FFI-rapport 2013/00048

Pulse Descriptor Word (PDW) simulator – Version 1.0

Gudrun Høye and Eirik Jensen Opland

FFI Forsvarets forskningsinstitutt

FFI-rapport 2013/00048

Pulse Descriptor Word (PDW) simulator – Version 1.0

Gudrun Høye and Eirik Jensen Opland

Norwegian Defence Research Establishment (FFI)

12 April 2013

FFI-rapport 2013/00048

1219

P: ISBN 978-82-464-2246-6
E: ISBN 978-82-464-2247-3

Keywords

Simulator

PDW

Radar

ESM

Pulssortering

Approved by

Berit Jahnsen	Project Manager
Anders Eggen	Director

English summary

When developing pulse handling algorithms, a developer may face various problems with respect to the use of real data to test his or her algorithms. Some of these problems can be solved by using simulated data instead (or in addition).

We have developed a first version of a Pulse Descriptor Word (PDW) simulator with a user-friendly interface in Matlab. The simulator produces PDWs with information about time of arrival (TOA), amplitude, frequency, pulse width, and angle of arrival (AOA), based on information about scene geometry and radar and sensor properties.

Applying the PDW simulator to a test scenario has shown that the simulator produces quite realistic output. The PDW simulator was also used to test a deinterleaving algorithm – the LINE Deinterleaver. The benefits of controlling the input scenario, as well as knowing the absolute truth about which pulse belongs to which radar, was clearly seen for this type of application. We conclude that the PDW simulator could be a valuable additional tool when developing and testing pulse handling algorithms.

The current version of the PDW simulator software simulates rotating radars in 2 dimensions on a flat Earth. Future versions of the software could be expanded to operate in 3 dimensions and to use an elliptical Earth. Obstacles could be added to the scenario, and integration of the software with 3D maps could open up for the possibility to create realistic geographic scenarios. Non-rotating radars and additional parameters and measurements such as crystal frequency and Doppler could also be included. The user interface could be upgraded to further facilitate the use of the PDW simulator.

Sammendrag

I arbeidet med å utvikle algoritmer som skal håndtere pulser, vil en programutvikler møte ulike problemstillinger med hensyn på det å bruke reelle data til å teste algoritmene sine. Noen av disse problemene kan løses ved å bruke simulerte data i stedet (eventuelt i tillegg).

Vi har utviklet en første versjon av en Pulse Descriptor Word (PDW)-simulator med et brukervennlig brukergrensesnitt i Matlab. Simulatoren genererer PDWer som inneholder informasjon om ankomsttid (TOA), amplitude, frekvens, pulsbredde og ankomstvinkel, basert på informasjon om radarenes og sensorenes egenskaper samt geometrien i scenen.

Bruk av PDW-simulatoren på et test-scenario har vist at simulatoren kan generere nokså realistiske PDWer. Simulatoren har også blitt brukt til å teste en pulssorteringsalgoritme – LINE pulssorterer. For denne typen applikasjon var det åpenbare fordeler med å ha kontroll på inputscenarioet, samt å kjenne ”fasiten” på hvilken puls som hører til hvilken radar. Vi konkluderer med at PDW-simulatoren vil kunne være et verdifullt tilleggsverktøy ved utvikling og testing av algoritmer som håndterer pulser.

Dagens versjon av PDW-simulator programvaren simulerer roterende radarer i 2 dimensjoner på flat jord. I fremtidige versjoner av programvaren kunne man tenke seg muligheten av å utvide simulatoren til å operere i 3 dimensjoner samt benytte elliptisk jord i beregningene. Hindringer (objekter som helt eller delvis blokkerer utsendte radarpulser) kunne inkluderes i scenarioet, og ved å integrere programvaren med 3D-kart ville man få muligheten til å skape realistiske geografiske scenarioer. Man kunne også inkludere ikke-roterende radarer samt nye parametere og målinger slik som krystallfrekvens og Doppler i programvaren. Brukergrensesnittet kunne oppgraderes ytterligere slik at PDW-simulatoren blir enda mer brukervennlig.

Contents

1 Introduction

When developing pulse handling algorithms, a developer faces several difficulties with respect to the testing of his or her algorithms:

1) It may be difficult to get access to real data to use for the testing.
2) Real data are often “messy”, i.e., it may be difficult to obtain “clean” data for testing of specific parts of the algorithm.
3) It may be difficult to get hold of the “right type” of data for specific testing purposes.
4) It may be difficult to get data from dense enough pulse environments to test the algorithm’s performance in challenging conditions.
5) Real data are often classified and may therefore be less convenient to work with, or even be inaccessible.
6) For real data the “truth” is not known and it may be difficult to evaluate exactly how well the algorithm is performing.

A Pulse Descriptor Word (PDW) simulator could potentially solve many of the problems described above. The user would be in full control of the scenario to investigate, and the simulator could produce both very “clean” data as well as simulate extremely dense pulse environments. It would be possible to examine and evaluate the performance of the algorithm in question precisely, since it would be known which pulses belong to which radar. Of course, in the end pulse handling algorithms must always be tested on real data, since a simulator will not be able to fully recreate all aspects of a real scenario. However, a PDW simulator could be a valuable additional tool when developing and testing new pulse handling algorithms.

We have developed a first version of such a PDW simulator, and its capabilities and use will be described in the following chapters. Chapter 2 gives detailed information about what the simulator can do, Chapter 3 describes how to use the software, and Chapter 4 demonstrates how to apply the PDW simulator to a test scenario and discusses the resulting output. In Chapter 5 an example of an application is given, where the PDW simulator is used to test the LINE Deinterleaver. A summary is given in Chapter 6, while the algorithm for the PDW simulator software is given in Appendix A.

2 What can the simulator do?

The PDW simulator generates simulated PDWs for a given input scenario that contains one or more rotating radars and one or more sensors that receive the emitted radar pulses. The sensors “measure” the time of arrival (TOA), amplitude, frequency, angle of arrival (AOA), and pulse width (PW) of the received pulses. Figure 2.1 shows a simplified block diagram for the PDW simulator.

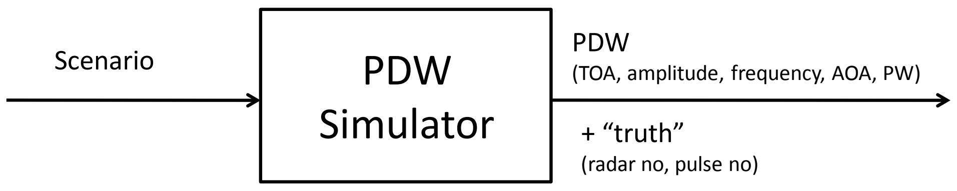


Figure 2.1 Simplified block diagram for the PDW simulator indicating the input and the output from the simulator.

The PDW simulator keeps track of which radar a given pulse on a given sensor comes from. This information is for instance useful if the simulator is used to evaluate the performance of algorithms/programs for deinterleaving. In this case it is easy to identify if the simulated pulses (PDWs) have been sorted correctly or not by the deinterleaver. We will see an example of this in Chapter 5, where the LINE deinterleaver is applied to pulses generated by the PDW simulator.

In addition, the simulator numbers all pulses from a given radar so that it is possible to identify if a pulse received by two or more sensors is the same pulse. This information is useful for instance if the PDW simulator is used to evaluate programs or methods that associate measurements from two or more sensors.

