APPENDIX B: MATLAB Code

B.1 Example 1: PRN Correlation Function

The MATLAB code reported below has been used to generate Figures 21.3 and 21.4.

```
% --------------------------------------------------------------------
% Example 1: Correlation between two spreading codes
% --------------------------------------------------------------------

clear all; close all; clc

Rc = 0.5e6;                         % Code Rate [chip/s]
Fs = 8e6;                           % Sampling Frequency [Hz]

c_loc = [1 -1 1 -1 -1 -1 -1 1 -1 ...
         1 -1 -1 -1 1 -1 -1 1 1 1 -1]; % PRN code

% sample the spreading codes
L = length(c_loc);                  % Code Length [chip]
N = floor(Fs*L/Rc);                 % Code Length [samples]

% ------------------------ Sample the local code ------------------------
k = 0:N-1;
c_loc_sampled = c_loc(floor(k*Rc/Fs)+1); % c_loc code sampled @ Fs
% --------------------------------------------------------------------

% - Generate the incoming code with 3 periods of c_loc, sample and shift --
c_in = [c_loc c_loc c_loc]; k = 0:3*N-1;
c_in_sampled = c_in(floor(k*Rc/Fs)+1); % c_in code sampled @ Fs

D = 4*Fs/Rc;                        % Code Delay [samples]

% samples of the incoming code with a code-phase shift
c_in_sampled_shift = circshift(c_in_sampled,[0 D]);
% --------------------------------------------------------------------

% correlate the two sequences of samples
for index=0:N-1,
   % correlate the codes...
   Corr(index+1)
= c_in_sampled_shift(1+index:N+index)*c_loc_sampled(1:N)';
end

% plot correlation functions
x_axis = [0:(N-1)]./Fs.*Rc;        % Prepare x-axis [chip]

figure(1),plot(x_axis,Corr,'.-k'), grid on;
xlabel('Delay [chip]')
```

```
ylabel('Correlation')
title('PRN code correlation')
```

B.2 Example 2: PRN Correlation Function in Presence of Noise

The MATLAB code reported below has been used to generate Figures 21.5 and 21.6.

```
% -------------------------------------------------------------------------
% Example 2: Correlation between two spreading codes with noise
% -------------------------------------------------------------------------

clear all; close all; clc

Rc = 0.5e6;                       % Code Rate [chip/s]
Fs = 8e6;                         % Sampling Frequency [Hz]

c_loc = [1 -1 1 -1 -1 -1 -1 1 -1 ...
        1 -1 -1 -1 1 -1 -1 1 1 1 -1]; % PRN code

% sample the spreading codes
L = length(c_loc);                % Code Length [chip]
N = floor(Fs*L/Rc);               % Code Length [samples]

% ------------------------ Sample the local code ------------------------
k = 0:N-1;

c_loc_sampled = c_loc(floor(k*Rc/Fs)+1); % c_loc code sampled @ Fs
% -------------------------------------------------------------------------

% - Generate the incoming code with 3 periods of c_loc, sample and shift --
c_in = [c_loc c_loc c_loc]; k = 0:3*N-1;
c_in_sampled = c_in(floor(k*Rc/Fs)+1); % c_in code sampled @ Fs

D = 4*Fs/Rc;                      % Code Delay [samples]

% samples of the incoming code with a code-phase shift
c_in_sampled_shift = circshift(c_in_sampled,[0 D]);

% add noise
sigma = 12;
c_in_sampled_noise = c_in_sampled_shift + sigma*randn(1,3*N);
% -------------------------------------------------------------------------

% correlate the two sequences of samples
for index=0:N-1,
   % correlate the codes...
   Corr(index+1)
= c_in_sampled_noise(1+index:N+index)*c_loc_sampled(1:N)';
end
```

```
% plot correlation functions
x_axis = [0:(N-1)]./Fs.*Rc;        % Prepare x-axis [chip]

figure(1),plot(x_axis,Corr,'.-k'), grid on;
xlabel('Delay [chip]')
ylabel('Correlation')
title('PRN code correlation')
```

