# Appendix A

## A. Simulation documentation

Chapter 7 presented a practical problem where timing measurement improvements were needed. A solution was developed using all the methods presented in this thesis: M/N detection, integration, threshold/correlation. The solution was implemented on a PC computer with a 800 MHz processor. The code was developed under MATLAB. This appendix presents the program used for this simulation.

### *2.7 Main program*

The program needed for the experiment of Chapter 7 is composed of a main loop that organizes the different actions of the processing algorithms. A description of each step of the code is given in Section A.1.1.1. This description is then followed by the code itself, which is provided in Section A.1.1.2.

#### Program description

After defining the parameters of the simulation: the values of M and N in the M/N detection, the sampling frequencies, the length of the time window and the true time delay between the pre-recorded pulses, a first loop dedicated to the SNRs that are studied is started. This loop is followed by a loop controlling the number of repetitions of the experiment.

The received signals are then generated using a function called "**createPulseNoiseMN**" which takes as an input the number of cycles needed for the M/N detection process: N, and the SNR that is simulated. The M/N detection process is performed on the generated signal with the function: "**mnProcess**". The output of this process: "detected pulses" is a X-by-2 matrix which columns give respectively the beginning and end of each of the pulses that are detected by the M/N detection process. There are as many lines in this matrix as there are pulses detected.

Since two pulses only need to be processed, the next step of the program is to choose two pulses out of the detected signal. The solution that is implemented consists in determining which of the peaks that crossed the threshold are the two longest ones. This choice is motivated by a probability consideration: a Gaussian pulse being of a fixed length related to its bandwidth, and the noise being at a bandwidth at least equal or higher than the pulse bandwidth yields the conclusion that noise pulses will have in average a smaller width than the Gaussian pulse, especially as the threshold level is high. This operation is implemented using the function "**longest2Chains**". Note however that another solution as mentioned in Chapter 8 would be to give more weight to detected pulses that were processed during the previous cycle of detection / integration / measurement, or close in time to the previous solutions so as to take account of the motion of pulses. The information concerning the samples that should be integrated at 5MHz based on the choice of these two pulses is obtained from the function "**determineIntegrationLimits**".

The selected limits of integration: "columnIntegration" are then used in the integration process in order to integrate the signal within these limits. Prior to this operation, Z cycles of signal are generated, corresponding to the number of cycles that are integrated: Z. This is performed using "**createPulseNoiseZ**". These cycles of signals are then integrated using the function "**integration**".

The last step is the timing measurement part. For the integration process, the center of each of the two pulses that are integrated within a time window: "matrixCenterThreshold" is returned by the function "**measurementThreshold**". This is converted into a time delay by function "**measureDelayThreshold**".

For the correlation process, the center of the correlation is provided by the function named: "**measurementThresholdCorrelation**", and this center: "matrixCenterCorrelation" is then converted into a time delay using the same function as the threshold case: "**measureDelayThreshold**".
The last operations that are performed by the program are the computation of the error of the measurement, and the storage of these values within a matrix.

