Quadrature Sampling

Mac Radigan

1 Preliminaries

1.1 Context of Discussion

```
#!/usr/bin/octave -q
%% parameters.m
%%
%% Specifies the paramters used throughout this discussion.

  function y = parameters()

    fs    = 1e4;       % [Hz]       sampling frequency
    fc    = 3e2;       % [Hz]       carrier frequency
    fi    = 1e2;       % [Hz]       intermediate frequency
    A     = 1.0;       % [unitless] gain for signal of interest
    fk    = fc + 1e2;  % [Hz]       frequency for signal of interest
    phi   = 0.45 * pi; % [rad]      phase angle for signal of interest
    T     = 5e-2;      % [s ]       duration
    t     = 0:1/fs:T;  % [s ]       time vector

    y = struct(   ...
      'fs',  fs,  ...
      'fc',  fc,  ...
      'fi',  fi,  ...
      'A',   A,   ...
      'fk',  fk,  ...
      'phi', phi, ...
      'T',   T,   ...
      't',   t    ...
      );

  end % parameters

%% *EOF*
```

1.2 Even and Odd Functions

1.2.1 Definition of Even and Odd Functions

even function:

$$f(x) = f(-x) \tag{1}$$

odd function:

$$f(-x) = -f(x) \tag{2}$$

1.2.2 Trigonometric Symmetries

$$cos(x) = cos(-x) \tag{3}$$

$$sin(-x) = -sin(x) \tag{4}$$

1.3 Inner Products

conjugate symmetry

$$\langle \cdot, \cdot \rangle : \mathbb{V} \times \mathbb{V} \mapsto \mathbb{F} \tag{5}$$

$x, y, z \in \mathbb{V}$ and $\alpha \in \mathbb{F}$
conjugate symmetry

$$\langle x, y \rangle = \overline{\langle y, x \rangle} \tag{6}$$

linearity

$$\begin{gathered} \langle \alpha x, y \rangle = \alpha \langle x, y \rangle \\ \langle x + y, z \rangle = \langle x, z \rangle + \langle y, z \rangle \end{gathered} \tag{7}$$

positive-definateness

$$\begin{aligned}\langle x, x\rangle &\geq 0 \\ \langle x, x\rangle = 0 &\leftrightarrow x = 0\end{aligned} \tag{8}$$

1.3.1 Inner Product Defined

For the space of points in $\mathbb{R}^n$, the inner product is defined as

$$\begin{aligned}\left\langle \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}, \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} \right\rangle &\triangleq x^{\mathrm{T}} y \\ &= \sum_{k=1}^{n} x_k y_k \\ &= x_1 y_1 + \cdots + x_n y_n\end{aligned} \tag{9}$$

For the space of continuous complex functions on the interval $[a, b]$, i.e. $\mathbb{C}\,[a, b]$, the inner product is defined as

$$\langle f, g\rangle \triangleq \int_a^b f\,(t)\,\overline{g\,(t)}dt \tag{10}$$

1.3.2 Properties of Inner Products

$$\langle x, x\rangle = \overline{\langle x, x\rangle} \tag{11}$$

$$\begin{aligned}\langle x, -x\rangle &= -1\langle x, x\rangle \\ &= \overline{-1}\langle x, x\rangle \\ &= \langle x, -x\rangle\end{aligned} \tag{12}$$

1.4 Trigonomic Identities

1.4.1 Angle Sum and Differences

$$\begin{aligned}\sin(\alpha \pm \beta) &= \sin\alpha\cos\beta \pm \cos\alpha\sin\beta \\ \cos(\alpha \pm \beta) &= \cos\alpha\cos\beta \mp \sin\alpha\sin\beta\end{aligned} \tag{13}$$

1.4.2 Identities Involving Complex Exponentials

Euler's formula:

$$e^{ix} = \cos x + i \sin x \tag{14}$$

Written in terms of cosine and sine:

$$\begin{aligned} \cos x &= \frac{e^{ix} + e^{-ix}}{2} \\ \sin x &= \frac{e^{ix} - e^{-ix}}{2i} \end{aligned} \tag{15}$$

1.5 Fourier Transform

1.5.1 Fourier Transform Defined

forward transform:

$$\mathbb{F}\{s(t)\} = S(f) \triangleq \int_{-\infty}^{+\infty} s(t)\, e^{-2\pi i f t} dt$$

inverse transform:

$$\mathbb{F}^{-1}\{S(f)\} = s(t) \triangleq \int_{-\infty}^{+\infty} S(f)\, e^{2\pi i t f} df$$

1.6 Fourier Transform as the Superposition of Inner Products

1.7 Fourier Analysis of a Single Frequency Component

1.7.1 Fourier Transform of Analytic Signals, Single Frequency

$$\begin{aligned} \mathbb{F}\{e^{2\pi i f_k t + \phi}\} &= \int_{-\infty}^{+\infty} e^{2\pi i f_k t + \phi} e^{-2\pi i f t} dt \\ &= \int_{-\infty}^{+\infty} e^{2\pi i (f_k - f) t} e^{\phi} dt \\ &= 2\pi e^{-i\phi} \delta(f - f_k) \end{aligned}$$

