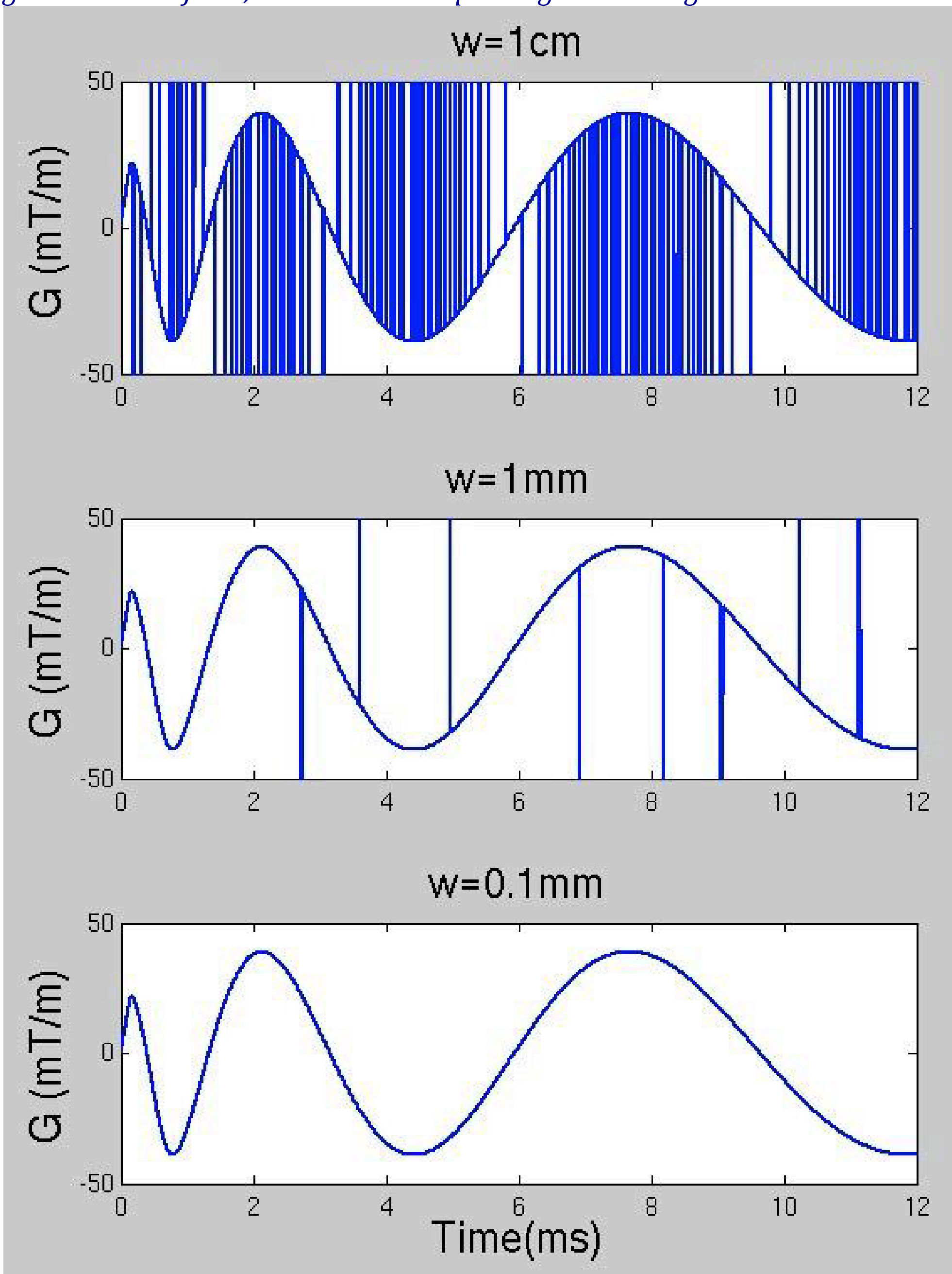
gradient waveform, or intravoxel dephasing leads to signal loss.



2. **MR Fingerprinting:** Here we will look at a very simple form of MR fingerprinting that uses a <u>gradient-spoiled</u> sequence with TR/TE=10,2 ms and a randomized RF flip angle. For simplicity define <a,b> as the inner product **sum(a.*conj(b))** in Matlab. Load the file **mrf.mat** and note the contents of the file: **dict** is a "dictionary" whose columns are the evolution of signal for the given flip angle train **fa** (magnitude in degrees, and phase is the phase of the RF pulse), and the corresponding T_1 and T_2 in **T1list** and **T2list** in ms.