## Program code

```
%*********************************************************************%
% Raphael Renault                                                     %
% thesis: Detection of fast moving pulses in a Noisy Environment      %
% 11/20/00                                                            %
% Main Loop for final experiment                                      %
%*********************************************************************%

clear all

%*********************************************************************%
% Parameter definition                                                %
%*********************************************************************%

% Parameters of M/N detection
samplingFrequencyMN = 2e6;
M = 6;
N = 9;

% Parameters of integration
samplingFrequencyZ = 5e6;
Z = 18;

% Parameter of the length of the time window | true time delay value
% 5ms is chosen, that is 25pts at 5MHz, or 12pts apart from the center
halfLengthTimeWindow = 12;
trueDelay = 3.24e-3;

% Preparation of the matrix of the results
matrixErrorThresh = [];
matrixErrorCorr = [];

% Definition of the number of repetitions of the simulation
repetition = 100;

% Beginning of the loop in SNR
SNdB = 0:2:20;

for index1 = 1:length(SNdB)
```

```
   SNR = SNdB(index1);

   matrixErrorThresh = [matrixErrorThresh ; SNR zeros(1,repetition)];
   matrixErrorCorr = [matrixErrorCorr ; SNR zeros(1,repetition)];

   % Beginning of the loop in repetition
   for index2 = 1:repetition

      errorThreshold = [];
      errorCorrelation = [];

%*********************************************************************%
% M/N process                                                         %
%*********************************************************************%

      numberDetectedPulses = 1;

      while numberDetectedPulses < 2

         % Create pulse embedded in noise for the M/N process
         signalReceived = createPulseNoiseMN( N , SNR );
         threshold = 0.5;
         detectedPulses = mnProcess( signalReceived , threshold , M );

         % Note: if the threshold needs to be adjusted, it should be
         % done here so that the number of detected pulses is at least
         % equal to 2

         numberDetectedPulses = size( detectedPulses , 1 );

      end;

%*********************************************************************%
% Determination of the time window location                           %
%*********************************************************************%

      % Find the 2 longest chains
      finaldetectedPulses = longest2Chains( detectedPulses );

      % Determine the integration limits within a cycle of signal
      columnIntegration = determineIntegrationLimits( ...
         finaldetectedPulses , halfLengthTimeWindow , ...
         ratioSampling , 25000);
      % 25000 is the length of a cycle at 5MHz.

%*********************************************************************%
% Integration Process                                                 %
%*********************************************************************%

      clear signalReceived;

      signalReceived = createPulseNoiseZ( columnIntegration , Z , SNR);

      matrixIntegration = ...
         integration( columnIntegration , signalReceived );

%*********************************************************************%
```

```
% Threshold measurement Process                                          %
%************************************************************************%

      % Note: the threshold level should be the one determined
      % in the MN process.
      matrixCenterThreshold = ...
         measurementThreshold( matrixIntegration , threshold );

      timeDelayThreshold = ...
         measureDelayThreshold( columnIntegration , ...
         matrixCenterThreshold , samplingFrequencyZ );

%************************************************************************%
% Correlation measurement Process
%************************************************************************%

      matrixCenterCorrelation = ...
         measurementThresholdCorrelation( matrixIntegration );

      timeDelayCorrelation = ...
         measureDelayThreshold( columnIntegration , ...
         matrixCenterCorrelation , samplingFrequencyZ );

%************************************************************************%
% Comparison of the rms error for each method                            %
%************************************************************************%

      errorThreshold = timeDelayThreshold - trueDelay;
      errorCorrelation = timeDelayCorrelation - trueDelay;

      matrixErrorThresh( end , index2+1 ) = errorThreshold;
      matrixErrorCorr( end , index2+1 ) = errorCorrelation;

   end; % Loop repetition

end; % Loop SNR
```

## 2.8 M/N detection process

### Program description

The M/N detection is performed by creating a replica of the matrix of the signal of equivalent size: “`replica`”, except that the value at each point is a digitized version using a 2-bit quantization process. If a point of the matrix is above the threshold, a 1 is stored; otherwise a 0 is stored. The N lines of this matrix are then summed. The points for which more than M threshold crossings were recorded “`aboveThreshold2`” are detected pulses.

"**indexation**" is a function that is called in order to take all the columns that correspond to a detected point, and create a matrix composed of the beginning and end of all the detected pulses on each of its lines. This is performed by taking the vector corresponding the columns of the pulses that were detected, and subtracting its replica shifted by one value. The values that are different than one indicate a sequence break, and therefore a new pulse, whose beginning and end is stored into the output matrix.

## Program code

```
%*****************************************************************%
% Raphael Renault                                                 %
% thesis: Detection of fast moving pulses in a Noisy Environment  %
% 11/20/00                                                        %
% Perform the MN process                                          %
%*****************************************************************%

function [outputMN] = mnProcess( signal , threshold , M )

% Create the matrix with ones at each threshold crossings
replica = zeros( size(Signal) );
for i = 1:size( Signal , 1 )
   aboveThreshold1 = find( signal(i,:) >= threshold );
   replica( i , aboveThreshold1 ) = 1;
end;

% Sum the number of threshold crossings
replica = sum(replica,1);

% Find different pulses detected
aboveThreshold2 = find( replica >= M );

% Take all the pulses that are detected, and determine the beginning
% and end of each of them
outputMN = indexation( aboveThreshold2 );

return


%*****************************************************************%
% Raphael Renault
% thesis: Detection of fast moving pulses in a Noisy Environment  %
% 11/20/00                                                        %
% Fast Indexation of the sequences within a matrix                %
%*****************************************************************%

function [matrixResult] = indexation(matrix)

% determination of the sequence breaks within the matrix,
```

```
% corresponding to a new detected pulse.
matrix1 = matrix;
matrix2 = [matrix(2:end) matrix(end)];
replica = matrix2 - matrix1;

% computation of the beginning and end of each of the sequence
matrixResult = [matrix1(1) 0];
for index = 1:size(matrix1,2)-1
   if replica( index ) ~= 1;
      matrixResult(end,2) = matrix1(index);
      matrixResult = [ matrixResult ; matrix1(index+1) 0 ];
   end;
end;

matrixResult(end,end) = matrix1(end);

return
```