1.7.2 Fourier Transform of Real Signals, Single Frequency

$$
\begin{aligned}
\mathbb{F}\{\mathbb{RE}\{e^{2\pi i f_k t+\phi}\}\} &= \mathbb{F}\{\mathbb{RE}\{\cos\left(2\pi i f_k t+\phi\right)+i\sin\left(2\pi i f_k t+\phi\right)\}\} \\
&= \mathbb{F}\{\cos\left(2\pi i f_k t+\phi\right)\} \\
&= \mathbb{F}\{\frac{e^{2\pi i f_k t+\phi}+e^{-2\pi i f_k t+\phi}}{2}\} \\
&= \frac{1}{2}\left[\int_{-\infty}^{+\infty} e^{2\pi i f_k t+\phi} e^{-2\pi i f t} dt + \int_{-\infty}^{+\infty} e^{-2\pi i f_k t+\phi} e^{-2\pi i f t} dt\right] \\
&= \frac{1}{2}\left[\int_{-\infty}^{+\infty} e^{2\pi i (f_k - f) t} e^{\phi} dt + \int_{-\infty}^{+\infty} e^{2\pi i (-f_k - f) t} e^{\phi} dt\right] \\
&= 2\pi\left[e^{-i\phi}\delta\left(f+f_k\right)+e^{i\phi}\delta\left(f-f_k\right)\right]
\end{aligned}
$$

1.8 Hilbert Transform

1.8.1 Hilbert Transform Defined

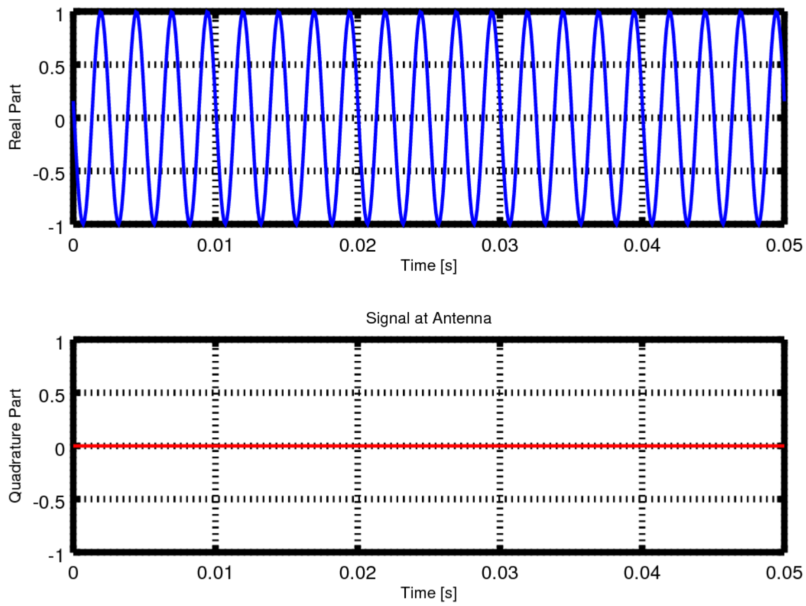
$$
\begin{aligned}
H\{g\left(t\right)\} &= g\left(t\right) \\
&= \frac{1}{\pi}\int_{-\infty}^{+\infty}\frac{g\left(\tau\right)}{t-\tau}d\tau \\
&= \frac{1}{\pi}\int_{-\infty}^{+\infty}\frac{g\left(t-\tau\right)}{\tau}d\tau
\end{aligned}
$$

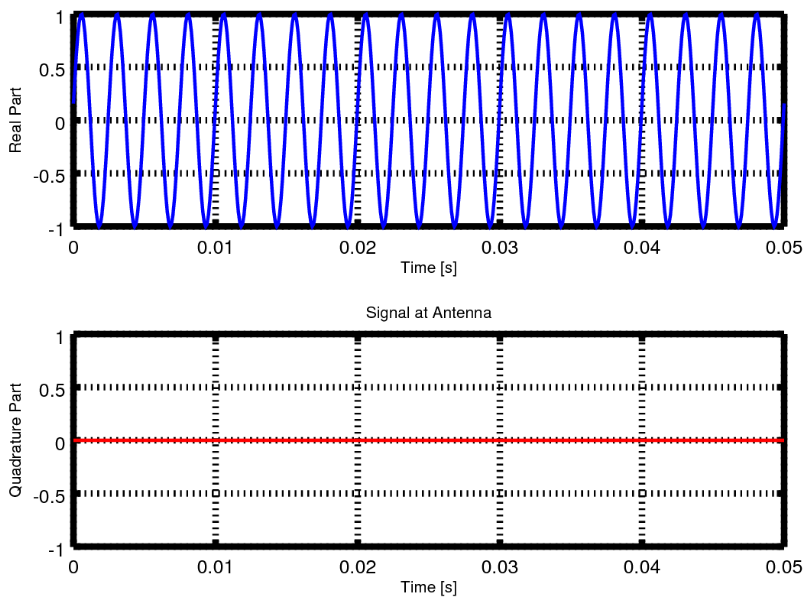
1.8.2 Hilbert Tranfsorm Defined in Terms of the Fourier Transform