The input scenario must contain information about the geometry (radar and sensor positions and movements), radar properties (such as rotation period, lobe pattern, PRI, frequency, and pulse width), and sensor properties (ability to detect pulses and to perform precise measurements of various pulse parameters). This will be described in more detail below.

2.1 Scenario geometry

The scenario geometry is defined by the sensor and radar positions and movements. The current version of the PDW simulator operates in 2 dimensions (the horizontal plane) and assumes that the Earth is flat, i.e., uses Cartesian coordinates. Figure 2.2 shows an example of a scenario geometry.

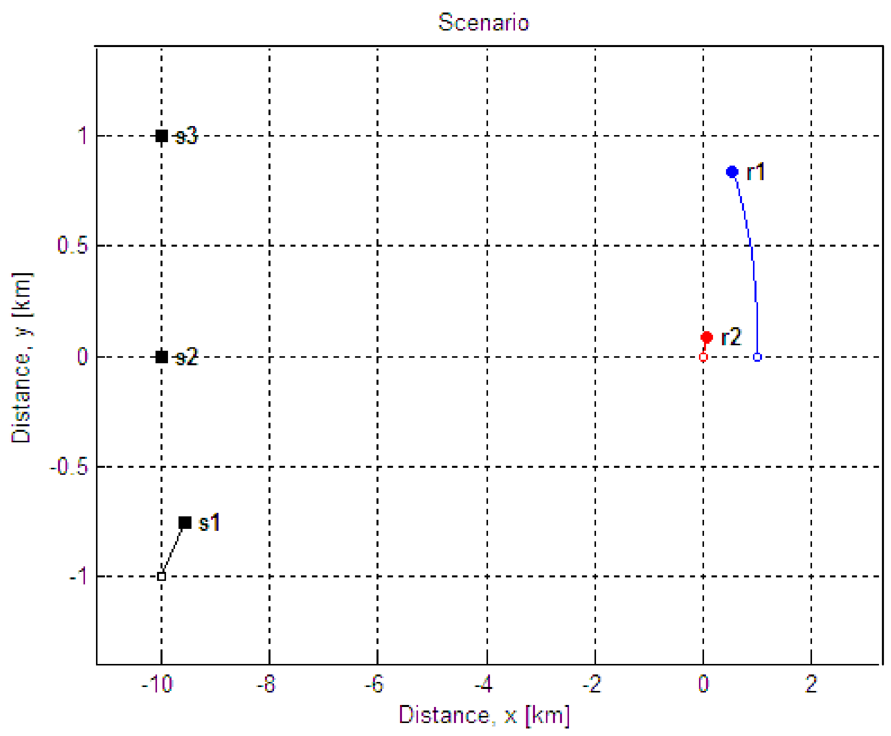


Figure 2.2 Example of scenario geometry. This scenario contains three sensors and two radars. Two of the sensors (s2 and s3) are stationary, while the third sensor (s1) moves along a straight line. One of the radars (r2) also moves along a straight line, while the second radar (r1) moves along a circle arc.

The PDW simulator offers four different types of movement for the sensors and radars:

1) Stationary
2) Movement along a straight line with constant speed
3) Movement along a circle arc with constant speed
4) Arbitrary movement

Movement types 2) - 4) will be described in more detail below.

2.1.1 Movement along a straight line with constant speed

Figure 2.3 shows an example of movement along a straight line with constant speed. The radar/sensor position (*x*, *y*) at time *t* is given by

$$\begin{aligned} x &= v_x \cdot (t - t_0) + x_0 \\ y &= v_y \cdot (t - t_0) + y_0 \end{aligned} \tag{2.1}$$

where (x_0, y_0) is the start position, t_0 is the start time and v_x and v_y are the speeds along the *x*- and *y*-axis respectively, as given by

$$v_x = v \cdot \cos\alpha$$
$$v_y = v \cdot \sin\alpha \tag{2.2}$$

where *v* is the (constant) radar/sensor speed along the trajectory and *α* is the angle of the trajectory relative the *x*-axis.

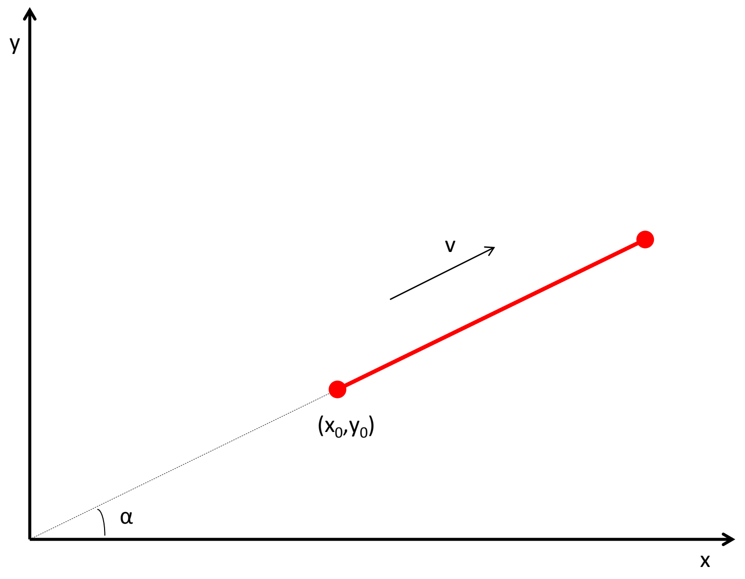


Figure 2.3 Movement along a straight line with constant speed.

2.1.2 Movement along a circle arc with constant speed

Figure 2.4 shows an example of movement along a circle arc with constant speed. The radar/sensor position (*x*, *y*) at time *t* is given by:

$$x = r \cdot \cos[\alpha(t)] + x_0$$
$$y = r \cdot \sin[\alpha(t)] + y_0 \tag{2.3}$$

where *r* is the radius of the circle, (x_0, y_0) is the position of the center of the circle, and *α(t)* is the angle relative the *x*-axis given by

$$\alpha(t) = \frac{v}{r}(t - t_0) + \alpha_0 \tag{2.4}$$

where *v* is the (constant) speed of the radar/sensor along the trajectory, t_0 is the start time, and α_0 is the start angle relative the *x*-axis for the trajectory.

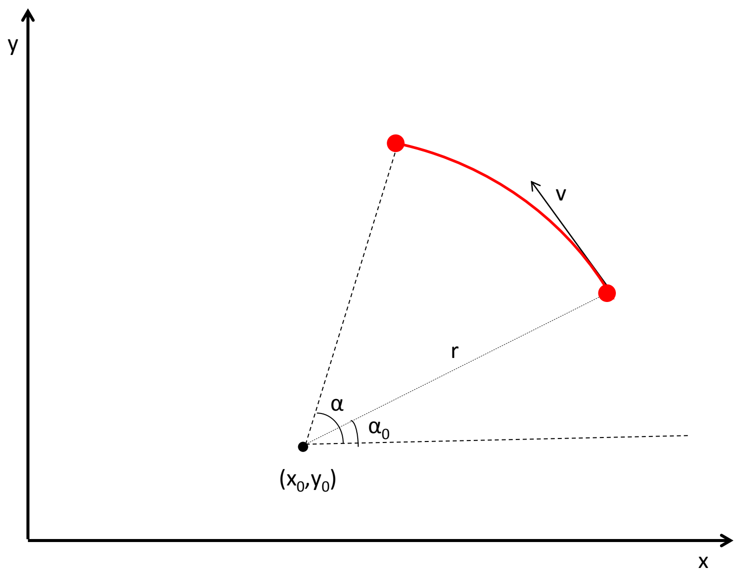


Figure 2.4 Movement along a circle arc with constant speed.