## 2.9 Determination of the integration boundaries

### Program description

Two steps are necessary to determine the limits for the integration process based on the detected pulses. The first one is to determine the longest two chains that were detected, which is implemented in the function: "**`longest2Chains`**". This function subtracts the beginning and end of each detected pulse in order to determine its length. The maximum length obtained is stored in "`index`", the associated value is fixed to –1 in the original matrix "`matrixLength`" in order not to take account of this value again, and the second maximum length is tracked. The beginning and end of these two longest chains are again put into a matrix that constitutes the output of "**`longest2Chains`**".

The result of the above function is processed in "**`determineIntegrationLimits`**", which computes the center of each pulse associated to the 5MHz sampling scale. That practically means that the computed center of each pulse "`centerColumn`" is rounded to the nearest value in a 5MHz sampling signal scale. A time window is then centered on this value. Since the center that is determined may be close to the boundaries of a signal cycle, the function "**`checkProblem`**" is called to correct an out of limit time window.

## Program code

```
%****************************************************************%
% Raphael Renault                                                %
% thesis: Detection of fast moving pulses in a Noisy Environment  %
% 11/20/00                                                       %
% Determine the longest 2 chains of a matrix                     %
%****************************************************************%

function [matrixResults] = longest2Chains(matrix)

matrix1 = matrix;
matrixLength = matrix1(:,2) - matrix1(:,1);

max = 0;
indexMax = 0;
index = [];
for i = 1:2
   [max,indexMax] = max(matrixLength);
   index = [ index indexMax ];
   matrixLength(index) = -1;
end;

results = [ matrix1(index(1),:) ; matrix1(index(2),:) ];

if results(1,1) < results(2,1)
   matrixResults = results;
else
   matrixResults = [ results(2,:) ; results(1,:) ];
end;

return


%****************************************************************%
% Raphael Renault                                                %
% thesis: Detection of fast moving pulses in a Noisy Environment  %
% 11/20/00                                                       %
% Determine the integration boundaries                           %
%****************************************************************%

function [matrix] = determineIntegrationLimits( pulsesLocation , ...
   lengthWindow , ratioSampling , numberPointsProcessed );

% Determine the center of each pulse within the 5MHz scale
centerColumn = mean(pulses,2) .* ratioSampling;
centerColumn = round(centerColumn);

% Determine the columns of the time window at 5MHz
matrix = centerColumn * ones(1,2) + lengthWindow .* ...
   [ -ones(2,1) ones(2,1) ];

% Check if detected pulses are out of boundaries of the cycle length
matrix = checkProblem( columnIntegration , numberPointsProcessed );

return
```

```
%*****************************************************************%
% Raphael Renault                                                 %
% thesis: Detection of fast moving pulses in a Noisy Environment  %
% 11/20/00                                                        %
% Check out of boundaries integration                             %
%*****************************************************************%

function [matrixResults] = checkProblem( matrix , limit )

matrixResults = matrix;

if matrixResults(1,1) < 1
   matrixResults(1,2) = 1 - matrixResults(1,1) + matrixResults(1,2);
   matrixResults(1,1) = 1;
end;

if matrixResults(2,1) < 1
   matrixResults(2,2) = 1 - matrixResults(2,1) + matrixResults(2,2);
   matrixResults(2,1) = 1;
end;

if matrixResults(1,2) > limit
   matrixResults(1,1) = limit - matrixResults(1,2) +
matrixResults(1,1);
   matrixResults(1,2) = limit;
end;

if matrixResults(2,2) > limit
   matrixResults(2,1) = limit - matrixResults(2,2) +
matrixResults(2,1);
   matrixResults(2,2) = limit;
end;

return
```

## 2.10 Integration process

### Program description

Integration is a straightforward process that consists in taking the matrix of the signal, and summing altogether the lines of this matrix, each line representing a cycle of signal. The result is then divided by the number of lines that were summed in order to obtain the mean of the result.