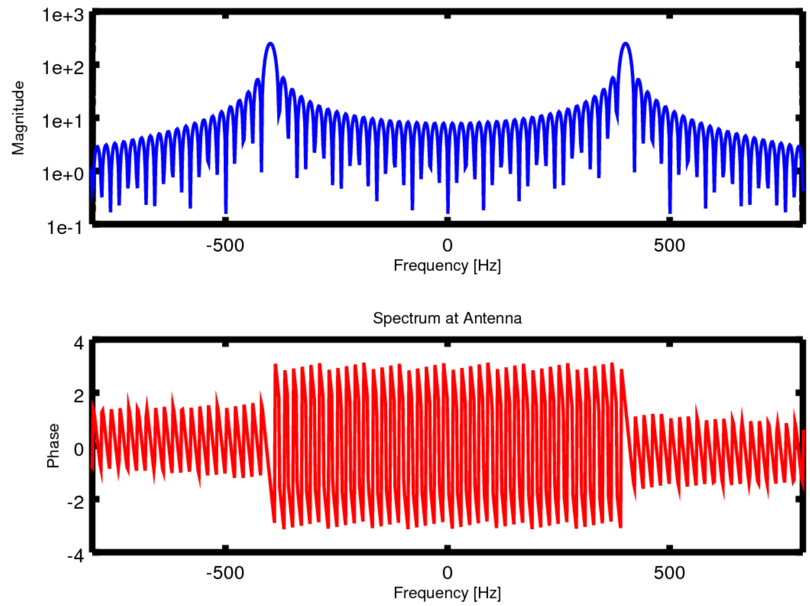
2 Receiver Antenna

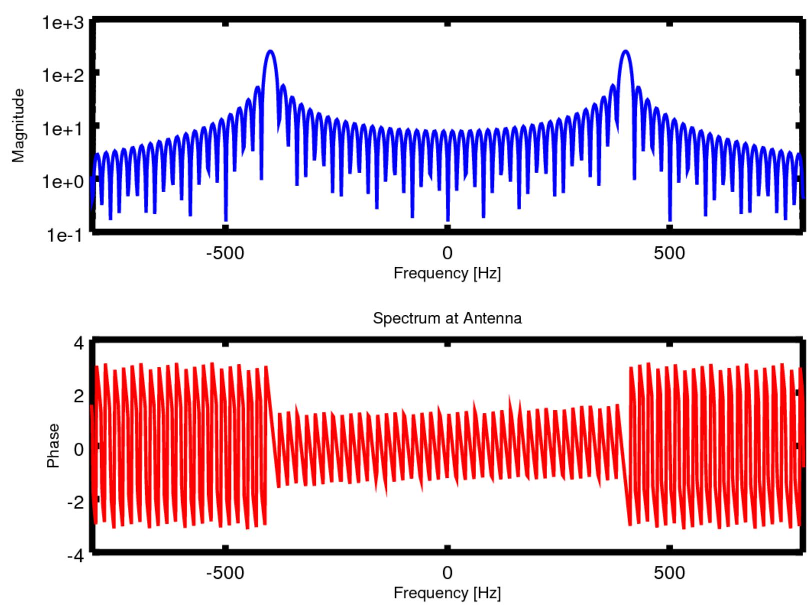
$$x_{+}\left(t\right) = \cos\left(2\pi f_k + \phi\right) \tag{16}$$

$$x_{-}\left(t\right) = \cos\left(-2\pi f_k + \phi\right) \tag{17}$$


Real Part
Time [s]
Signal at Antenna
Quadrature Part
Time [s]


Real Part
Time [s]
Signal at Antenna
Quadrature Part
Time [s]


Magnitude
1e+3
1e+2
1e+1
1e+0
1e-1
-500
0
500
Frequency [Hz]
Spectrum at Antenna
Phase
4
2
0
-2
-4
-500
0
500
Frequency [Hz]


Magnitude
1e+3
1e+2
1e+1
1e+0
1e-1
-500
0
500
Frequency [Hz]
Spectrum at Antenna
Phase
4
2
0
-2
-4
-500
0
500
Frequency [Hz]

