Readme File: MRM MATLAB Sample App (USB)

Overview:

This document describes the functions provided by the MATLAB Sample code. The code is intended to demonstrate operation of the P410 MRM (Radar) API over a USB link.

The code consists of a main script (example_commands.m) and a collection of useful subroutines. The main script exercises six demonstration programs. Each demonstration program calls several of the subroutines. The subroutines illustrate a variety of useful functions including examples of:

- How to view and change the configuration of the P410 radar
- How to cause the radar to generate radar scans
- How to parse the returning data
- How to motion filter and define a simple detection filter
- How to open and close a USB communications port
- How to use the Hilbert Transform to produce radar I and Q data streams
- How to use an FFT to extract Doppler data
- How to generate waterfall plots

All of these programs use standard MATLAB code. However, the final program also uses a Hilbert Transform. Note that the Hilbert Transform function is only available with MATLAB's Signal Processing Toolbox.

This document is divided in two parts: a description of how to connect and disconnect from a USB port and descriptions of the six example programs.

To start double click on the file: example_commands.m.

1. Making and terminating the USB connection

The command

```
srl = open_com_port(8)
```

is used to make connection to P410 through the USB port. This code assumes that your P410 is connected to Com Port 8. To determine which port you are actually connected to, you can use available tools on your OS (e.g. Device Manager on Windows) or your can connect with MRM RET. On connection, MRM RET indicates the port number.

Once that srl = open_com_port(8) has been properly updated to match the actual comm connection and not the assumed com8, connect to the P410 by clicking on line 7 of the code and then clicking <Ctrl><Enter>.

If the variable srl is somehow cleared you can use the command srl = instrfind to retrieve the variable.

To disconnect from the com port and P410, click <Ctrl><Enter> on line 48, fclose(srl). If you forget to disconnect from the P410 you may need to exit and reenter MATLAB.

Programming Note: The way in which the Host computer waits for messages from a USB connection varies by computer, supplies and operating system. The sample code has been set up such that each communication request to the P410 has an associated time out. That amount of time is defined by the K_{try} parameter. This value has been set to a size that works on most machines. If the user receives messages such as "Message scan data not returned", then increase the size of each occurrence of the parameter K_{try}.

2. Description of the six example demonstration programs.

plot_one_scn : loads the P410 with setup instructions, commands the unit to scan, retrieves the resulting radar scan and plots it. An example output is shown in **Figure 1**.

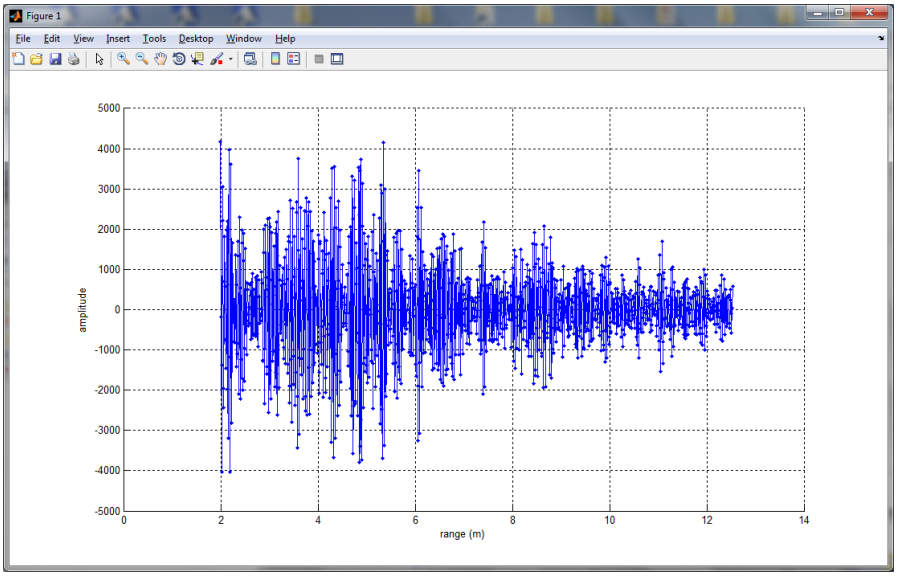

Figure 1: Output of plot_one_scn

plot_multi_scn(srl,Nrqst): loads the P410 with setup instructions, commands the unit to scan, retrieves the radar scan, computes the difference between the current scan and the previous scan, envelopes the data, plots the results and repeats 100 times. Since the time between the control request is determined by the MATLAB processing and any other CPU load on the host, the time between scans will likely vary, so the update rate is not guaranteed to be fixed.