2.1.3 Arbitrary movement

Arbitrary movement is defined by pairs of corresponding time t and position (x, y) that are listed in an input file. Intermediate positions are found by linear interpolation. Any type of movement for the sensors and radars can be described by this input file.

2.2 Radar properties

Several properties of the radar are included in the PDW simulator:

1) Rotation period
2) Radar antenna lobe pattern
3) PRI
4) Frequency
5) Pulse width

Each of these is described in more detail below.

2.2.1 Rotation period

The radar rotation period determines the angle α between the radar main lobe pointing direction and the x-axis at a given time, see Figure 2.5.

The PDW simulator provides two choices for the radar rotation period:

1) Constant
2) Varying

Each option will be described in more detail below.

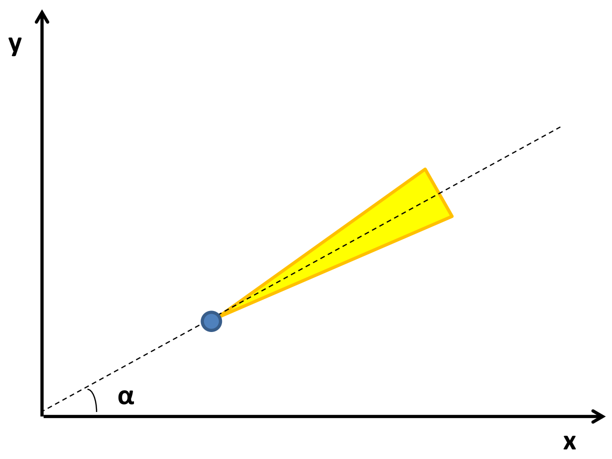


Figure 2.5 Angle between the radar main lobe pointing direction and the x-axis.

2.2.1.1 Constant rotation period

For a constant rotation period T_{rot} the angle α between the radar main lobe pointing direction and the x-axis at time t is given by

$$\alpha(t) = \frac{2\pi}{T_{rot}} \cdot (t - t_0) + \alpha_0 \tag{2.5}$$

where α_0 is the angle between the radar main lobe pointing direction and the x-axis at time t_0.

2.2.1.2 Varying rotation period

The variation in the rotation period is set to correspond to a sinusoidal variation in the angular speed. This angular speed ω at time t can be expressed as

$$\omega(t) = \omega_0 \cdot \left(1 + A \cdot \sin\left(s\omega_0 t + \varphi_0\right)\right) \tag{2.6}$$

where A is the amplitude of the variation relative to the constant component of the angular speed, s is the angular frequency of the variation relative to the constant component of the angular speed, φ_0 is the start phase of the variation, and ω_0 is the constant component of the angular speed given by

$$\omega_0 = \frac{2\pi}{T_{rot}} \tag{2.7}$$

where T_{rot} is the constant component of the rotation period. The corresponding varying rotation period $T(t)$ can be written

$$T(t) = \frac{2\pi}{\omega(t)} = T_{rot} \cdot \left(\frac{1}{1 + A \cdot \sin\left(s \frac{2\pi}{T_{rot}} t + \varphi_0 \right)} \right) \tag{2.8}$$

The angle *α* between the radar main lobe pointing direction and the *x*-axis at time *t* is then given by

$$\alpha(t) = \int_{t_0}^{t} \omega(t) dt + \alpha_0 = \omega_0 \cdot t - A \cdot s^{-1} \cdot \cos\left(s\omega_0 t + \varphi_0 \right) + B \tag{2.9}$$

where t_0 is the start time, α_0 is the angle between the radar main lobe pointing direction and the *x*-axis at time t_0, and *B* is a constant given by

$$B = A \cdot s^{-1} \cdot \cos\left(s\omega_0 t_0 + \varphi_0 \right) - \omega_0 t_0 + \alpha_0 \tag{2.10}$$

2.2.2 Radar antenna lobe pattern

The PDW simulator provides two choices for the radar antenna lobe pattern:

1) Sinc-function shaped
2) Arbitrarily shaped

Each option will be described in more detail below.

2.2.2.1 Sinc-function shaped radar antenna lobe pattern

The PDW simulator uses a modified sinc-function as standard radar antenna lobe pattern. The amplitude P_θ [dB] at given angle θ is given by

$$P_\theta = 20 \cdot \lg\left(\frac{\sin(\pi x)}{\pi x} \right) + P_{ML} \tag{2.11}$$

in the range $\theta = [-\pi/2\ \pi/2]$ where P_{ML} is the amplitude of the main lobe, and by

$$P_\theta = 20 \cdot \lg\left(\frac{\sin(\pi x)}{\pi x} \right) + P_{ML} + \frac{2}{\pi} \cdot P_{BL} \cdot \left(\theta - \frac{\pi}{2} \right) \tag{2.12}$$

when $\theta > \pi/2$, and by

$$P_\theta = 20 \cdot \lg\left(\frac{\sin(\pi x)}{\pi x} \right) + P_{ML} - \frac{2}{\pi} \cdot P_{BL} \cdot \left(\theta + \frac{\pi}{2} \right) \tag{2.13}$$

when $\theta < -\pi/2$. Here P_{BL} is the back lobe amplitude and the parameter x is calculated from

$$x = \frac{0.443 \cdot \sin(\theta)}{\sin(\theta_{ML}/2)} \tag{2.14}$$

where θ_{ML} is the main lobe opening angle.

Figure 2.6 shows an example of a radar antenna lobe pattern where the main lobe opening angle is 10º, the main lobe amplitude is 0 dB, and the back lobe amplitude is -20 dB.

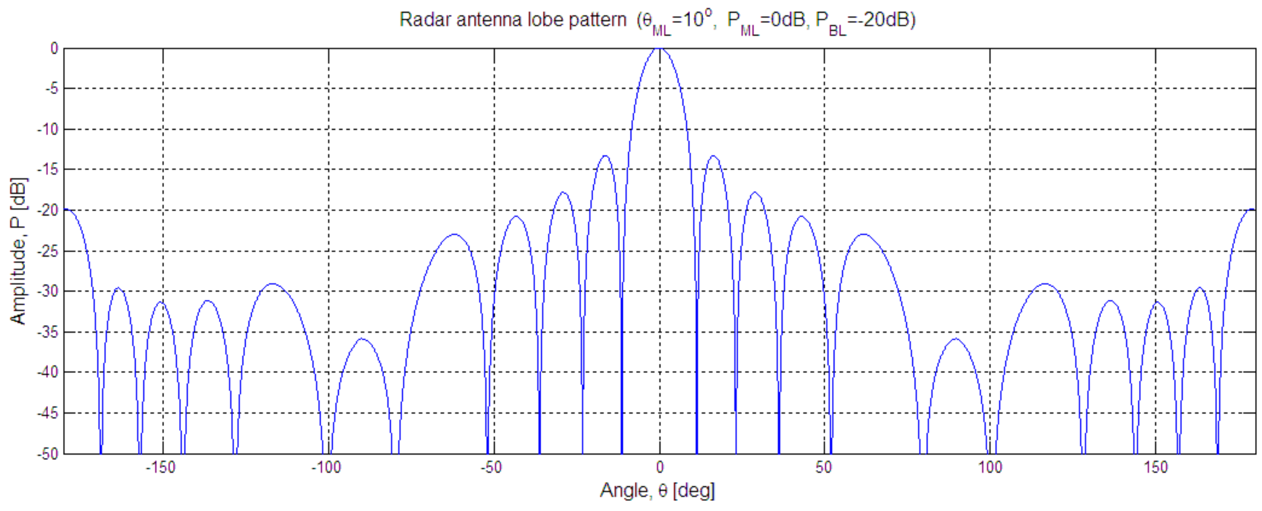


Figure 2.6 Sinc-function shaped lobe pattern with main lobe opening angle equal to 10º, main lobe amplitude equal to 0 dB, and back lobe amplitude equal to -20 dB.