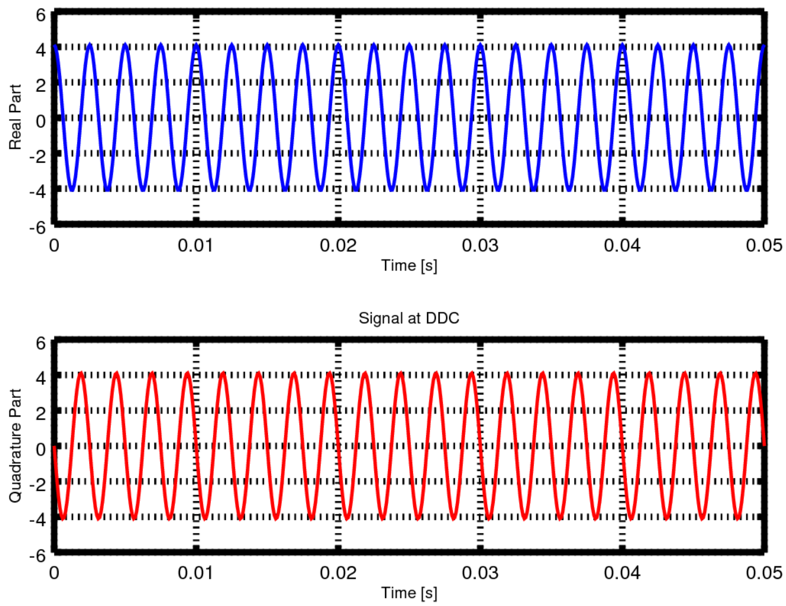
```
#!/usr/bin/octave -q
%% rcv_antenna.m
%%
%% Displays the time domain signal as seen by the antenna.

  function rcv_antenna(fs, t, fc, A, fk, phi, tag)

    x = A*cos(2*pi*fk*t + phi); % pure real signal

    %% time domain plot
    ax = figure(301);
      subplot(2,1,1);
        plot(t, real(x), ...
          'linewidth', 2, 'color', 'b');
        ylabel('Real Part');
        xlabel('Time [s]');
        grid on;
        set(gca, 'linewidth', 4, 'fontsize', 12)
      subplot(2,1,2);
        plot(t, imag(x), ...
          'linewidth', 2, 'color', 'r');
        title('Signal at Antenna');
        ylabel('Quadrature Part');
        xlabel('Time [s]');
        grid on;
        set(gca, 'linewidth', 4, 'fontsize', 12)
      set(ax, 'visible', 'off');
      saveas(ax, sprintf('../figures/%s_%s.png', mfilename, tag));

    %% Fast Fourier Transform
    N = 10 * 2^nextpow2(numel(x));
   %x_f = fftshift(fft(x, N))/N;
    x_f = fftshift(fft(x, N));
    if mod(N,2), k=-N/2:N/2-1; else k=-(N-1)/2:(N-1)/2; end
    T = N/fs;
    n_f = k/T;

    %% frequency domain plot
    ax = figure(302);
      xrange = sort( (2*fk*[-1 +1]) );
      subplot(2,1,1);
        semilogy(n_f, abs(x_f), ...
          'linewidth', 2, 'color', 'b');
        ylabel('Magnitude');
        xlabel('Frequency [Hz]');
```

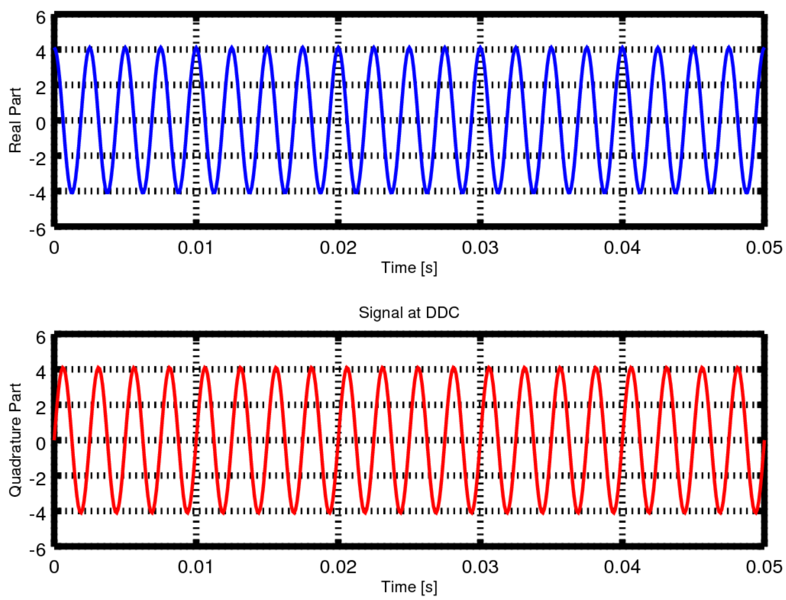
```
      xlim(xrange);
      set(gca, 'linewidth', 4, 'fontsize', 12)
    subplot(2,1,2);
      plot(n_f, angle(x_f), ...
        'linewidth', 2, 'color', 'r');
      title('Spectrum at Antenna');
      ylabel('Phase');
      xlabel('Frequency [Hz]');
      xlim(xrange);
      set(gca, 'linewidth', 4, 'fontsize', 12)
    set(ax, 'visible', 'off');
    saveas(ax, sprintf('../figures/%s_freq_%s.png', mfilename, tag));

  end % rcv_antenna

 %o = parameters();
 %rcv_antenna(...);

%% *EOF*
```

