GEORGIA INSTITUTE OF TECHNOLOGY

SCHOOL OF ELECTRICAL ENGINEERING

ECE 6272 FALL 2010

COMPUTER PROJECT #3

Assigned: Friday, October 8, 2010

**Due Date for On-Campus Students: Tuesday, October 26 @ 9:35 AM**

**Due Date for DLPE Students: Tuesday, November 2 @ 4:00 PM Eastern Time**

* This project is to be done *individually.* Each student must develop his or her own computer code in its entirety. Students are not to discuss the theory or approaches to coding the theory with one another, nor are they to assist in debugging each other’s work. You may ask Dr. Richards questions regarding theory and implementation of the project, including asking them at the beginning or end of class, when others can benefit as well.
* MATLAB is the preferred language, but others are acceptable; the point is to try the experiments, not to improve your MATLAB skills.
* Data required for this project are available for download from the class T-Square site in the Computer Resources 🡪 Projects 🡪 Project Data and solutions area. Each student is assigned one of several specific data files to work with; see Section 4 for details. ***Whether you begin working right away or not, be sure you download your data set and make sure you can load it into MATLAB (or other computing environment) as soon as possible to avoid last minute difficulties. Also, if you are NOT using MATLAB, be sure to note special instructions in Section 4*.**
* Reports will be graded on completeness in addressing the assignment and quality of results. The main table of results is a large fraction of the grade. Reports will not be graded on programming style or efficiency or on writing quality, except that the programming and the writing should be clear enough to be reasonably understandable. Questions or clarifications about the assignment should be directed to Dr. Richards.[[1]](#footnote-1) Errata, revisions and hints (if any) will be made available via the class T-Square site or during class.

# PROBLEM

You will be given a fast-time/slow-time matrix of raw baseband I&Q radar data that contains returns from exactly four moving targets as well as interference (noise and clutter). This project requires that you develop a radar signal processing algorithm that will allow you to detect each of the targets, and to estimate the range, radial velocity, and relative amplitude of each one found.

**Be sure to read everything below carefully. There are many details in the remaining sections that can help you get better results.**

**Also, this is the most involved of the computer projects. It is NOT a good idea to wait until the last minute to get started!**

# REQUIREMENTS

You must submit a brief, informal report of your methods and findings that must include:

1. An overview description of how your processing algorithm works. This must include a *block diagram* of the major processing steps with a description of the rationale for each step, but need not include minute details of the implementation.

2. A listing of the computer program used to make your measurements. This is sufficient to convey most details. Code must be sufficiently modular and well-commented to be understood readily. Listings must also be included for any functions used which are not built into either MATLAB or its Signal Processing Toolbox.

3. Results showing how the program performs on the raw data. You must specifically provide the following information:

* The name of the data file that you used.
* The number of targets actually found in the data (ideally, four; but if your algorithm identifies more or fewer than 4, admit it!).
* The estimated range to each target, *in kilometers*. Targets are to be listed in order of increasing range.
* The estimated velocity of each target, *in meters/sec*. This must include the sign of the velocity. Positive velocities represent targets approaching the radar; negative velocities represent receding targets.
* Identify which target has the largest radar cross section (RCS), and then give the estimated RCS of the others *relative to the largest* in dB. Thus, your strongest target will have a relative RCS of 0 dB. No absolute values of RCS are required.
* Answer the following question: assuming no aliasing of target velocities (*i.e.*, actual Doppler shifts are limited to ±*PRF*/2 Hz), what is the maximum change in range due to range-Doppler coupling? Are your estimates of target range affected by this range-Doppler coupling?

**The 2nd through 4th items in this list must be assembled into a single table in the first section of your report!**  Entries in the table should be in order of increasing range. A *limited number* of hard copy plots should be provided to substantiate your conclusions.

4. Three more details:

* You may assume that the range and Doppler measurements are *not* ambiguous, *i.e.* all ranges are less than *cT*/2 meters, and all Doppler shifts are in the range of ±*PRF*/2 Hz.
* Interpret your Doppler axis as ranging over ±*PRF*/2 Hz, not over (0,*PRF*) Hz. In other words, assume the maximum magnitude of the Doppler shift is *PRF*/2 Hz and allow for both approaching and receding targets.
* Additional information is given in Section 6 concerning the accuracy desired in the range, velocity, and RCS measurements. More accurate measurements produce better grades, but once you achieve the desired accuracies or better, further improvement is not necessary.