2.2.2.2 Arbitrarily shaped radar antenna lobe pattern

The arbitrarily shaped lobe pattern is defined by pairs of corresponding angle θ and lobe pattern amplitude P_θ, that are listed in an input file. Intermediate values are found by linear interpolation. Any type of radar antenna lobe pattern can be described by this input file.

2.2.3 PRI

The PDW simulator provides the possibility to choose between four different PRI types:

1) Fixed
2) Stagger
3) Switched
4) Jitter

Each option will be described in more detail below.

2.2.3.1 Fixed PRI

The PRI is constant, see Figure 2.7 for an example.

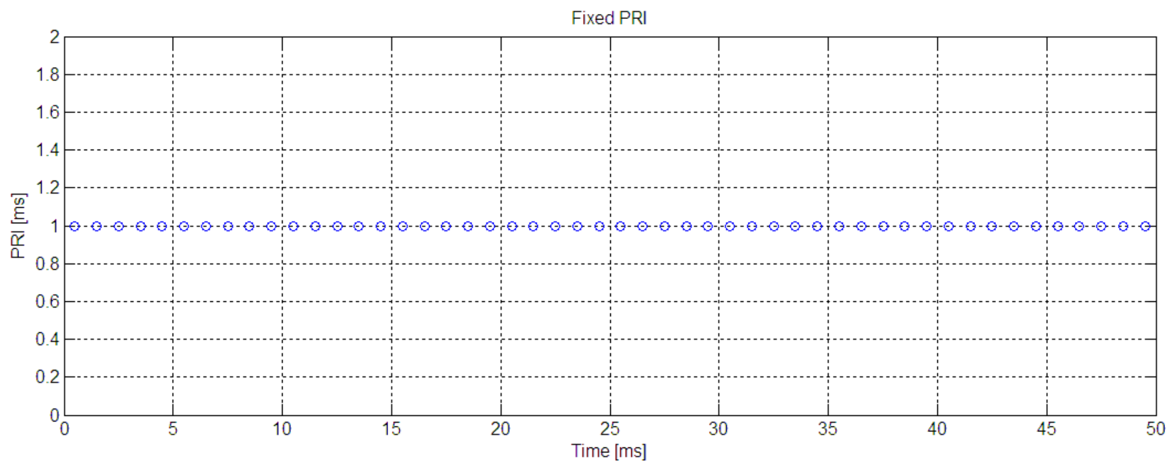


Figure 2.7 Example of fixed PRI, where the PRI is 1 ms.

2.2.3.2 Stagger PRI

The PRI varies according to a specified pattern. Figure 2.8 shows stagger PRI with repeated blocks of the PRI values [0.8 0.9 1.0 1.1 1.2] ms.

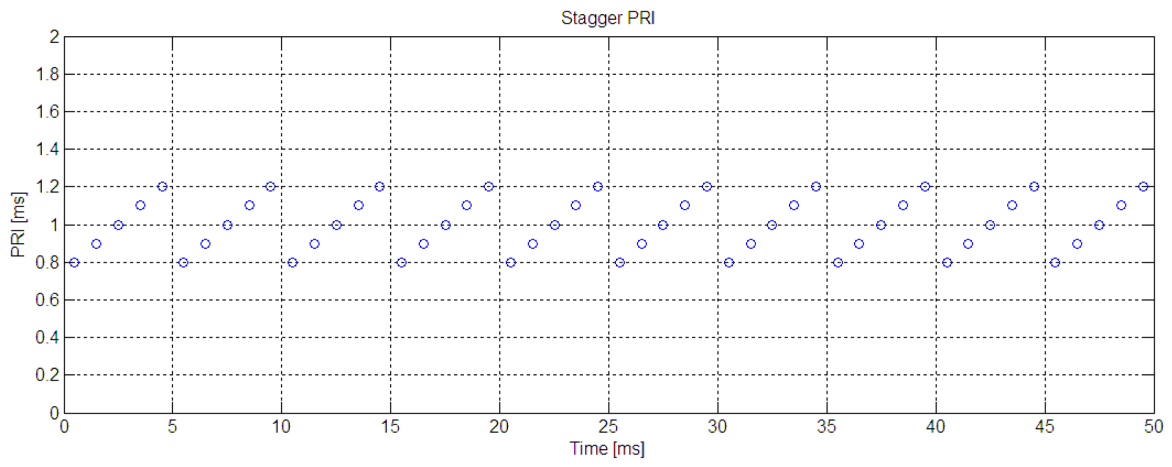


Figure 2.8 Example of stagger PRI, where the PRI pattern is constructed from repeated blocks of the PRI values [0.8 0.9 1.0 1.1 1.2] ms.

2.2.3.3 Switched PRI

The PRI varies according to a specified pattern, where each PRI value may be repeated several times. Figure 2.9 shows switched PRI with repeated blocks of the pattern [6x0.8 4x1.0 6x1.2] ms.

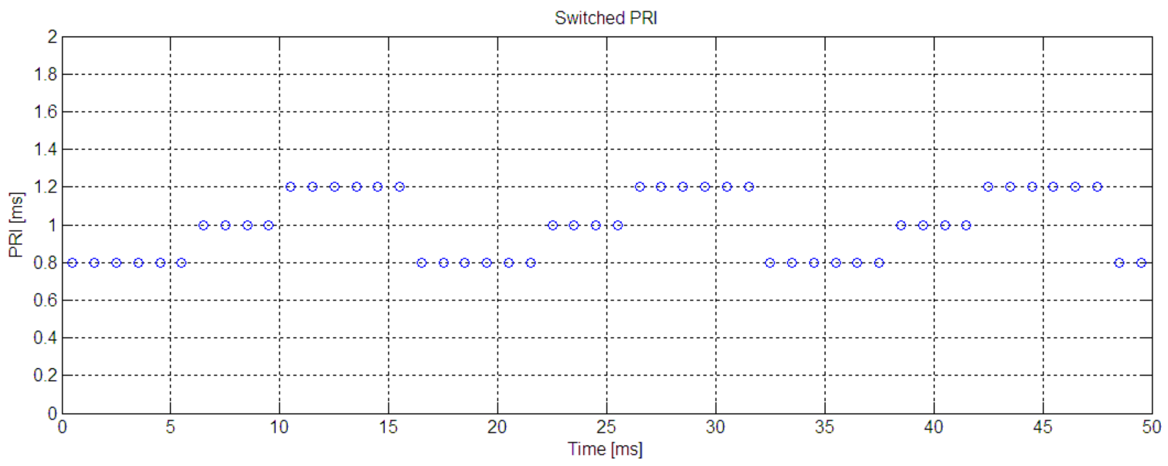


Figure 2.9 Example of swithced PRI, where the PRI pattern is constructed from repeated blocks of the PRI values [6x0.8 4x1.0 6x1.2] ms.