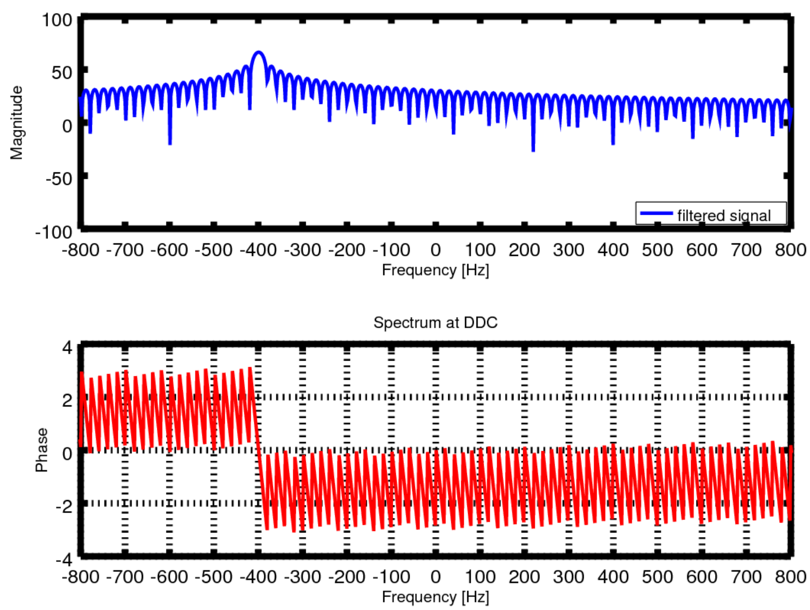
3 Analytic Signal

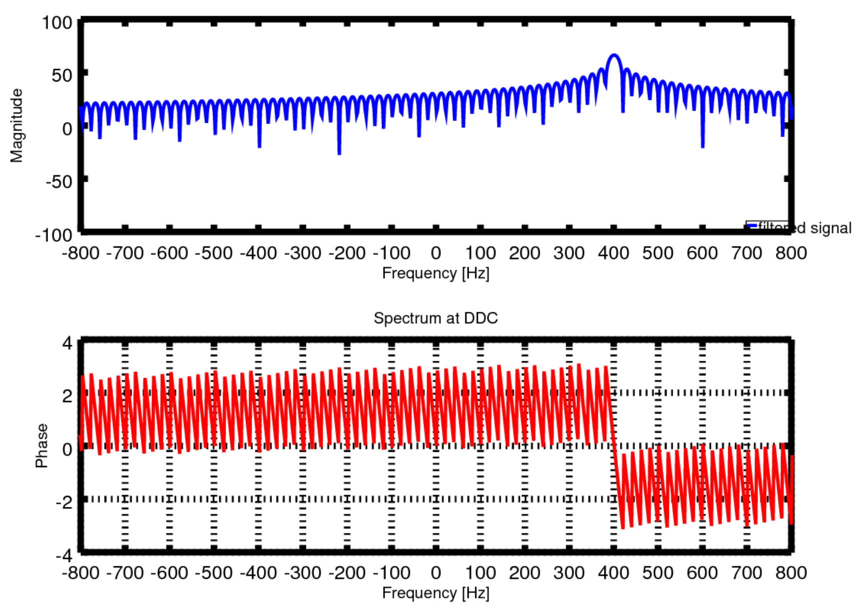
$$\begin{aligned} x_+(t) &= e^{+i2\pi f_k t+\phi} \\ &= \cos(2\pi f_k + \phi) + i\sin(2\pi f_k + \phi) \end{aligned} \tag{18}$$

$$\begin{aligned} x_-(t) &= e^{-i2\pi f_k t+\phi} \\ &= \cos(-2\pi f_k + \phi) + i\sin(-2\pi f_k + \phi) \\ &= \cos(2\pi f_k + \phi) - i\sin(2\pi f_k + \phi) \end{aligned} \tag{19}$$


Real Part
Time [s]
Signal at DDC
Quadrature Part
Time [s]
6
4
2
0
-2
-4
-6
0
0.01
0.02
0.03
0.04
0.05


Real Part
Time [s]
Signal at DDC
Quadrature Part
Time [s]
6
4
2
0
-2
-4
-6
0
0.01
0.02
0.03
0.04
0.05

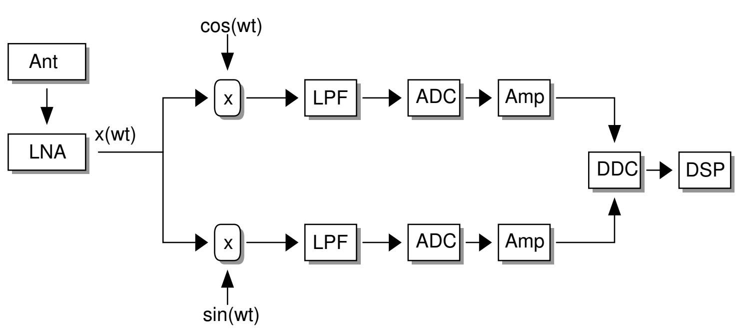

Magnitude
filtered signal
Frequency [Hz]
Spectrum at DDC
Phase
Frequency [Hz]