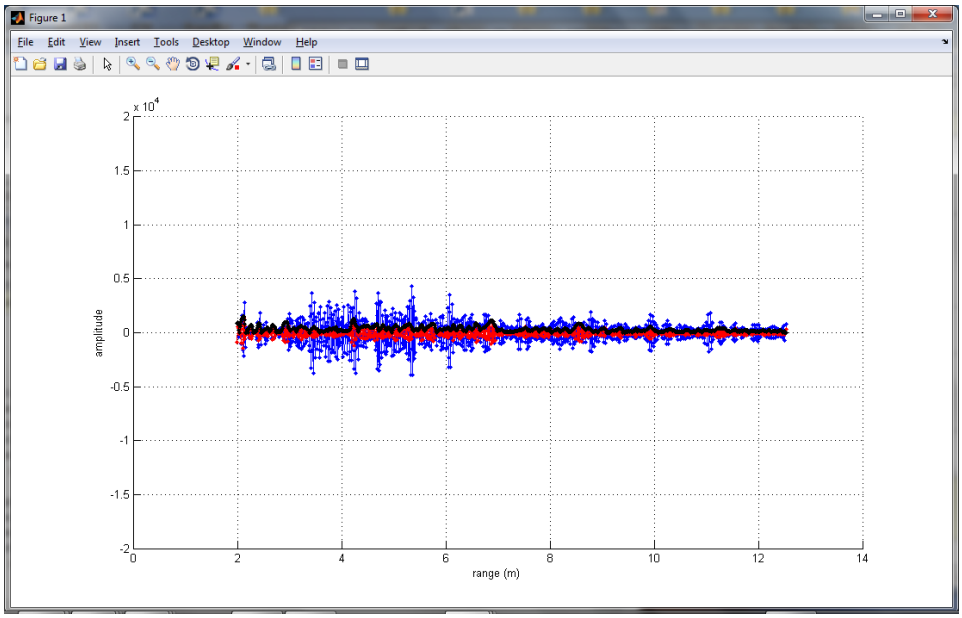

Figure 2: Current scan (blue), difference between first and current scan (red), envelope of the difference (black)

plot_one_double_scn(srl): loads the P410 with setup instructions, commands the unit to scan twice, retrieves the results, then plots the first scan, second scan and the difference between the two. Since the P410 has been commanded to scan twice, it will control the timing which defines when the scan is produced. The P410 timing is extremely precise. As a result, the update rate between scans is very accurate.

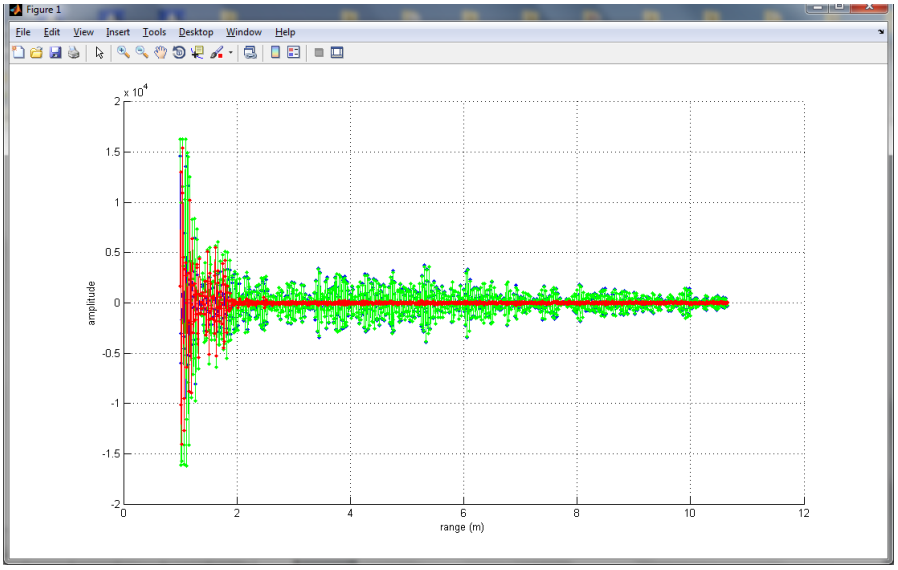

Figure 3: Scan 1 (blue), Scan 2 (green), difference (red)

plot_ multi_double_scn(srl): loads the P410 with setup instructions, commands the unit to scan twice, retrieves the results, then plots the first scan, second scan and the difference between the

two and then repeats this process 100 times. Since the P410 has been commanded to scan twice, it will control the timing which defines when the scan is produced. The time between each scan in a pair is precise, but the time between one pair and the next is not.