2.2.3.4 Jitter PRI

Jitter PRI has a Gaussian variation around its PRI mean value. Figure 2.10 shows jitter PRI with mean value 1 ms and standard deviation 0.01 ms (1%).

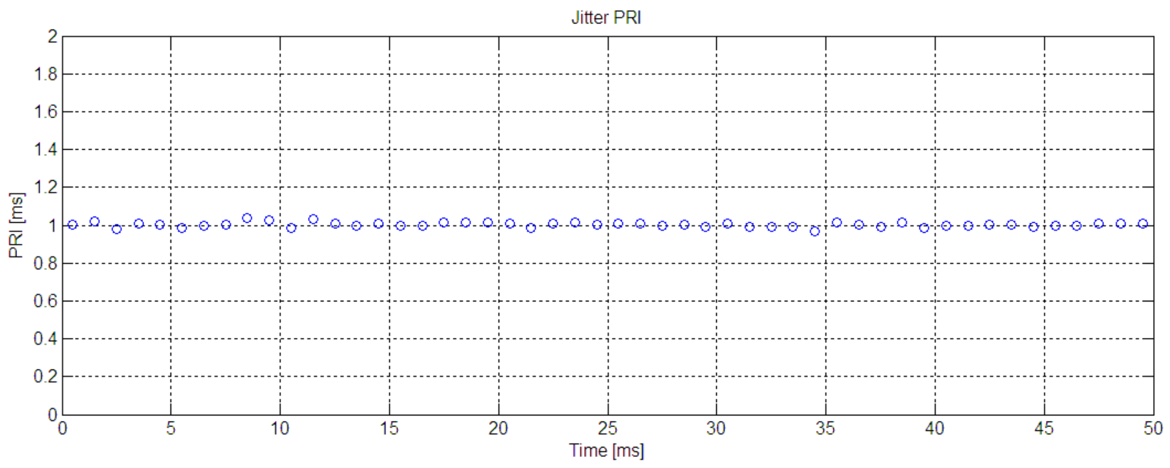


Figure 2.10 Example of jitter PRI with mean value 1 ms and standard devation 0.01 ms (1%).

2.2.4 Frequency

The PDW simulator provides the possibility to choose between four different types of frequency:

1) Fixed
2) Stagger
3) Switched
4) Jitter

This is similar to what was described above for the PRI, see Section 2.2.3.1-2.2.3.4.

2.2.5 Pulse width

The PDW simulator provides the possibility to choose between four different types of pulse width:

1) Fixed
2) Stagger
3) Switched
4) Jitter

This is similar to what was described above for the PRI, see Section 2.2.3.1-2.2.3.4.

2.3 Sensor properties

The sensor is characterized by its ability to detect pulses and to perform precise measurements of various pulse parameters such as amplitude, TOA, frequency, pulse width, and AOA.

2.3.1 Detection probability and saturation level

The probability that a sensor detects a given pulse is assumed to be determined only by the pulse amplitude. The PDW simulator allows setting several different amplitude levels with corresponding detection probabilities. See Section 3.1.5.3 for details.

The PDW simulator can also simulate that pulses are saturated. The saturation level for the pulse amplitude must then be given.

2.3.2 Amplitude measurements

The amplitude P (measured in dB) of a pulse received by the sensor is calculated from

$$P = P_0 + P_\theta + P_r + \Delta P_{syst} + \Delta P_{arb} \tag{2.15}$$

where P_0 is the amplitude of an emitted pulse from an equivalent omnidirectional radar antenna (as measured in the close vicinity of that antenna), P_θ is the amplitude correction due to the radar antenna lobe pattern as given by Equations (2.11)-(2.14), ΔP_{syst} and ΔP_{arb} are the systematic and arbitrary errors in the amplitude measurements, and P_r is the amplitude correction due to the distance r between the radar and the sensor as given by

$$P_r = -20\lg(r) \tag{2.16}$$

The arbitrary amplitude error ΔP_{arb} is modeled as being Gaussian, while the systematic amplitude error ΔP_{syst} is modeled as one of the following:

1) Constant
2) Linearly changing
3) Varying as a sinus

For option 2 (linearly changing) the following equation is used to calculate the systematic error:

$$\Delta P_{syst} = k \cdot (t - t_0) + \Delta P_0 \tag{2.17}$$

where k is the rate of change of the error and ΔP_0 is the systematic error at start time t_0.

For option 3 (sinus variations) the following equation is used to calculate the systematic error:

$$\Delta P_{syst} = A \cdot \sin(2\pi f \cdot t + \varphi_0) \tag{2.18}$$

where A is the error amplitude, f is the frequency of the variation, and φ_0 is the phase of the error variation at t=0.

2.3.3 TOA measurements

The time of arrival (TOA) of a pulse at the sensor is calculated from

$$TOA = TOA_0 + \Delta T_r + \Delta TOA_{syst} + \Delta TOA_{arb} \tag{2.19}$$

where TOA_0 is the time when the pulse was emitted from the radar and ΔT_r is the time it takes for the pulse to travel the distance r between the radar and the sensor

$$\Delta T_r = \frac{r}{c} \tag{2.20}$$

where c is the speed of light.

The systematic and arbitrary errors ΔTOA_{syst} and ΔTOA_{arb} are modeled in the same way as for the amplitude measurements, see Section 2.3.2.

2.3.4 Frequency measurements

The measured pulse frequency f is calculated from

$$f = f_0 + \Delta f_{syst} + \Delta f_{arb} \tag{2.21}$$

where f_0 is the true pulse frequency and the systematic and arbitrary frequency errors Δf_{syst} and Δf_{arb} are modeled in the same way as for the amplitude measurements, see Section 2.3.2.

2.3.5 Pulse width measurements

The measured pulse width (PW) is calculated from

$$PW = PW_0 + \Delta PW_{syst} + \Delta PW_{arb} \tag{2.22}$$

where PW_0 is the true pulse width and the systematic and arbitrary pulse width errors ΔPW_{syst} and ΔPW_{arb} are modeled in the same way as for the amplitude measurements, see Section 2.3.2.

2.3.6 AOA measurements

The measured angle of arrival (AOA) is calculated from

$$AOA = AOA_0 + \Delta AOA_{syst} + \Delta AOA_{arb} \tag{2.23}$$

where AOA_0 is the true angle of arrival and the systematic and arbitrary errors ΔAOA_{syst} and ΔAOA_{arb} are modeled in the same way as for the amplitude measurements, see Section 2.3.2.

For the AOA systematic error ΔAOA_{syst} there is an additional option, that simulates that the error has a sinusoidal dependency on the direction that the sensor receives the pulse from. The error is then calculated from

$$\Delta AOA_{syst} = A \cdot \sin\left[f \cdot (AOA - AOA_{ref}) \right] \tag{2.24}$$

where A is the error amplitude, f is the "frequency" of the error variation (number of sinus periods per 360°), and AOA_{ref} is the reference angle where ΔAOA_{syst}=0.