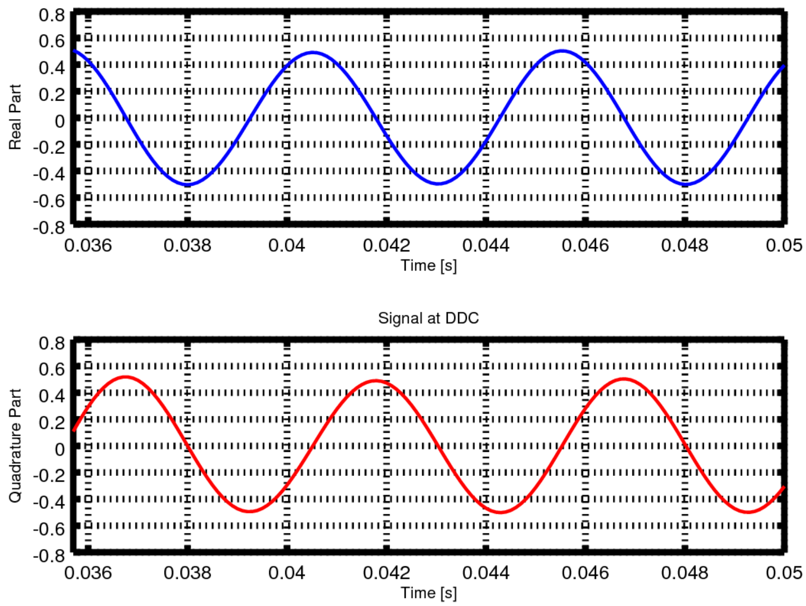

Magnitude
filtered signal
Frequency [Hz]
Spectrum at DDC
Phase
Frequency [Hz]

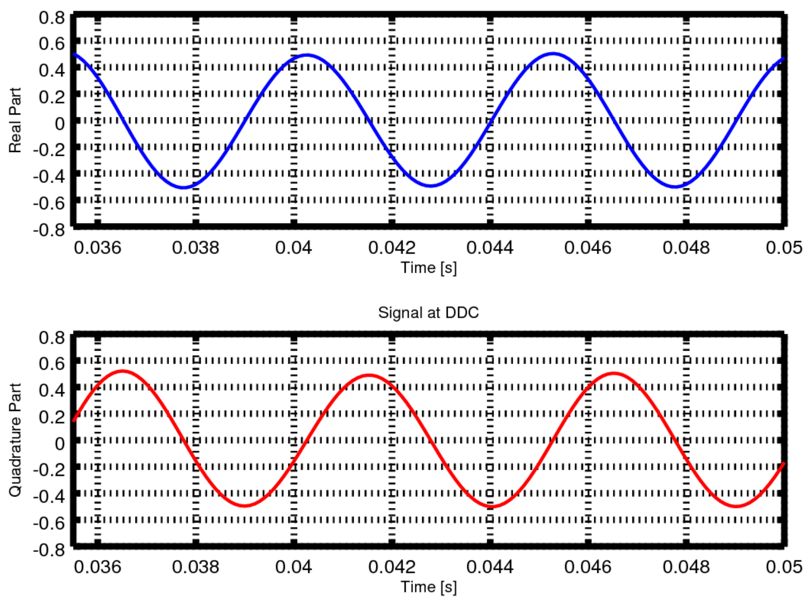
```
#!/usr/bin/octave -q
%% rcv_ddc.m
%%
%% Displays the time domain signal as seen by the DDC.

  function rcv_analytic(fs, t, fc, fi, A, fk, phi, tag)

    %% Receiver Channels
    x_a = A*exp(-1i*2*pi*fk*t + phi);     % analytic signal

    %% time domain plot
    ax = figure(321);
      subplot(2,1,1);
        plot(t, real(x_a), 'linewidth', 2, 'color', 'b');
        ylabel('Real Part');
        xlabel('Time [s]');
        grid on;
        xlim([t(1) t(end)]);
        set(gca, 'linewidth', 4, 'fontsize', 12)
      subplot(2,1,2);
        plot(t, imag(x_a), 'linewidth', 2, 'color', 'r');
        title('Signal at DDC');
        ylabel('Quadrature Part');
        xlabel('Time [s]');
        grid on;
        xlim([t(1) t(end)]);
        set(gca, 'linewidth', 4, 'fontsize', 12)
      set(ax, 'visible', 'off');
      saveas(ax, sprintf('../figures/%s_%s.png', mfilename, tag));

    %% Fast Fourier Transform
    N = 10 * 2^nextpow2(numel(x_a));
    x_f = fftshift(fft(x_a, N));
    if mod(N,2), k=-N/2:N/2-1; else k=-(N-1)/2:(N-1)/2; end
    T = N/fs;
    n_f = k/T;

    %% frequency domain plot
    ax = figure(322);
      xrange = sort( (2*fk*[-1 +1]) );
      subplot(2,1,1);
        plot(n_f, 20*log10(abs(x_f)), ...
          'linewidth', 2, 'color', 'b');
        ylabel('Magnitude');
        xlabel('Frequency [Hz]');
```