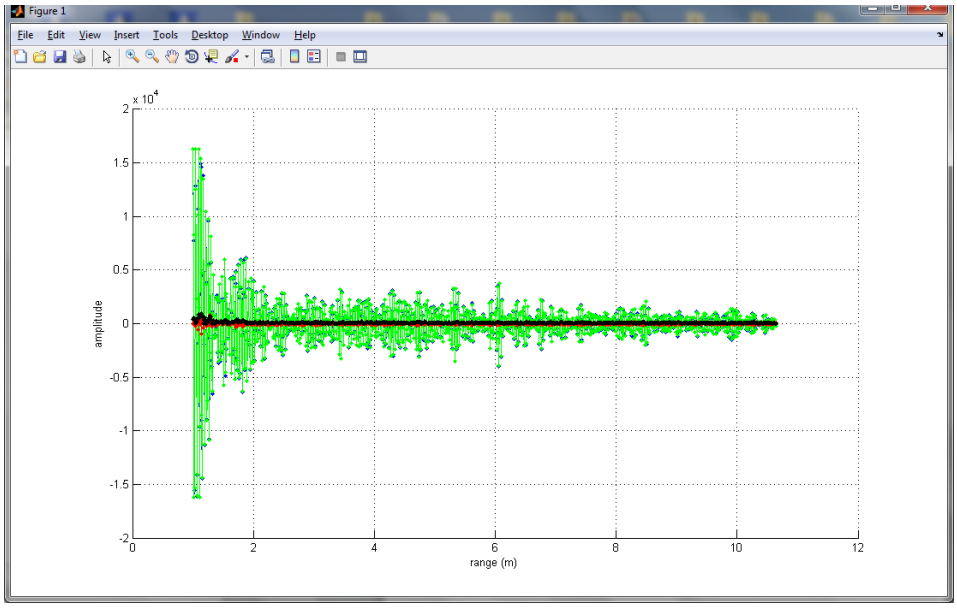

Figure 4: Scan 1 (blue), Scan 2 (green), difference (red), envelope of the difference (black)

img_ multi_scn(srl,100): loads the P410 with setup instructions, commands the unit to scan, retrieves the scan data, computes the difference between the current scan and the previous scan, envelopes the data, and plots the envelop as a raster in which the magnitude of the envelop is proportional to the color. It repeats this process 100 times such that each scan envelope is a separate raster. This final plot is frequently called a "waterfall" plot.

Waterfall plots are quite useful in that they make it easier for the human eye to pick out motion and features. The following data was taken as described below:

1) A P410 radar was located approximately 3 meters from the host computer.
2) The person started the Matlab.
3) The person paused for a few seconds and
4) then moved quickly towards the radar.

This activity can be seen in the **Figure 5**. The top waterfall plot shows the raw data, the bottom trace is identical but annotated.