B.3 Example 3: Serial Search Acquisition Algorithm

The MATLAB code reported below implements a serial search acquisition algorithm. It can be adapted to work with real GNSS signals.

```
% -----------------------------------------------------------------------
% Example 3: Serial search acquisition algorithm.
% -----------------------------------------------------------------------

clear all; close all; clc

Rc = 0.5e6;                        % Code Rate [chip/s]

Fs = 8e6;                          % Sampling Frequency [Hz]
Fc = 2e6;                          % Carrier Frequency [Hz]

c_loc = [1 -1 1 -1 -1 -1 -1 1 -1 ...
         1 -1 -1 -1 1 -1 -1 1 1 1 -1]; % PRN code

L = length(c_loc);                 % Code Length [chip]
N = floor(Fs*L/Rc);                % Code Length [samples]

% --------- Generate the incoming signal, with 3 periods of c_loc ---------
c_in = [c_loc c_loc c_loc];

% sample the spreading codes
k = 0:3*N-1;
c_in_sampled = c_in(floor(k*Rc/Fs)+1); % c_in code sampled @ Fs

D = 4*Fs/Rc;                       % code delay [samples]
sigma = 0.5;

% modulate the spreading code sampled @ Fs and add noise
c_in_sampled_noise = circshift(c_in_sampled,[0 D]).*cos(2*pi*Fc.*k/Fs)
...
+ (sigma)*randn(1,3*N);
% -----------------------------------------------------------------------

% ---------------------- Generate the local code ------------------------
k = 0:N-1; % samples vector
c_loc_sampled = c_loc(floor(k*Rc/Fs)+1); % c_loc code sampled @ Fs
% -----------------------------------------------------------------------
```

```
for idx_freq = 1:10,

   fd = Fc-50e3 + 10e3*idx_freq;      % local carrier frequency [Hz]
   loc_cos = cos(2*pi*fd.*k/Fs);      % In-phase local carrier
   loc_sin = sin(2*pi*fd.*k/Fs);      % Quadrature local carrier

   % prepare the local signal, multipling local code and carriers
   Loc_cos = c_loc_sampled.*loc_cos;
   Loc_sin = c_loc_sampled.*loc_sin;

for idx_shift = 0:N-1,
   % correlate incoming and local signals
   Corr(idx_freq,idx_shift+1) = ...
     (c_in_sampled_noise(1+idx_shift:N+idx_shift)*Loc_cos').^2 + ...
     (c_in_sampled_noise(1+idx_shift:N+idx_shift)*Loc_sin').^2;
end

end

% plot correlation functions
x_axis = [0:(N-1)]./Fs.*Rc;            % Prepare x-axis [chip]
y_axis = Fc-5e3 + 1e3.*[1:10]

figure(1), surf(x_axis,y_axis,abs(Corr)), shading interp,
xlabel('Delay [chip]')
ylabel('Freq [MHz]')
title('Cross Ambiguity Function')
```