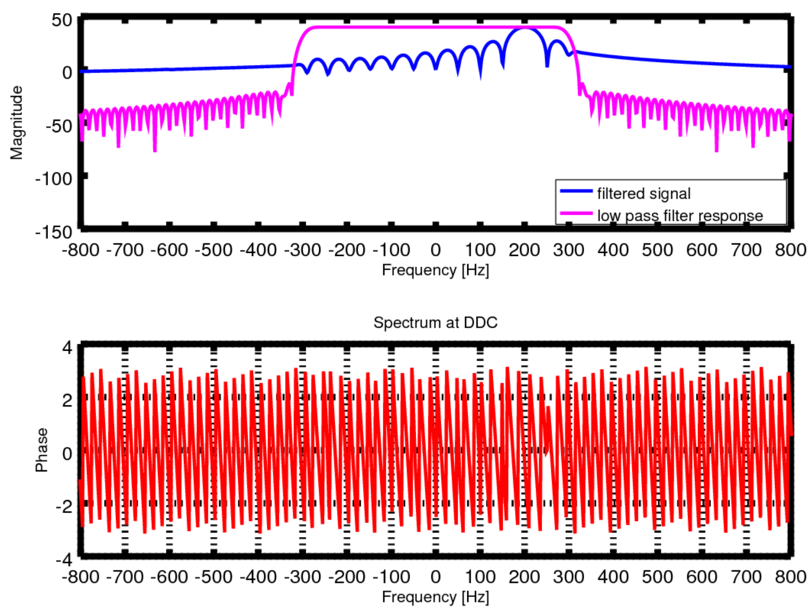
```
        xlim(xrange);
       %grid on;
        set(gca,'XTick',xrange(1):100:xrange(end));
        legend({'filtered signal', 'low pass filter response'}, ...
          'location', 'southeast');
        set(gca, 'linewidth', 4, 'fontsize', 12)
      subplot(2,1,2);
        plot(n_f, angle(x_f), 'linewidth', 2, 'color', 'r');
        title('Spectrum at DDC');
        ylabel('Phase');
        xlabel('Frequency [Hz]');
        xlim(xrange);
        grid on;
        set(gca,'XTick',xrange(1):100:xrange(end));
        set(gca, 'linewidth', 4, 'fontsize', 12)
      set(ax, 'visible', 'off');
      saveas(ax, sprintf('../figures/%s_freq_%s.png', mfilename, tag));

  end % rcv_analytic

 %o = parameters();
 %rcv_analytic(...);

%% *EOF*
```

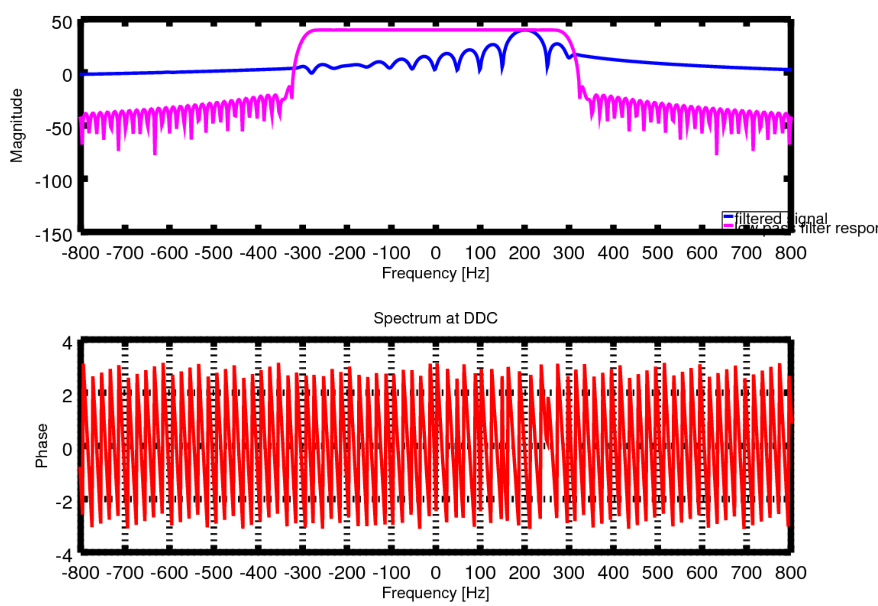
4 Quadrature Sampling




Real Part
Time [s]
Signal at DDC
Quadrature Part
Time [s]


Real Part
Time [s]
Signal at DDC
Quadrature Part
Time [s]


Magnitude
filtered signal
low pass filter response
Frequency [Hz]
Spectrum at DDC
Phase
Frequency [Hz]