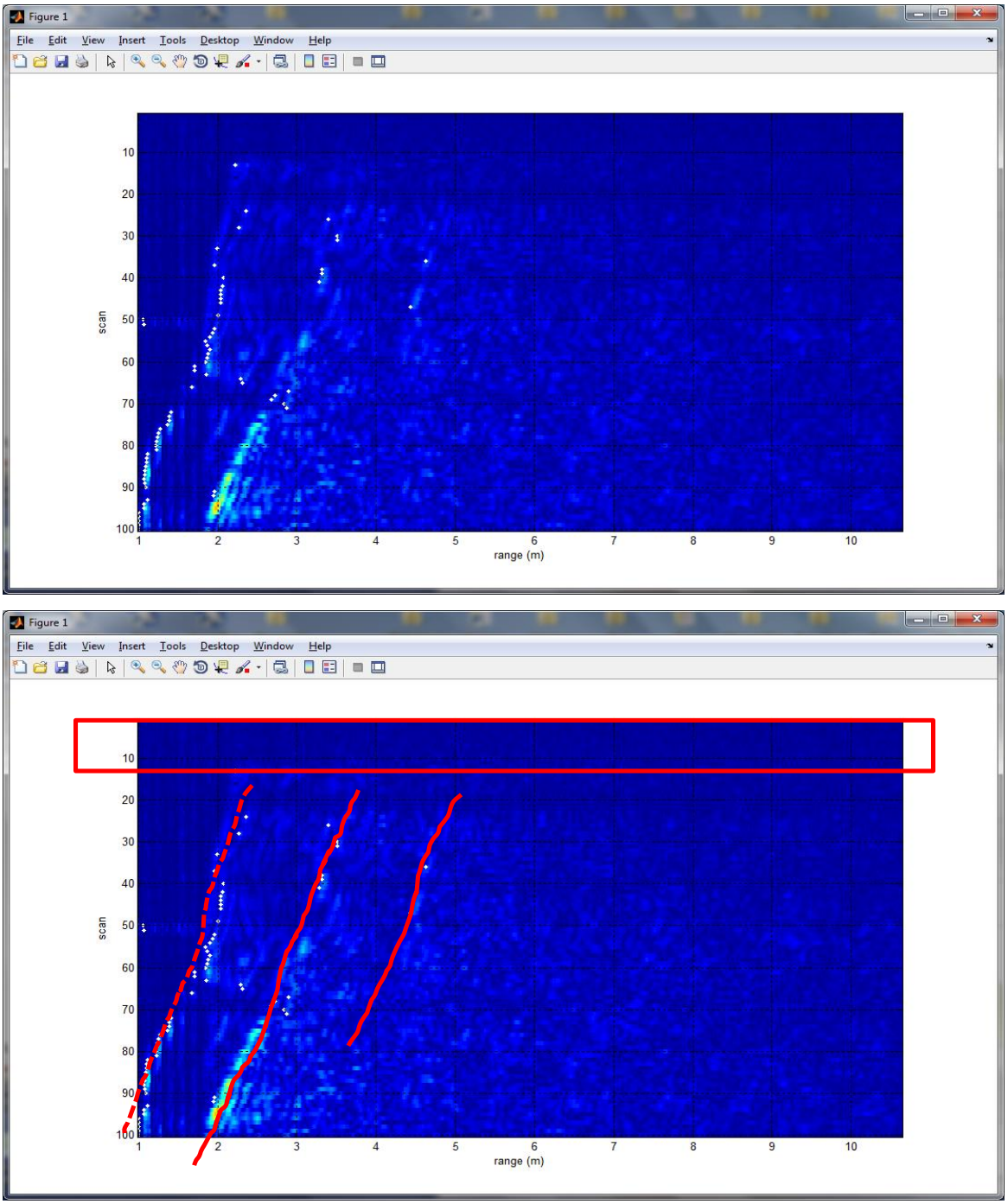

Figure 5: Raw data (top) Annotated data (below)

The first 15 seconds (marked with a red box) are very quiet. This corresponds to the time when the target was standing still. The target starts moving at 18seconds and moves closer to the target. At 90 seconds he passes by the radar. This is indicated by the dashed red line. However, as the target moves towards the target he is also creating a shadow. This shadow is changing with time and trails behind the target.

img_scn_fft(srl,Nrqst,Nfft): this code performs the following steps:

- loads the P410 with setup instructions,
- commands the unit to scan 32 times,
- retrieves the data,
- performs a 32 point FFT on the 1st bin in each scan (each data point in a scan is a bin), then performs a 32 point FFT on the 2nd bin in each scan, and so on for all bins in the scan,
- Plots the results as range (scan sample time * speed of light/2) vs FFT frequency (scaled to velocity).
- It also filters the zero velocity bin by zeroing it entirely. (The zero velocity entry corresponds to signal returns from all of the stationary items in the environment. Since there are a lot of stationary targets, the magnitude of their signal is massive and would overwhelm the more subtle signals from the moving target. Consequently, the program filters (zeroes) the zero velocity bin.)
- This process is performed 100 times.