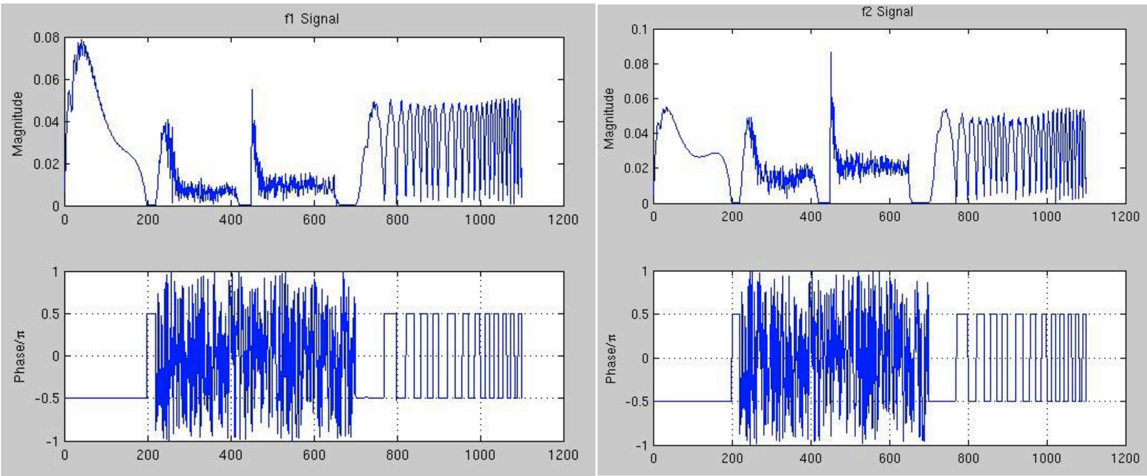
a. Build an EPG simulation that produces the signal evolution as a function of T_1 and T_2 for the array of N flip angles, **fa**. Note the fact that the flip angle includes a phase for the RF! Call this evolution f(t,T1,T2). *Note that you could use this to create the dictionary* ***dict*** *above, but we did it for you!*

The Matlab code is here:

```
%
%       function s = mrfevol(T1,T2,fa)
%
%       Calculate MR "fingerprint" for T1 and T2.
%       Assume a gradient-echo sequence with TR=10ms and
%       TE=2ms.  T1 and T2 are in ms.  fa is in degrees.
%


function s = mrfevol(T1,T2,fa)

TR = 10;
TE = 2;

N = length(fa); % # TRs
F = [0;0;1];    % Equilibrium Magnetization.
                % [F+; F-; Z],  all longitudinal in Z0 state.


S1 = zeros(N,1);

fa = pi/180*fa;

for k=1:N
  F = epg_rf(F,abs(fa(k)),angle(fa(k)));        % RF
  S1(k) = F(1,1);                      ;        % Record TE=0 signal.
  F = epg_grelax(F,T1,T2,TR,1,0,1,0);           % Relaxation, spoiling
end;

s = S1 * exp(-TE/T2);
```

b. Find the magnitude and phase of this evolution for T1=1000, T2=200 and for T1=300, T2=80, call these (complex) vectors f1 and f2. Normalize f1 by sqrt(<f1,f1>) and f2 by sqrt(<f2,f2>) and plot the magnitude and phase of each, as a function of TR number. *You should check these against the signal in* ***dict*** *generated for similar T1/T2!*

Plots are shown below, and obviously match ***dict*** *since this uses the same code! Note that they are not all that different.*



c. Using the normalized values from (b), calculate <f1,f2>.

<f1,f2> = 0.95 (The "fingerprints" are not very orthogonal!)

d. Now calculate <f1,dict> for all columns of dict, and find the column for which the magnitude of the inner product is maximized. In the arrays **T1list, T2list**, what does this correspond to? (ie what is your estimate of T_1 and T_2?) Repeat for <f2,dict>.

For f1: (T1,T2) = (992 ,201)
For f2: (T1,T2) = (317 ,79)

e. Add noise **sg*randn(size(fa))** to the original (non-normalized) **f1** and **f2** and plot the noisy signal (magnitude and phase), then do part (d) for values **sg** = 0.005, 0.01, 0.02 and 0.05.
(Answers may vary due to random noise!)

Sg	***0.005***	***0.01***	***0.02***	***0.05***
T1,T2 for f1	*992,201*	*992,201*	*992,201*	*827,187*
T1,T2 for f2	*317,87*	*270,82*	*270,82*	*367,101*

3. **Fat/Water Separation and B_0 mapping:** Begin here by loading the file **me.mat**, which contains k-space data that is (k_x,k_y,TE) with a TE spacing of 0.5ms. There are 16 echoes for a duration of ~8ms, which is convenient. You can reconstruct the images with dat = fftshift(fft(fftshift(dat,n),[],n),n); where n is the dimension that you are Fourier transforming over (ie 1, and 2, and maybe 3 in some cases). For the problem, you will not use all 16 echoes (though you could reconstruct images over x,y,f for fun!).

   a. Fourier transform in x and y to form images. Display the magnitude images for echo times 1,2,3 and 4.