2.4 Possible future expansions to the PDW simulator software

The current version of the PDW simulator software simulates rotating radars in 2 dimensions on a flat Earth. Later versions of the software could be expanded to operate in 3 dimensions and to use an elliptic Earth (i.e., spherical coordinates) instead of the flat Earth approach. This would allow for airborne and /or spaceborne emitters and sensors to be included in the simulations. Obstacles could be added to the scenario, and integration of the software with 3D maps would open for the possibility to create realistic geographic scenarios. Non-rotating radars should be allowed for and additional parameters such as radar crystal frequency could be included. Additional measurements such as Doppler could also be performed. Further work should be done on improving the user interface of the simulator. This includes, amongst others, improved error handling and possibility to edit the radars and sensors that have been added to the scenario.

3 How to use the simulator?

The PDW simulator software was developed in Matlab 2012a, and use of the program requires Matlab to be installed on the computer. Start the simulator by typing “PDW_Simulator” in the Matlab command window. The following program window appears (Figure 3.1):

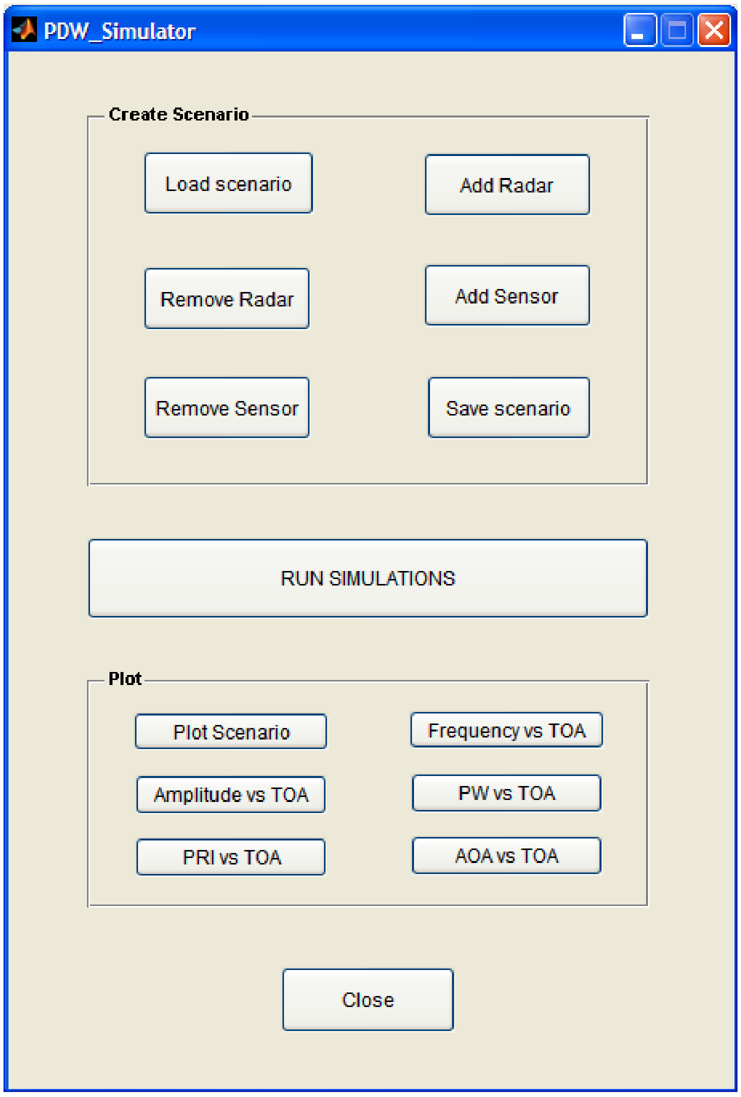


Figure 3.1 PDW simulator program window.

The simulator consists of three separate parts:

1) Create scenario
2) Run simulations
3) Plot results

First, a scenario must be created that will be used as input to the simulations. How to create the scenario will be described in Section 3.1.

After the scenario has been created, the PDW's are calculated by pressing the "Run Simulations" button (see Section 3.2 for details).

Finally, the results can be plotted by choosing from the options in the plot panel (Section 3.3).

3.1 Create scenario

It is possible to create an entirely new scenario or to use (and/or modify) an already existing scenario.

Create a new scenario by using the "Add Radar" and "Add Sensor" buttons to insert radars and sensors into the scenario. Finish by pressing the "Save Scenario" button to save the scenario.

Use an existing scenario by pressing the "Load Scenario" button to load the scenario into the simulator. The scenario can then be modified by using the "Remove Radar", "Remove Sensor", "Add Radar", and "Add Sensor" buttons. Finish by pressing the "Save Scenario" button to save the scenario.

The different buttons are described in more detail in subsections 3.1.1-3.1.6 below.

3.1.1 Load Scenario

Pressing the "Load Scenario" button in the program window in Figure 3.1 loads an existing scenario into the simulator. The program asks for the filename of the scenario you want to load (Figure 3.2). The default filename is "Scenario", which contains the scenario from the previous simulation. The file that contains the scenario must be placed in the folder "Scenario".