Start working early. If you encounter problems, please ask questions to clarify anything that is vague.

# REPORT FORMAT

To aid in the grading, please observe the following format constraints:

* The first section of your report should be a summary table of the information requested in item #3 in the list above, *i.e.* the name of your data file, and a table of the ranges, velocities, and relative amplitudes for each target found. The targets must be listed in the table in order of increasing range. Be absolutely certain you give your results in the requested engineering units (range in km, velocity in m/s, relative amplitude in dB. Also answer the question about range-Doppler coupling above.
* All code should be kept together at the end.
* Any additional plots, diagrams, and explanations should be between these two things.

# DATA

All of the data you need is obtained by downloading the Winzip file Project\_3\_data.zip available from the class T-Square site in the Computer Projects section. When unzipped, this will produce the following files:

* Five unique radar data files. Each is a MATLAB .mat file, with a name of the form data*n*\_mys.mat, where *n* = 1, 2, 3, 4, or 5. Thus the file names are data1\_mys.mat, data2\_mys.mat, *etc.* Each data file contains one 337x20 array of complex data samples; the name of this data variable is y. Also included are a few other variables representing some of the parameters needed to interpret the data as further described below. If you are not using MATLAB, see below for further instructions.
* A file named data\_file\_assignment.pdf. This is an Adobe pdf file listing each student (on-campus and distance learning) in the class by name; beside each name is one of the digits 1 through 5. Look up your name and use the data file corresponding to the digit by your name. That is, if the digit “3” appears by your name, use the data file data3\_mys.mat to do your work. You will not need the other data files, though you are welcome to try them out as additional tests of your code. You will be graded, however, only on the results obtained for the assigned data file. Do not present results for any other data file in your report. *If you are unsure which file you are supposed to work with, please ask!*
* **UNDER NO CIRCUMSTANCES SHOULD YOU PROVIDE ANY INFORMATION ON YOUR TARGET RANGE/DOPPLER/RCS ESTIMATES FOR YOUR DATA FILE OR FOR ANY OTHER DATA FILE TO ANY OTHER STUDENT IN THIS CLASS. To do so is a violation of the Georgia Tech Honor code.**
* Copies of the M files radar.m and git\_chirp.m included at the end of this problem assignment.

If you are not using MATLAB, I will provide on request the equivalent text files data*n*r\_mys.txt and data*n*i\_mys.txt to replace data*n*\_mys.mat. The file data*n*r\_mys.txt will contain only the real part of the 337x20 data matrix y. Each line of the file will be one row of the data matrix. Each line will thus contain 20 values, represented by simple text strings and separated by spaces. There will be a total of 337 lines of data. Similarly, the file data*n*i\_mys.txt will contain the imaginary part of y. You will need to combine these to form the final complex data matrix. Exactly how you do this will depend on your programming environment. The other variables included in the .mat files for MATLAB users are listed explicitly below and can be entered into your code manually. I will not provide these files unless specifically requested by e-mail.

***Even if you don’t start work right away, be sure to download your data file(s) and make sure you can load and work with it as soon as possible!***

# SIMULATION

The parameters of the radar simulation (frequency, PRF, waveform, range, *etc.*) are given in Table 1. These were chosen to match those that might be expected in a basic medium PRF (pulse repetition frequency) target search mode of an airborne multimode radar application*, e.g.* the type of radar in a typical fighter aircraft. This scenario would require a pulse burst waveform to measure velocity accurately; on the other hand, the maximum range is limited due to the relatively high PRF used in the burst. You should keep this scenario in mind as a check of the reasonableness of your estimated velocities and ranges.