### Program code

```
%*************************************************************%
% Raphael Renault                                             %
% thesis: Detection of fast moving pulses in a Noisy Environment  %
% 11/20/00                                                    %
% Integration process                                         %
%*************************************************************%

function [matrixResults]=integration(columns,signal)

matrixResults = [ mean( signal( : , columns(1,1):columns(1,2)) , 1 )...
      ; mean( signal( : , columns(2,1):columns(2,2)) , 1 ) ];

return
```

## *2.11 Threshold timing measurements*

### Program description

The threshold process is performed by using the function “find{x >= y}” of MATLAB that returns the columns of a matrix that satisfy the condition x>=y. Consequently, by using this function on the integrated signal values as compared to a threshold level, and taking the mean of the first and last column that are obtained: “`aboveThreshold1(1)`” and “`aboveThreshold1(end)`”, the centers of the pulses are computed: “`center1`” and “`center2`”. However, if no signal crosses the threshold, the threshold level is decreased, and the process repeated. “**`measureDelayThreshold`**” is then used to determine what the time delay between the two measured center is, taking into account the time separation between the time windows of each pulse.

### Program code

```
%*************************************************************%
% Raphael Renault                                             %
% thesis: Detection of fast moving pulses in a Noisy Environment  %
% 11/20/00                                                    %
% Threshold process                                           %
```

```
%*****************************************************************%

function [matrixResult] = measurementThreshold(signal,threshold)

threshold1 = threshold;
threshold2 = threshold;

aboveThreshold1 = [];
while isempty(aboveThreshold1)
   aboveThreshold1 = find(signal(1,:) >= threshold1);
   threshold1 = threshold1*(0.9);
end;

aboveThreshold2 = [];
while isempty(aboveThreshold2)
   aboveThreshold2 = find(signal(2,:) >= threshold2);
   threshold2 = threshold2*(0.9);
end;

center1 = ( aboveThreshold1(1) + aboveThreshold1(end) ) / 2;
center2 = ( aboveThreshold2(1) + aboveThreshold2(end) ) / 2;

matrixResult = [center1 center2];

return


%*****************************************************************%
% Raphael Renault                                                 %
% thesis: Detection of fast moving pulses in a Noisy Environment  %
% 11/20/00                                                        %
% determine the final time delay for threshold                    %
%*****************************************************************%

function [result]= measureDelayThreshold( window , centers , ...
   samplingFrequency )

result = ( centers(2) - centers(1) ) + ( window(2,1) - window(1,1) );

result = result / samplingFrequency;

return
```

## 2.12 Correlation timing measurements

### Program description

The correlation operation is performed using the MATLAB function “`xcorr`”. The center of the obtained signal within the time window is then calculated using the same technique

as for the threshold detection: the values above threshold are stored in "`aboveThreshold`"; the first and last value of this vector are summed and divided by two to give the center "`center2`". "`center2`" however, is taken as the length of the signal that is studied since this value corresponds to the center of the autocorrelation of the pulse, which constitutes the reference for the calculated error. This point is a specificity of MATLAB since the length of the autocorrelation of a pulse of length N is a signal of length 2×N+1, which maximum is obtained at column N.

Finally, the corresponding time delay between the two pulses is computed by the same function "**`measureDelayThreshold`**" used in the previous section.

## Program code

```
%*************************************************************%
% Raphael Renault%
% thesis : Detection of fast moving pulses in a Noisy Environment %
% 11/20/00                                                    %
% Determine the center of correlation                         %
%*************************************************************%

function [matrixResult] = measurementThresholdCorrelation(signal)

correlationSignal = xcorr( signal(1,:) , signal(2,:) );
maxCorrelationSignal = max(correlationSignal);
threshold = 0.5;

aboveThreshold = [];
while isempty(aboveThreshold)
   aboveThreshold = find( correlationSignal >= ...
      threshold * maxCorrelationSignal );
   threshold = threshold * (0.9);
end;

center1 = size(signal,2);
center2 = ( aboveThreshold(1) + aboveThreshold(end) ) / 2;

matrixResult = [ center1 center2 ];

return
```