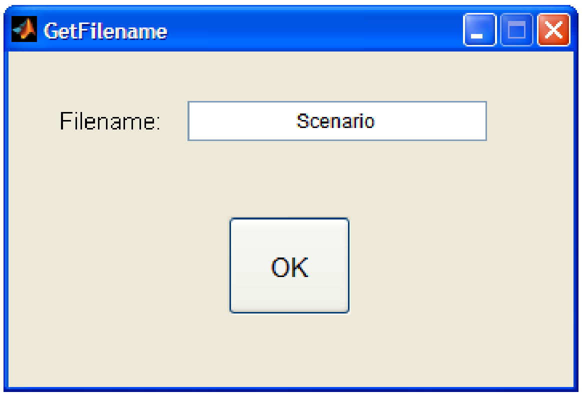
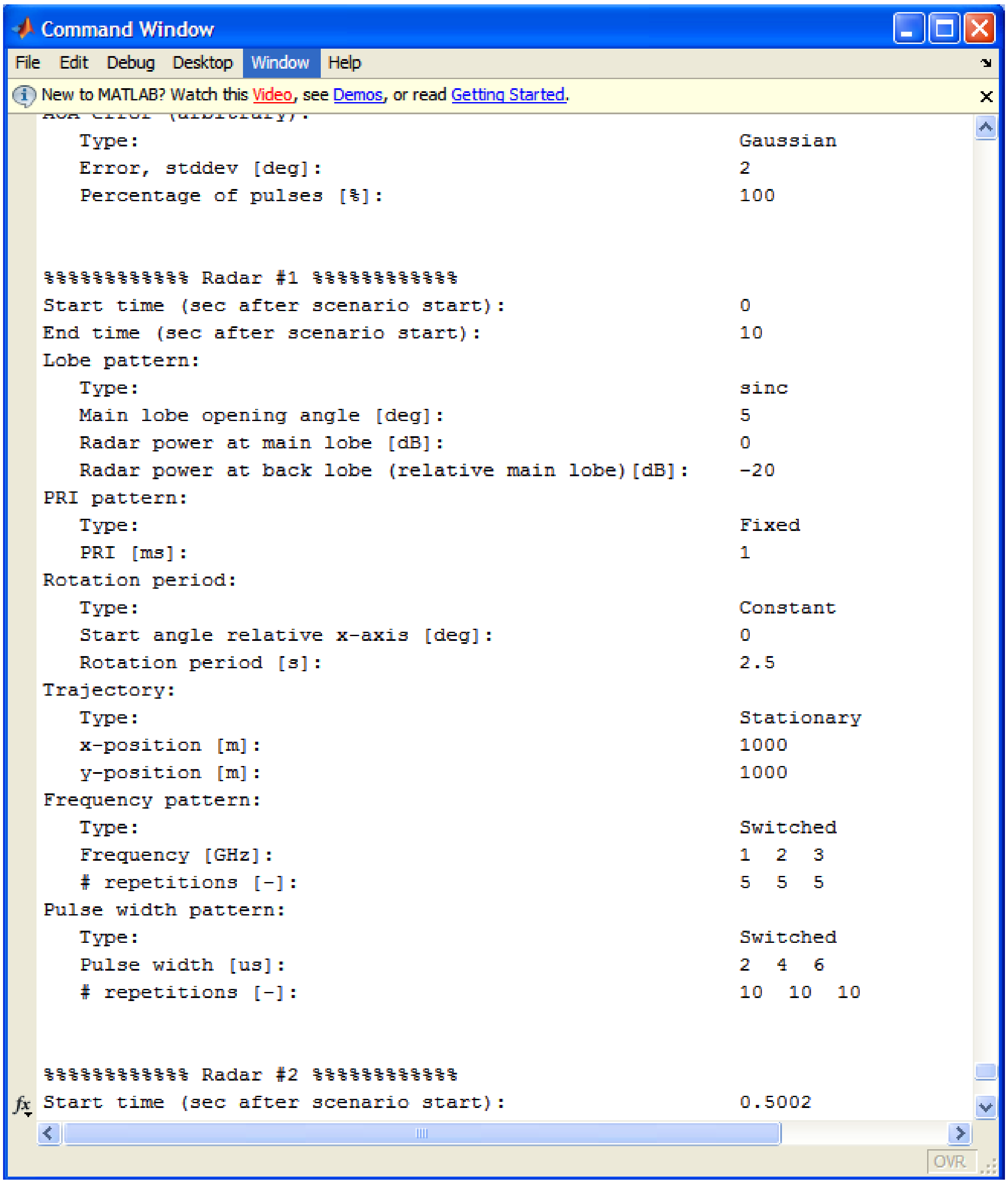


Figure 3.2 "Load Scenario" pop-up menu.

The loaded scenario is shown in the Matlab command window and lists all the sensors and radars with relevant parameters (Figure 3.3). It is then possible to inspect the content of the scenario and determine if any radars or sensors should be removed or added.


Command Window
File Edit Debug Desktop Window Help
New to MATLAB? Watch this Video, see Demos, or read Getting Started.
OVR

```
   Type:                                                Gaussian
   Error, stddev [deg]:                                 2
   Percentage of pulses [%]:                            100


%%%%%%%%%%%% Radar #1 %%%%%%%%%%%%
Start time (sec after scenario start):                  0
End time (sec after scenario start):                    10
Lobe pattern:
   Type:                                                sinc
   Main lobe opening angle [deg]:                       5
   Radar power at main lobe [dB]:                       0
   Radar power at back lobe (relative main lobe)[dB]:   -20
PRI pattern:
   Type:                                                Fixed
   PRI [ms]:                                            1
Rotation period:
   Type:                                                Constant
   Start angle relative x-axis [deg]:                   0
   Rotation period [s]:                                 2.5
Trajectory:
   Type:                                                Stationary
   x-position [m]:                                      1000
   y-position [m]:                                      1000
Frequency pattern:
   Type:                                                Switched
   Frequency [GHz]:                                     1  2  3
   # repetitions [-]:                                   5  5  5
Pulse width pattern:
   Type:                                                Switched
   Pulse width [us]:                                    2  4  6
   # repetitions [-]:                                   10  10  10


%%%%%%%%%%%% Radar #2 %%%%%%%%%%%%
Start time (sec after scenario start):                  0.5002
```

Figure 3.3 Scenario listed in Matlab command window (only part of the scenario is visible in this example).

3.1.2 Remove Radar

Pressing the “Remove Radar” button in the program window in Figure 3.1 gives the possibility to remove one or more radars from the scenario. A pop-up menu appears that asks for the number of the radar that should be removed (Figure 3.4). Press the “Remove another” button to remove further radars. Press “Finished” when you are done with removing radars.

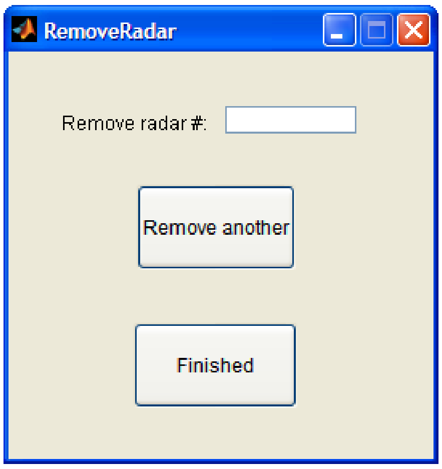


Figure 3.4 ”Remove Radar” pop-up menu.

3.1.3 Remove Sensor

Pressing the “Remove Sensor” button in the program window in Figure 3.1 gives the possibility to remove one or more sensors from the scenario. A pop-up menu appears that asks for the number of the sensor that should be removed (Figure 3.5). Press the “Remove another” button to remove further sensors. Press “Finished” when you are done with removing sensors.

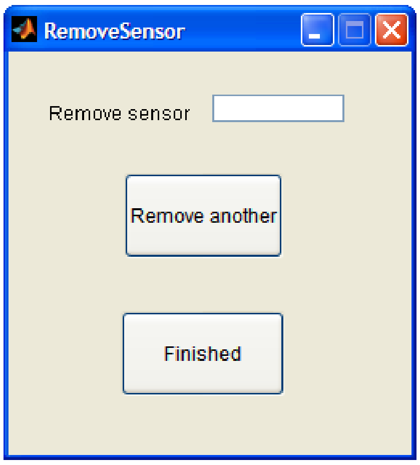


Figure 3.5 ”Remove Sensor” pop-up menu.

3.1.4 Add Radar

Press the “Add Radar” button in the program window (Figure 3.1) to add radars to the scenario. A pop-up menu appears that asks for the characteristics of the radar (Figure 3.6). Start time and end time for the radar’s transmission must be given together with information about the following:

1) Trajectory
2) Rotation period
3) Lobe pattern
4) PRI
5) Frequency
6) Pulse width

which will be described in more detail in the subsections below.