The data file data*n*\_mys.mat available for processing contains radar data for a scenario containing exactly four point targets having varying ranges, relative amplitudes, and velocities. The data is synthetic, and contains signal, thermal noise, and ground clutter components. The signal component was produced by the m-file radar.m. If all goes well it is unnecessary, but you can consult the listing of this file (listing attached; electronic copy in Project\_3\_data.zip) for some clues as to details of how the simulated target returns were actually derived. The noise and clutter were then added to the data from radar.m by another program that I have not provided to you, however, the details should be unimportant. Some of the parameters of the radar and its transmitted LFM (linear FM, or “chirp”) radar signal are given in the table below, along with the names of the corresponding variables that should be established when you load data*n*\_mys.mat. Furthermore, the baseband chirp signal itself is also included in data*n*\_mys.mat; its variable name is s. For non-MATLAB users, you can reproduce the chirp s from the parameters supplied in Table 1 and the algorithm given in git\_chirp.m (listing attached; electronic copy in Project\_3\_data.zip).

Table . Radar Parameters of the Transmitted Waveform.

|  |  |  |  |
| --- | --- | --- | --- |
| **RADAR PARAMETERS** | | | |
| *Parameter* | Data*n*\_mys*.*mat *Variable* | *Value* | *Units* |
| Radar Frequency | fc | 10 | GHz |
| Pulse Length | T | 10 | **sec |
| Swept Bandwidth | W | 10 | MHz |
| Fast-time Sampling Frequency | fs | 12 | MHz |
| Pulse Repetition Frequency | PRF | 10 | kHz |
| Number of Pulses | Np | 20 | pulses |
| Time Delay to Start of Receive Window | T\_out[1] | 12 | **sec |
| Time Delay to End of Receive Window | T\_out[2] | 40 | **sec |

# DISCUSSION OF ADDITIONAL DATA AND PROCESSING ISSUES

***Signal Processing***

It will be necessary to analyze the radar waveform for its delay and Doppler content. This will require the following steps at least, though not necessarily in the order given:

*Matched filtering*. This requires that you recreate the transmitted LFM chirp pulse. From this transmitted pulse, the impulse response of the matched filter can be obtained. If you are using MATLAB, you have a copy of the chirp waveform in the variable s. If you are not using MATLAB, you can re-synthesize the chirp using an equivalent to the MATLAB function git\_chirp.m, which will produce an arbitrary LFM pulse; it is the same function used to produce a chirp for synthesizing the data. A copy is attached. You may want to consider including a window in the matched filtering process to reduce range sidelobes, using techniques developed in project #2 on matched filtering of LFM waveforms. On the other hand, you don’t have to use a window if it seems unnecessary for the required measurements.

*Doppler analysis*. It is necessary to perform a spectrum analysis of the data in the slow time dimension to extract the target Doppler frequencies. It is possible that there may be more than one target at the same range, so the only way to separate them will be by distinguishing their different Doppler frequencies. You must use a DFT to perform your spectrum analysis. You may want to restrict the analysis to ranges where there are likely to be targets, or you may want to process all of the data as an actual processor would do. Plots are useful for trying to identify “likely” ranges. Identify the major velocity components and amplitudes from peaks in the frequency domain plots. Make sure that you consider the possibility of positive and negative velocities. Also, realize that portions of the Doppler response of targets having Doppler shifts near ±*PRF*/2 may “wrap around” on the Doppler axis; be careful not to interpret one target as two separate ones if this occurs. You may want to compute more spectral samples than you have data points in order to improve the sampling density of the Doppler spectrum; see the subsection on “velocity accuracy” below for additional considerations on DFT size. You may also want to use a window function to reduce Doppler sidelobes. On the other hand, you don’t have to use a window if it seems unnecessary for the required measurements.

*Range-Doppler coupling*. Since we are using an LFM waveform, there will be range-Doppler coupling. By thinking about the maximum possible Doppler shift that you can measure with these parameters, and the size of the range bins, you can decide if range-Doppler coupling could be significant or not, that is, if it could distort your range measurements significantly (where “significantly” means enough to violate the range accuracy requirements given below). If so, you may want to try to compensate for it based on your velocity measurements. If not, you can ignore it.

*Clutter filtering*. The returned radar signal contains a large distributed ground clutter component, as would be the case it the radar was looking down at the ground. Consequently, some clutter filtering may be advisable to make the target peaks more evident. Two- or three-pulse cancellers are one approach that could be used. On the other hand, if the clutter is not interfering with your ability to detect and accurately measure the targets, clutter filtering may not be required.