Note the following:

- Since the data was generated by the P410 each of the 32 scans is fully coherent.
- The retrieved data is processed using a Hilbert Transform to produce radar I and Q data streams.
- The HilbertTransform can be found in the MATLAB Signal Processing Toolbox. If your version of MATLAB does not have the Signal Processing Toolbox, then you can expect the program to halt at this point and it will necessary to either a) to connect the Signal Processing Toolbox or b) implement a Hilbert Transform by hand. (This can be accomplished in a manner similar to a superheterodyne receiver. The user multiplies the scan by $\cos(2\pi f_o t)$ and $\sin(2\pi f_o t)$ for $f_o = 3.91$GHz and then low pass filter.)

The following three plots were taken using this program. The first plot (**Figure 6**) shows the output when nothing in the system is moving. The second (**Figure 7)** shows a target moving toward the radar. **Figure 8** shows a target moving away from the radar. As a note the energy at the longer ranges is the result of multipath reflections.

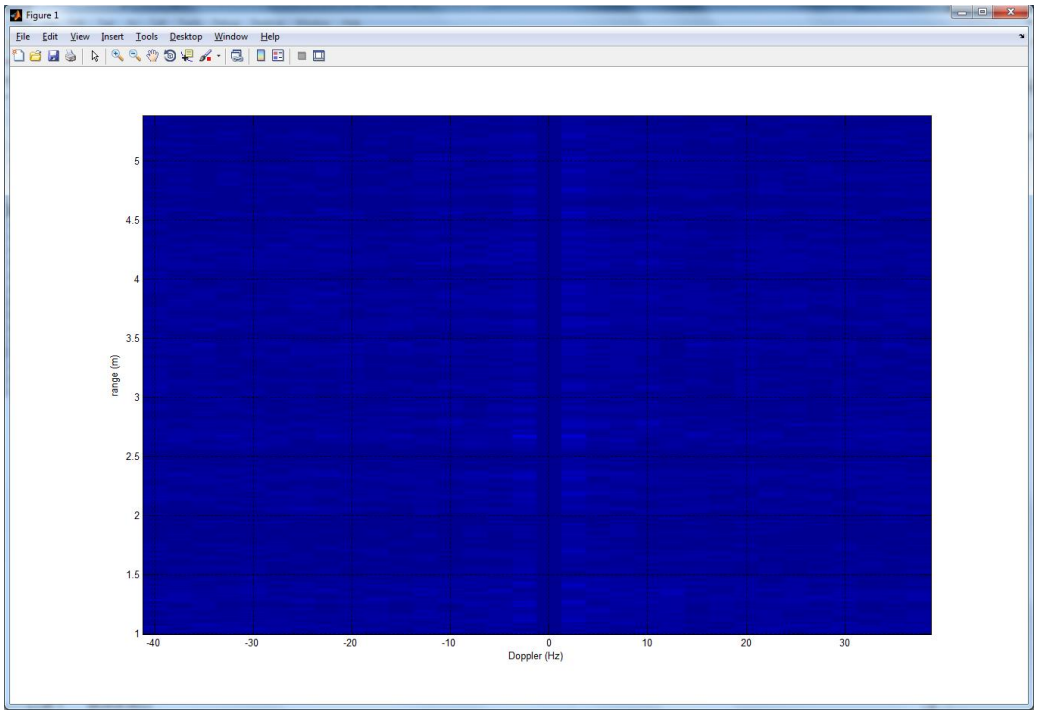

Figure 6: Output with no moving targets

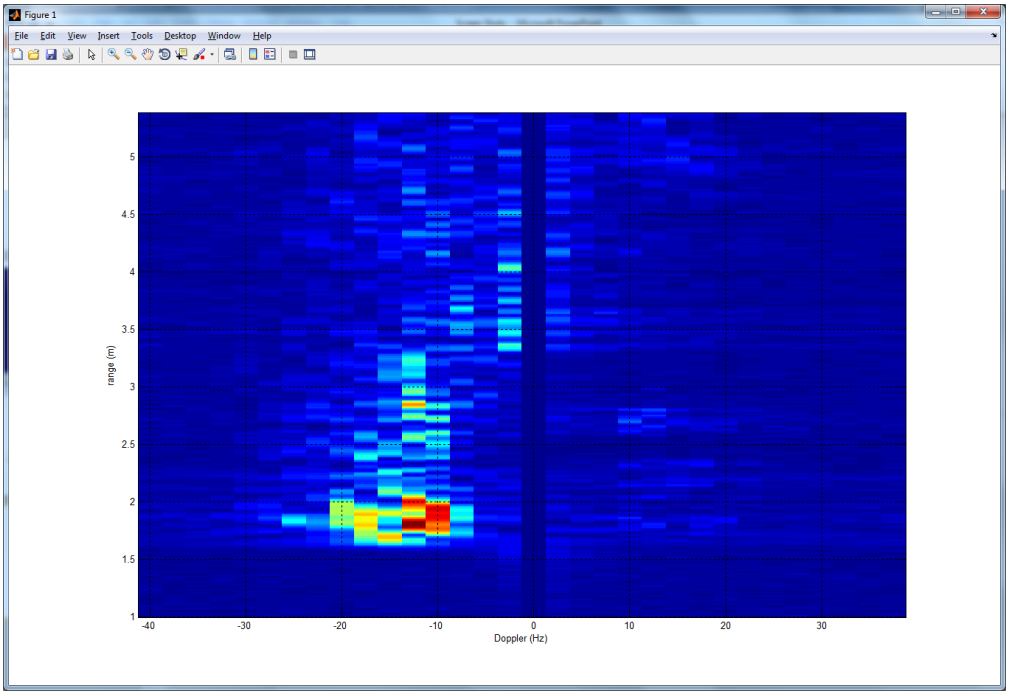

Figure 7: Output with target moving toward the radar

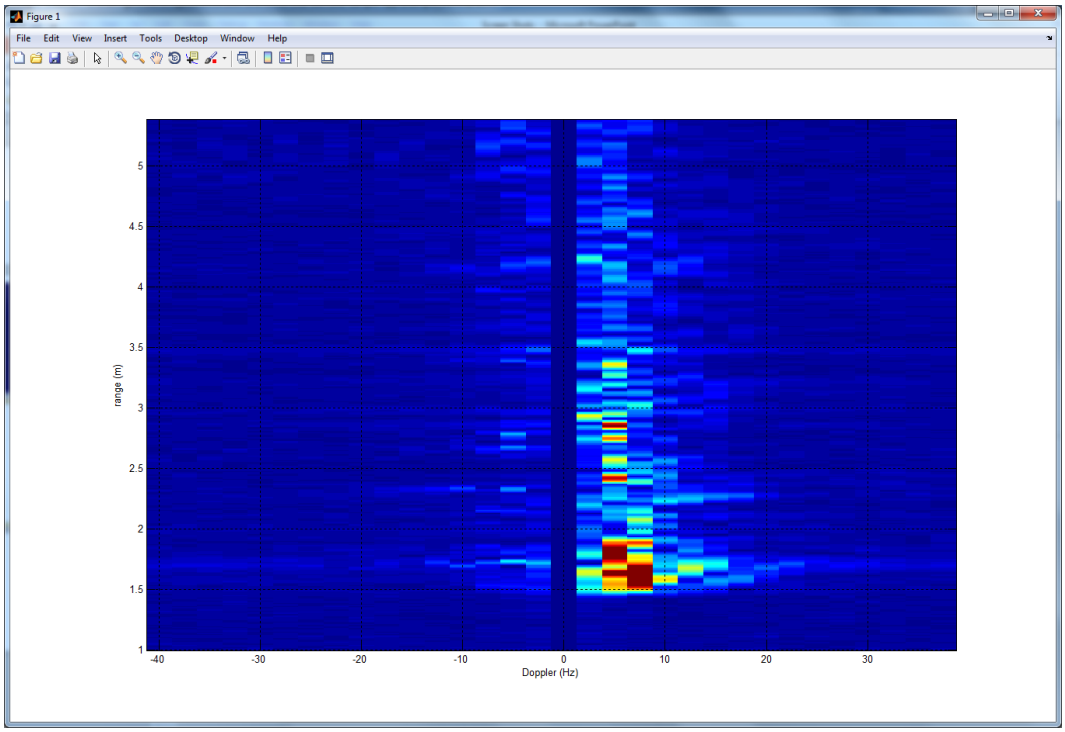

Figure 8: Output with target moving away from the radar