B.4 Example 4: Acquisition Algorithms Based on FFT

The MATLAB code reported below has been used to generate Figures 21.13 and 21.14.

```
%-------------------------------------------------------------
% 17 November 2017
% FFT-based CAF evaluation
%-------------------------------------------------------------

clear all
close all
clc

%------------------ Load of gold code (GPS)

load('GPS_code.mat')
disp('Contents of workspace after loading file:')
whos

%------------------ parameters of the carrier
fd=-2750;                              % Doppler frequency
FIF=4e6;                               % Intermediate frequency
```

```
BW=2*FIF;
phi0=0;
FRF=1575.42e6;                      % Radio frequency (GPS L1)
%------------------- parameters of the PRN code
Nchip=1023;                         % number of chips
Tcode=1e-3;                         % Code period
Tchip_nom=Tcode/Nchip;
Tchip=Tchip_nom/(1+fd/FRF);         % chip period with Doppler effect

%------------------- parameters related to noise and SIS power
CN0dBHz=48;                         % carrier to noise ratio
CN0=10^(CN0dBHz/10);
SNR=CN0/BW;                         % Signal to noise ratio
sigmanoise=1;
sigmanoise2=sigmanoise^2;
ampSIS=sqrt(2)*sqrt(SNR*sigmanoise2);

%------------------- sampling frequency
fs=2.1*BW;      % sampling frequency
Ts=1/fs;

%------------------- simulation paramters
N_Tcode=3;
T_fin=Tchip*Nchip*N_Tcode;          % Duration of the simulated signal
time=0:Ts:T_fin;
Ntime=length(time);

%---------------------------------------------------------------
%---------------------------------------------------------------
%---------------------------------------------------------------
% Signal generation

disp('Signal generation')

% N_Tcode code periods are generated

code_rep=code1023;
for indc=1:N_Tcode-1
   code_rep=[code_rep, code1023];
end

time_floor=floor(time/Tchip);

icounter=1;
for index=1:Ntime-1
   codesignal(index)=code_rep(icounter);
   if time_floor(index+1) > time_floor(index)
     icounter=icounter+1;
   end
```

```
end
% codesignal is the PRN code (with N_Tcode code periods)

carrier=cos(2*pi*(FIF+fd)*time(1:end-1)+phi0);
SIS_clean=ampSIS*codesignal.*carrier;
SIS=SIS_clean+sigmanoise*randn(1,length(carrier));

% SIS is the received signal with noise (with no data)

% ------------ Extraction of a SIS segment of 1ms
t1num=300;
t1=t1num*Tchip; % Simulation of an arbitrary delay
t2=t1+Tcode;
delay=Tcode-t1;

N1=floor(t1/Ts);
N2=floor(t2/Ts);

SIS_seg=SIS(N1:N2);
NSIS=length(SIS_seg);
% SIS_seg is a segment of the received signal (1 code period)

%---------------------------------------------------------------
%---------------------------------------------------------------
%---------------------------------------------------------------
% CAF evaluation (FFT in the time domain)

disp('CAF evaluation (FFT in the time domain)')
fmax=5000; % maximum Doppler frequency
fdbar=-fmax:300:+fmax; % Frequency bins for CAF
LF=length(fdbar);
timeCAF_bin=0:Ts:Tcode; % Time bins for CAF

DFTc=conj(fft(codesignal(1:length(timeCAF_bin))));

for indf=1:LF
   eloc=exp(1i*2*pi*(FIF+fdbar(indf))*timeCAF_bin);
   qSe=SIS_seg.*eloc;
   DFTq=fft(qSe);
   CAF_in_TD(:,indf)=ifft(DFTc.*DFTq);
end
% CAF_in_TD is the CAF evaluated by FFT in the time domain

% --- CAF sections
[m1,asc1]=max(abs(CAF_in_TD));
[m2,asc2]=max(m1);
CAFtime=CAF_in_TD(:,asc2);
[m3,asc3]=max(CAFtime);
CAFfreq=CAF_in_TD(asc3,:);
```

```
%------------------------------------------------------------------
%------------------------------------------------------------------
%------------------------------------------------------------------

% CAF evaluation (FFT in the frequency domain)

disp('CAF evaluation (FFT in the frequency domain)')

% --------- Definition of the prefilter parameters
BW_FIR=4*fmax; % value arbitrarily chosen
T_FIR=1/BW_FIR;
N_dec=round(T_FIR/Ts);
bFIR=ones(1,N_dec)/N_dec;

% --------- Parameters for signal decimation
Num_col=floor(NSIS/N_dec);
Ntot=Num_col*N_dec;
ND=round(N_dec/2);

% --------- Mixer with complex exponential (for down-conversion)
SIS_with_mixer=SIS_seg.*exp(-1i*2*pi*FIF*time(1:length(SIS_seg)));
num_delay=length(SIS_seg);

for indd=1:num_delay
   code_del=codesignal(1+indd-1:length(timeCAF_bin)+indd-1);
   SIS_car=code_del.*SIS_with_mixer;
   SIS_car_FIR=filter(bFIR,1,SIS_car);
   % down-conversion
   SIS_car_FIR_matrix=reshape(SIS_car_FIR(1:Ntot),[N_dec,Num_col]);
   SIS_dec=SIS_car_FIR_matrix(ND,:);
   % decimation (digital sampling)
   LS=length(SIS_dec);
   CAF_in_FD(indd,:)=fft([SIS_dec zeros(1,LS)]);
   % zero padding arbitrarily chosen (can be modified)
end
% CAF_in_FD is the CAF evaluated by FFT in the frequency domain

%------------------------------------------------------------------
%------------------------------------------------------------------
%------------------------------------------------------------------
% Figures

% CAF in the time domain
mesh(fdbar,timeCAF_bin,abs(CAF_in_TD))
title('CAF in the time domain')
xlabel('Doppler frequency (Hz)')
ylabel('Code delay (s)')

set(gcf, 'PaperPosition', [0 0 18 12]);
set(gcf, 'PaperSize', [18 12]);
```

```
saveas(gcf, 'CAF_TD', 'fig') %Save figure
saveas(gcf, 'CAF_TD', 'eps') %Save figure

figure
subplot(211)
plot(timeCAF_bin,abs(CAFtime))
legend(['True delay =',num2str(delay),' s'])
grid on
subplot(212)
plot(fdbar,abs(CAFfreq))
legend(['True Doppler freq. =',num2str(fd),' Hz'])
grid on

% CAF in the frequency domain
figure
[nrow,ncol]=size(CAF_in_FD);
ncdiv2=round(ncol/2);
cc1=CAF_in_FD(:,1:ncdiv2);
cc2=CAF_in_FD(:,ncdiv2+1:end);
CAF_in_FD_sim=[cc2 cc1];          % matrix reordering

Tdur=Tcode+LS*Ts*N_dec;
delf_fft=1/Tdur;
CAFfreq_FD=0:1:ncol-1;
CAFfreq_FD=CAFfreq_FD-ncdiv2;
CAFfreq_FD=CAFfreq_FD*delf_fft;

CAF_in_FD_sim_ud=flipud(CAF_in_FD_sim); % matrix reordering
mesh(CAFfreq_FD,timeCAF_bin,abs(CAF_in_FD_sim_ud))
title('CAF in the frequency domain')
xlabel('Doppler frequency (Hz)')
ylabel('Code delay (s)')

set(gcf, 'PaperPosition', [0 0 18 12]);
set(gcf, 'PaperSize', [18 12]);
saveas(gcf, 'CAF_FD', 'fig') %Save figure
saveas(gcf, 'CAF_FD', 'eps') %Save figure

[m1,asc1]=max(abs(CAF_in_FD_sim_ud));
[m2,asc2]=max(m1);
CAFtime=CAF_in_FD_sim_ud(:,asc2);
[m3,asc3]=max(CAFtime);
CAFfreq=CAF_in_FD_sim_ud(asc3,:);

figure
subplot(211)

plot(timeCAF_bin,abs(CAFtime))
legend(['True delay =',num2str(delay),' s'])
```

```
grid on
subplot(212)
plot(CAFfreq_FD,abs(CAFfreq))
legend(['True Doppler freq. =',num2str(fd),' Hz'])
grid on
```