*Order of Operations.* The primary operations you will likely consider are linear, and therefore you should get the same end results no matter what order they are applied in. However, some orderings might make it easier to some targets, or to see them earlier in the processing, than others. See the discussion of clutter filtering and stationary targets in the section on interference (below) for an example. You may choose the order of operations as you think appropriate.

*Peak detection*. The valid peaks in range and/or Doppler need to be identified by a peak picking algorithm. Automatic peak picking requires that you define a fixed threshold or one that depends on the data in some way, typically as a multiple of the estimated mean interference power. In this project, visual identification of the peaks is OK, especially since we have not yet discussed adaptive detection, but you should be cautious not to mistake sidelobes of one target or the clutter as another target. If you detect peaks very closely clustered in range and Doppler, you should consider the possibility that they represent only a single target.

*Range accuracy*. The spacing of samples in range (fast time) is determined by the fast time sampling rate fs given above. Target ranges should be measured to the nearest range bin.

*Velocity accuracy*. With only 20 pulses in the dwell, the Doppler resolution of the simulated radar is not particularly good. If you use a small FFT, the samples on the Doppler axis may be fairly far apart, so that straddle loss may become a problem in both amplitude estimation and velocity estimation. You might consider improving the estimation accuracy for the Doppler frequency of a target by interpolating the peak using either larger FFTs or a polynomial interpolation in the peak vicinity. A velocity estimation error of more than 2 m/s is fair, 1 m/s is good, and 0.5 m/s is very good. Thus, you may want to consider what size DFT will give you velocity sample spacings of no more than 0.5 m/s.

*RCS accuracy*. The relative RCS of the target echoes is affected not only by their relative RCS, but also by their relative range (because of *R*4 power losses from the radar range equation) and the effect of the MTI filter, *H*(**) on the target. Thus you will need to adjust the apparent amplitudes for their ranges in order to convert to relative RCS. Also, depending on exactly what stage in your data processing chain you estimate the amplitudes, the MTI filter (if you use one) may be distorting the relative amplitude of your targets and therefore you may need to compensate for that effect to get accurate estimates of relative RCS. An RCS estimation error of no more than 2 dB is fair, 1 dB is good, and 0.5 dB is very good. Failing to identify the correct target as the highest relative RCS is a significant error.

*Conversion to engineering units*. You *must* convert all your range plots and Doppler frequency plots to the correct engineering units (*i.e.*, Hertz, meters/sec, kilometers, *etc*.) This involves the use of the sampling interval (in range or Doppler as appropriate), the time delay to the initial range sample, and the number of Doppler spectral samples used. Getting these conversions correct is a significant detail in this project, so it requires your careful attention. Your final answers for target velocity must be in m/s, not equivalent Doppler frequency (Hz). Also, additional delay introduced by the matched filtering must be properly accounted for, as it was in project #2. It is easy to make “off by one” errors here, so give careful thought to labeling your range axis.

***Receive Window***

The radar returns can be spread over a very wide time span if the range coverage needs to be large. In order to avoid ambiguities, the range window must be limited to the time interval between successive pulses. The maximum unambiguous range is *Rmax* = *cPRI*/2 = *c*/2*PRF*; the minimum (or “eclipsing”) range is dictated by the length of the pulse (the receiver can’t start listening until the transmitter is finished transmitting the entire pulse), and is *c*/2. In the simulation function radar.m, a receive time window can be specified, so as to limit the number of returned samples that must be processed. The receive window used here of 12 **s to 40 **s means that no data samples are taken until 10 **s after transmission of the pulse begins; samples are then taken at an appropriate rate which depends on the waveform bandwidth (12 Msamples/sec here) until 40 **s after the pulse was transmitted. The set of fast-time samples so collected becomes one column of the data matrix (*i.e.*, the “range bins” or “range samples” or “range gates” or “range cells” for one pulse). The process is repeated for each pulse in the burst until the complete data matrix has been assembled. Defining a modest-sized receive window gives us a way to make the data set a manageable size.

***Interference***

The data contains two forms of interference:

*Thermal receiver noise* that is modeled as independent, but equal variance, zero-mean white Gaussian noise processes in both the I and Q channels. It is completely uncorrelated from one pulse (slow-time sample) to the next, and from one range bin (fast-time sample) to the next.

