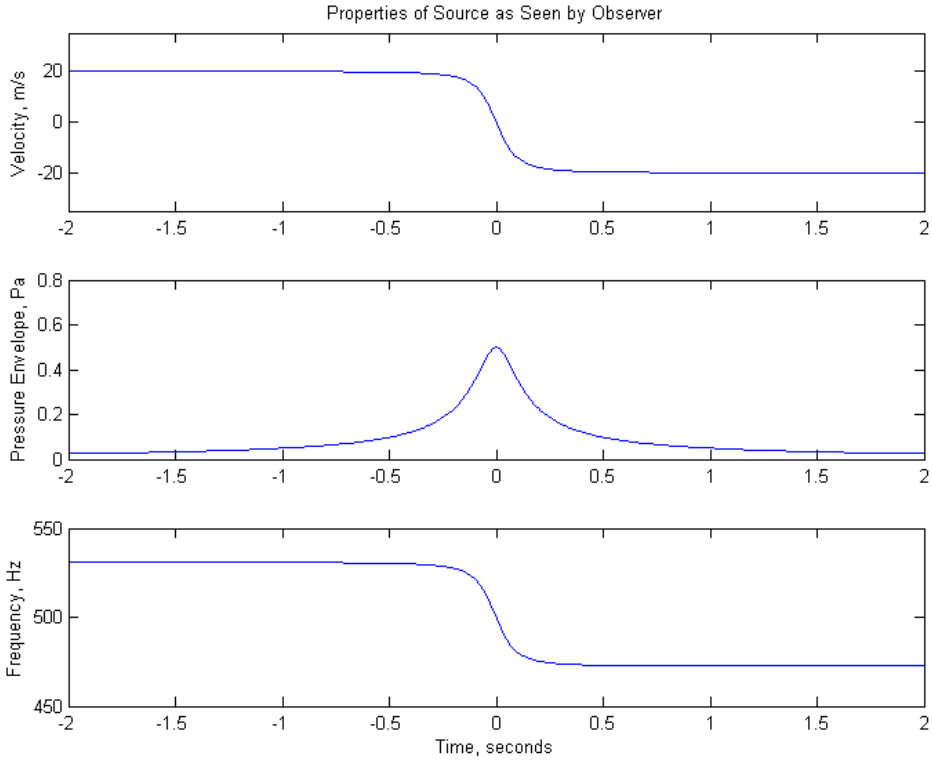
```
for t = -tw:Ts:tw
    r = sqrt((x - U*t).^2 + y.^2);
    p = (p0./(1 + r)).*exp(-j*k*x*M).*exp(-j*k*r)*exp(j*w*t);
    mesh(l, l, real(p));
    axis([-2.5*L, 2.5*L, -2.5*L, 2.5*L, -1.2 1.2]);
    caxis([-1 1]);
    title('Sound Field of Moving Source, 200 Hz, 200 m/s');
    xlabel('X, meters');
    ylabel('Y, meters');
    zlabel('Pressure, Pa');
    view(15, 55);
    writeVideo(video, getframe(gcf));
%     pause(0.01);
end

close(video);
```

****

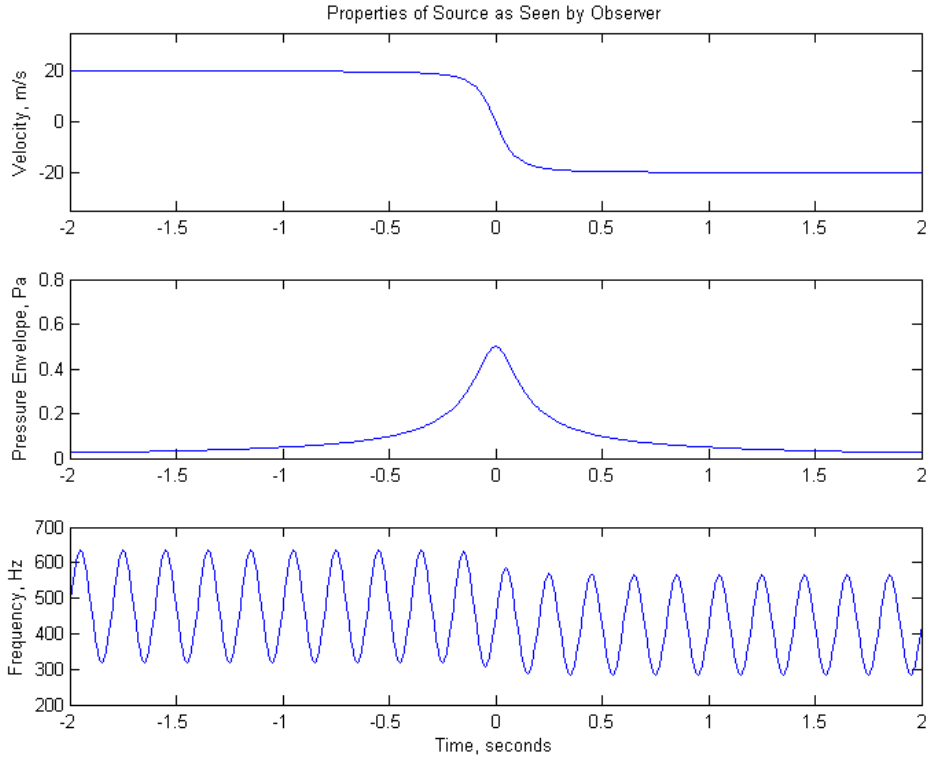
Part 1 Plot:



Part 2 Plot:

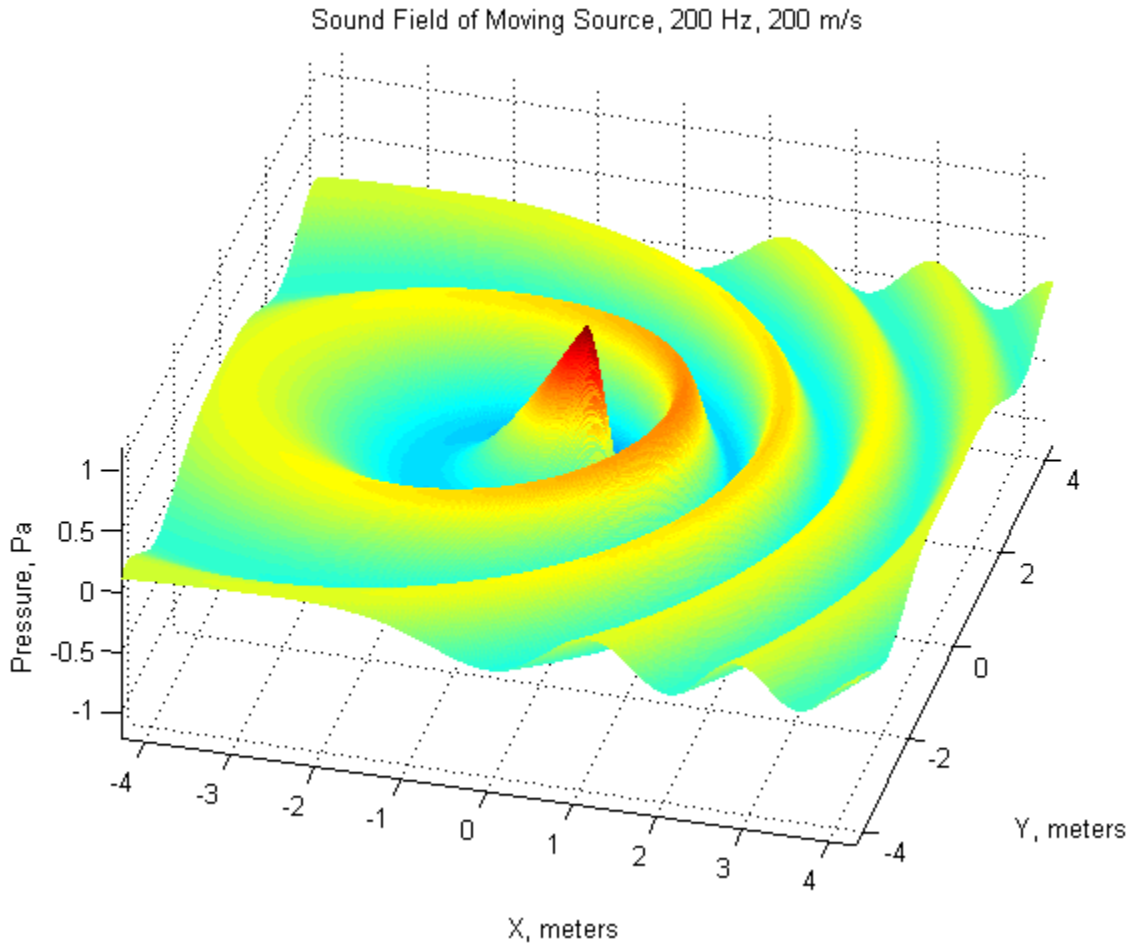


Part 3 Plot:



(Part 4 plots identical to Parts 2 and 3)

Part 5 Plot:



Observations and Conclusions:

One of the clear observations from parts 1 and 2 is the effect of the distance between the observer and the path of the source. Having the observer directly in the path of the source yields a very sharp spike in the envelope of the signal and also an instantaneous change of pitch as the source passes the observer. As the observer moves further away from the path of the source, however, the envelope becomes a smoother bell curve and the change in pitch is more gradual.

Parts 3 and 4 provided some interesting challenges in terms of signal processing to get the correct audio representation of the observed sound. The audio clip for part 4 is particularly effective when listening with headphones. Part 5 helped illustrate how the wavelength of a sound changes in two dimensions with respect to a moving source.