Magnitude
Frequency [Hz]
Spectrum at DDC
Phase
Frequency [Hz]

```
#!/usr/bin/octave -q
%% rcv_ddc.m
%%
%% Displays the time domain signal as seen by the DDC.

  function rcv_ddc(fs, t, fc, fi, A, fk, phi, tag)

    %% Receiver Channels
    x        = A*sin(2*pi*fk*t + phi);         % signal at antenna
    fd       = -1 * abs(fc - fi);              % mixer frequency
    x_i      = x .* cos(2*pi*fd*t);            % real channel
    x_q      = x .* sin(2*pi*fd*t);            % quadrature channel

    %% Low-Pass Filter
    fcut     = 3.0 * fi;                       % [Hz    ] cutoff frequency
    wcut     = fcut/(fs/2);                    % [f norm] cutoff frequency (normalized)
    attn     = 40;                             % [dB    ] attenuation
    ord      = (attn*fs)/(22*(1.1*fcut-fcut)); % [#     ] FIR filter order
    if mod(ceil(ord),2)
      ord = ceil(ord) + 1;
    else
      ord = ceil(ord);
    end
    hb       = fir1(ord, wcut, 'low');
    ha       = [1];
    x_i_lpf  = filter(hb, ha, x_i);    % real channel low pass filter
    x_q_lpf  = filter(hb, ha, x_q);    % quadrature channel low pass filter

    %% Analytic Signal
    x_a_full = x_i     + 1i*x_q;
    x_a      = x_i_lpf + 1i*x_q_lpf;   % analytic signal

    %% time-alignment to account for filter response delay
    t0       = 0.03;                   % [s     ] filter response time
    ks       = find(t>t0);
    ks       = ks(1) + find(real(x_a(ks)) == max(real(x_a(ks))));  % cosine peak
    t0       = t(ks(1));               % snap to cosine peak
    ks       = find(t>t0);

    %% time domain plot
    ax = figure(311);
      subplot(2,1,1);
        plot(t(ks), real(x_a(ks)), 'linewidth', 2, 'color', 'b');
        ylabel('Real Part');
        xlabel('Time [s]');
```

```
        grid on;
        xlim([t(ks(1)) t(ks(end))]);
        set(gca, 'linewidth', 4, 'fontsize', 12)
      subplot(2,1,2);
        plot(t(ks), imag(x_a(ks)), 'linewidth', 2, 'color', 'r');
        title('Signal at DDC');
        ylabel('Quadrature Part');
        xlabel('Time [s]');
        grid on;
        xlim([t(ks(1)) t(ks(end))]);
        set(gca, 'linewidth', 4, 'fontsize', 12)
      set(ax, 'visible', 'off');
      saveas(ax, sprintf('../figures/%s_%s.png', mfilename, tag));

    %% Fast Fourier Transform
    N = 10 * 2^nextpow2(numel(x_a));
    x_f = fftshift(fft(x_a, N));
    if mod(N,2), k=-N/2:N/2-1; else k=-(N-1)/2:(N-1)/2; end
    T = N/fs;
    n_f = k/T;

    x_f_full = fftshift(fft(x_a_full, N));
    x_f_max  = max(abs(x_f));

    x_lpf = abs(fftshift(fft(hb,N)));
    scale = x_f_max / max(x_lpf);
    x_lpf = x_lpf * scale;
    f_lpf = (-0.5:1/N:0.5-1/N)*fs;

    %% frequency domain plot
    ax = figure(312);
      xrange = sort( (2*fk*[-1 +1]) );
      subplot(2,1,1);
       %plot(n_f, 20*log10(abs(x_f_full)), 'linewidth', 2, 'color', 'k-');
       %hold on;
        plot(n_f, 20*log10(abs(x_f)), ...
          'linewidth', 2, 'color', 'b');
        hold on;
        plot(f_lpf, 20*log10(x_lpf), ...
          'linewidth', 2, 'color', 'm', 'linestyle', '-');
        ylabel('Magnitude');
        xlabel('Frequency [Hz]');
        xlim(xrange);
       %grid on;
        set(gca,'XTick',xrange(1):100:xrange(end));
```

```
        legend({'filtered signal', 'low pass filter response'}, ...
          'location', 'southeast');
        set(gca, 'linewidth', 4, 'fontsize', 12)
      subplot(2,1,2);
        plot(n_f, angle(x_f), 'linewidth', 2, 'color', 'r');
        title('Spectrum at DDC');
        ylabel('Phase');
        xlabel('Frequency [Hz]');
        xlim(xrange);
        grid on;
        set(gca,'XTick',xrange(1):100:xrange(end));
        set(gca, 'linewidth', 4, 'fontsize', 12)
      set(ax, 'visible', 'off');
      saveas(ax, sprintf('../figures/%s_freq_%s.png', mfilename, tag));

    %% Low-Pass Filter response
    frange = 3.0*fi * [-1 +1];
    ax = figure(313);
      subplot(1,1,1);
        plot((-0.5:1/N:0.5-1/N)*fs,20*log10(abs(fftshift(fft(hb,N)))), ...
          'linewidth', 2, 'color', 'r');
        xlim([frange(1) frange(end)]);
        set(gca, 'linewidth', 4, 'fontsize', 12)
      set(ax, 'visible', 'off');
      saveas(ax, sprintf('../figures/%s_lpf_%s.png', mfilename, tag));

  end % rcv_ddc

 %o = parameters();
 %rcv_ddc(...);

%% *EOF*
```

5 Demo

```
#!/usr/bin/octave -q
%% run_demo.m
%%
%% Runs the complete demo for all sections.

  function my_demo(o)

```

```
    %% positive frequency (+f_k)
    tag = 'pos';
    fk = +1 * o.fk;
    rcv_antenna( o.fs, o.t, o.fc,       o.A, fk, o.phi, tag);
    rcv_ddc(     o.fs, o.t, o.fc, o.fi, o.A, fk, o.phi, tag);
    rcv_analytic(o.fs, o.t, o.fc, o.fi, o.A, fk, o.phi, tag);

    %% negative frequency (-f_k)
    tag = 'neg';
    fk = -1 * o.fk;
    rcv_antenna( o.fs, o.t, o.fc,       o.A, fk, o.phi, tag);
    rcv_ddc(     o.fs, o.t, o.fc, o.fi, o.A, fk, o.phi, tag);
    rcv_analytic(o.fs, o.t, o.fc, o.fi, o.A, fk, o.phi, tag);

  end % my_demo

  pkg load signal;
  o = parameters();
  my_demo(o);

%% *EOF*
```