*Distributed ground clutter*. On a pulse-to-pulse (slow-time) basis, this type of interference is correlated (otherwise it would be. Effectively, just noise), and is often (but not always) removed by prefiltering with a canceller. It is usually considered independent from one range bin (fast-time) to the next, or nearly so, on the grounds that each range bin represents returns from physically distinct clutter. In this project, the ground clutter was created by forming a white random process having a log-normal amplitude distribution (typical of some types of real clutter) and uniform random phase. The desired correlation from one pulse to the next *in a fixed range bin* was created by forcing the spectrum of the samples from each range bin to have a narrow Gaussian shape. The clutter remains uncorrelated, however, from one range bin to the next.

For a single individual pulse, the signal to noise ratio (*SNR*) and/or signal to clutter (*SCR*) ratio of the echoes from some targets may be less than one. Therefore, the uncompressed pulse may be below the interference and can only be identified after some integration gain is achieved by either pulse compression or Doppler processing. It may be interesting to plot the absolute values of the columns of the data before any processing to see if there is any evidence of a target visible. Weaker targets may not be obvious even after one stage of processing (pulse compression or Doppler processing), and therefore cannot be identified until *both* range and Doppler processing are complete. On the other hand, clutter filtering, if used, will eliminate stationary targets. Such targets, if any, therefore will not be detected unless you look for them *before* clutter filtering, *and* they are large enough to stand out above the clutter. Fortunately, you are guaranteed that your four targets are all moving, so you don’t have to worry much about this.

*A suggestion*: Try plots of different type (contour, mesh, 1D) to examine your data at various stages in your processing and see how it comes together (or fails to!) to form the target peaks. (However, only include in your report those plots that turn out to be useful to illustrate key points or operations. The summary table is the main output I am looking for in this project.) I personally find range-Doppler mesh and contour plots (provided the contour levels are reasonably well-chosen) to be very instructive. A range-Doppler plot is obtained by taking the DFT of each slow-time row, so that the fast-time (range) – slow-time (pulse number) matrix becomes a range-Doppler matrix. Other students in previous offerings of this class have found other formats useful as well, so use what works for you.

Listing of radar.m

function y = radar( x, fs, T\_0, g, T\_out, T\_ref, fc, r, snr, v )

% RADAR simulate radar returns from a single pulse or burst

% of identical pulses

% usage:

% R = radar( X, Fs, T\_0, G, T\_out, T\_ref, Fc, R, SNR, V )

% X: baseband single pulse waveform (complex vector)

% Fs: sampling frequency of input pulse [in Hz]

% T\_0: start time(s) of input pulse(s) [sec]

% (number of pulses in burst assumed = length(g) )

% G: complex gain(s) of pulse(s)

% T\_out: 2-vector [T\_min,T\_max] defines output

% window delay times w.r.t. start of pulse

% T\_ref: system "reference" time, needed to simulate

% burst returns. THIS IS THE "t=0" TIME !!!

% Fc: center freq. of the radar. [in Hz]

% R: vector of ranges to target(s) [meters]

% (number of targets assumed = length(r) )

% SNR: vector of target SNRs (unit noise power assumed)

% This will be SNR \*after\* allowing for R^4

% V: vector of target velocities (optional) [in m/sec]

% (positive velocities are towards the radar)

%

% note(1): VELOCITY in meters/sec !!!

% distances in m, times in sec, BW in Hz.

% note(2): assumes each pulse is constant (complex) amplitude

% note(3): will accomodate up to quadratic phase pulses

% note(4): vector of ranges, R, allows DISTRIBUTED targets

%

% (c) jMcClellan 7/28/90

% Modified by M. A. Richards, August 1991

J = sqrt(-1);

c = 3e8; % velocity of light in m/sec

Mx = length(x);

delta\_t = 1/fs; % sampling interval (sec)

t\_y = [ T\_out(1):delta\_t:T\_out(2) ]'; % output sampling times (sec)

T\_p = Mx\*delta\_t; % length of input pulse (sec)

% Assume zero velocities (stationary targets) if no velocity

% vector provided

if nargin < 7

v = zeros(r);

end

% ensure that all vectors are column vectors

x=x(:); g=g(:); T\_0=T\_0(:); r=r(:); snr=snr(:); v=v(:);

% determine the quadratic phase modulation parameters for

% later interpolation of pulse samples

t\_x = delta\_t\*[0:(Mx-1)]';

x\_ph = unwrap(angle(x));

q = polyfit(t\_x,x\_ph,2);

% check result using correlation coefficient

xfit = polyval(q,t\_x);

if (x\_ph'\*xfit)/norm(x\_ph)/norm(xfit) < 0.99

disp('pulse phase is not quadratic')

keyboard

end

%

%--- Form (initially empty) output matrix ---

%

Mr = length(t\_y); Nr = length(g); % output samples in a matrix

y = zeros(Mr,Nr);

% Index 'i' loops over the number of targets

for i = 1:length(r)

ri = r(i);

vi = v(i);

f\_doppler = 2\*vi\*fc/c;

% Index 'j' loops over the number of pulses

for j = 1:length(g)

r\_at\_T\_0 = ri - vi\*T\_0(j);

% Compute start and end time of reflected pulse at receiver,

% ensure that it falls at least partially within the range (time) window

tau = 2\*r\_at\_T\_0/(c+vi); tmax = tau + T\_p;

if tau >= T\_out(2) | tmax <= T\_out(1)

fprintf('\nEcho from target #%g at range %g km',i,ri)

fprintf('\nis COMPLETELY OUT OF the range window')

fprintf('\non pulse #%g.\n',j)

else

% Figure out which sample locations in the output grid contain

% reflected pulse

t\_vals = t\_y - tau;

n\_out = find( t\_vals >= 0 & t\_vals < T\_p );

if tau < T\_out(1)

fprintf('\nEcho from target #%g at range %g km',i,ri)

fprintf('\nSTARTS BEFORE the range window')

fprintf('\non pulse #%g.\n',j)

end

if tmax > T\_out(2)

fprintf('\nEcho from target #%g at range %g km',i,ri)

fprintf('\nFINISHES AFTER the range window')

fprintf('\non pulse #%g.\n',j)

end

% Place scaled, range-delayed, Doppler shifted pulse into output matrix

% Unit noise power and unit nominal pulse amplitude assumed to

% get amplitude from SNR.

amp = 10^(snr(i)/20);

y(n\_out,j) = y(n\_out,j) + ...

( amp \* g(j) \* exp( -J\*2\*pi\*fc\*tau ) ) ...

.\* [ exp( J\*2\*pi\*f\_doppler\*t\_y(n\_out) ) ] ...

.\* [ exp( J\*polyval(q,t\_vals(n\_out)) ) ];

end % end of "if tau >= T\_out(2) ...."

end % end of loop over j

end % end of loop over i

Listing of git\_chirp.m

function x = git\_chirp( T, W, p )

%CHIRP generate a sampled chirp signal

% X = git\_chirp( T, W, <P> )

% X: N=pTW samples of a "chirp" signal

% exp(j(W/T)pi\*t^2) -T/2 <= t < +T/2

% T: time duration from -T/2 to +T/2

% W: swept bandwidth from -W/2 to +W/2

% optional:

% P: samples at P times the Nyquist rate (W)

% i.e., sampling interval is 1/(PW)

% default is P = 1

%

if nargin < 3

p = 1;

end

J = sqrt(-1);

%--------------

delta\_t = 1/(p\*W);

N = round( p\*T\*W ); %--same as T/delta\_t, but rounded

nn = [0:N-1]';

% x = exp( J\*pi\*W/T \* (delta\_t\*nn - T/2).^2 ); % old version

x = exp( J\*pi\*W/T \* (delta\_t\*nn - (N-1)/2/p/W).^2 ); % symmetric version

% even older version

%%%%% alf = 1/(2\*p\*p\*T\*W);

%%%%% git\_chirp = exp( J\*2\*pi\*alf\*((nn-N/2).\*(nn-N/2)) );

1. Office: Klaus 3354, 404-894-2714, <mark.richards@ece.gatech.edu>. Office hours TBD, but drop-ins and appointments welcome. [↑](#footnote-ref-1)