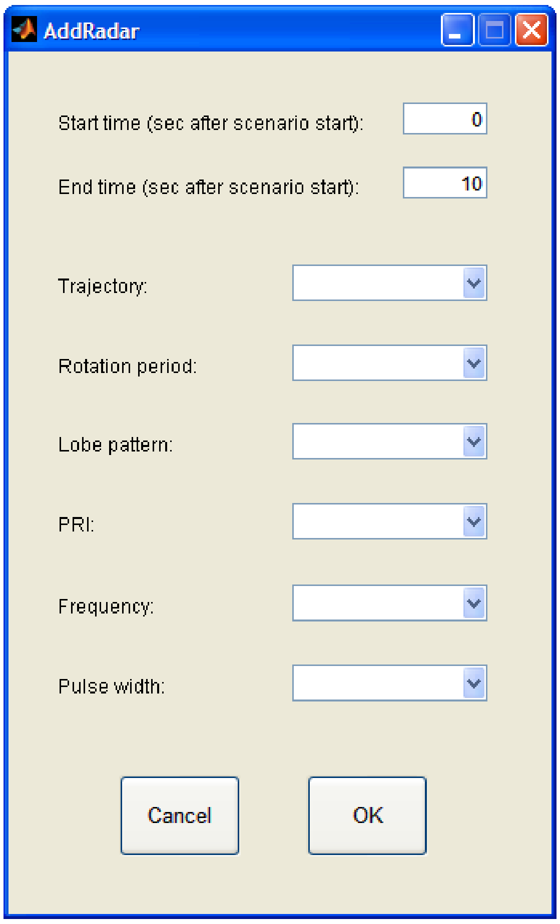


Figure 3.6 "Add Radar" pop-up menu.

3.1.4.1 Trajectory

For the radar trajectory it is possible to choose between the following options:

1) Stationary
2) Straight line
3) Arc
4) From file

The option "Stationary" simulates a stationary radar, and information about the radar's *x*- and *y*-positions must be given (Figure 3.7)

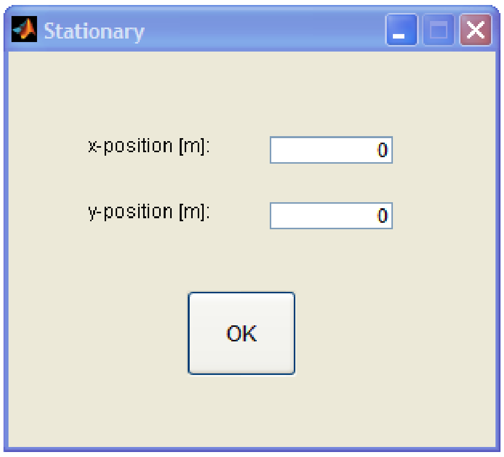


Figure 3.7 ”Trajectory” pop-up menu when the option “Stationary” is chosen.

The option “Straight line” simulates a radar that moves with constant speed along a straight line. Information about the radar’s *x*- and *y*- positions at the beginning of the movement (x_0, y_0), the speed (v), and the angle (α) of the radar’s trajectory relative to the *x*-axis must be given (Figure 3.8). See also Section 2.1.1.

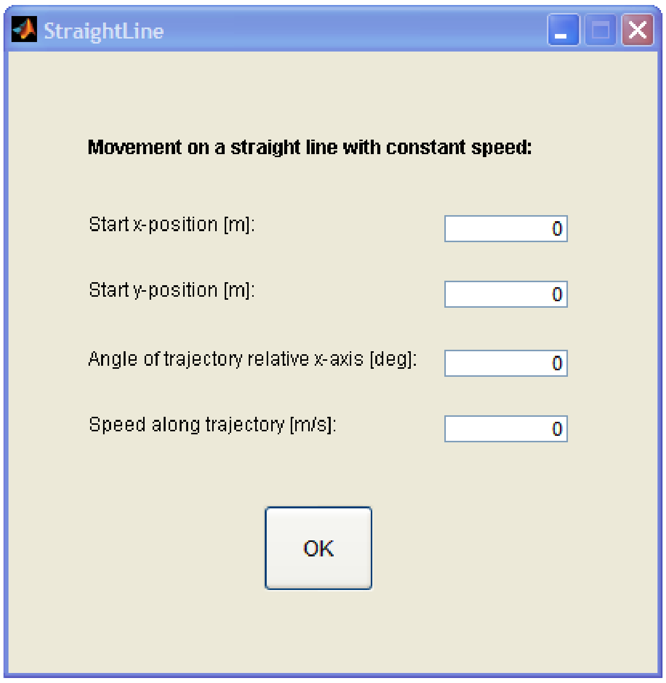
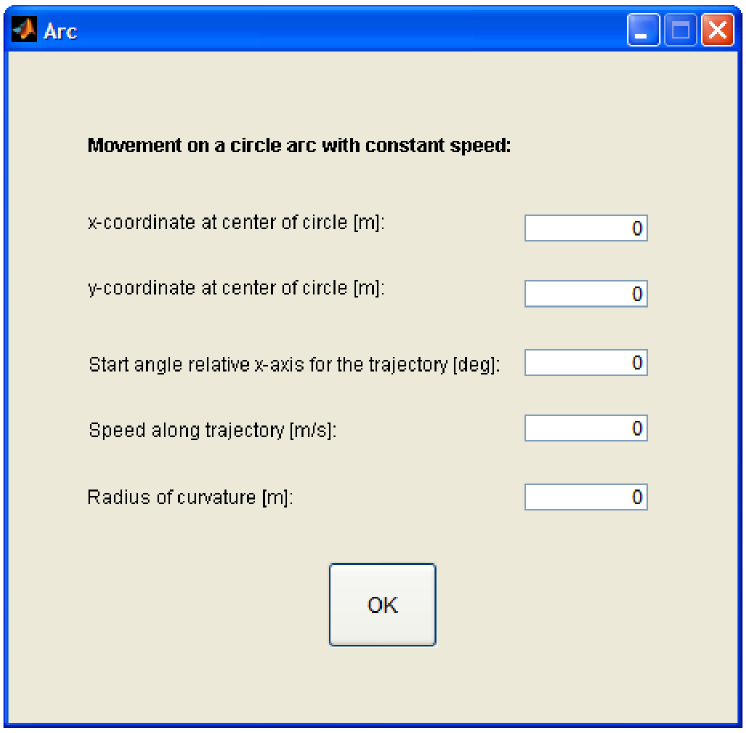

StraightLine
Movement on a straight line with constant speed:
Start x-position [m]: 0
Start y-position [m]: 0
Angle of trajectory relative x-axis [deg]: 0
Speed along trajectory [m/s]: 0
OK

Figure 3.8 ”Trajectory” pop-up menu when the option “Straight line” is chosen.

The option “Arc” simulates a radar that moves on a circle arc with constant speed. The *x*- and *y*-coordinates for the center of the circle (x_0, y_0) must be given together with the radius of curvature (r). Information about the radar’s speed (v) and the trajectory’s start angle (α_0) relative to the *x*-axis must also be given (Figure 3.9). See also Section 2.1.2.


Movement on a circle arc with constant speed:

x-coordinate at center of circle [m]:	0
y-coordinate at center of circle [m]:	0
Start angle relative x-axis for the trajectory [deg]:	0
Speed along trajectory [m/s]:	0
Radius of curvature [m]:	0

Figure 3.9 "Trajectory" pop-up menu when the option "Arc" is chosen.

The option "From file" uses *x*- and *y*-coordinates (at given times) that are read from a text-file, to simulate the radar's movement. The file must be placed in the folder "Inputfiles" prior to the simulations and the content of the file must be arranged as follows: Column #1 contains time stamps for the positions, column #2 contains the corresponding *x*-coordinates, and column #3 contains the corresponding *y*-coordinates. The name of the file must be given in the corresponding pop-up menu (Figure 3.10).

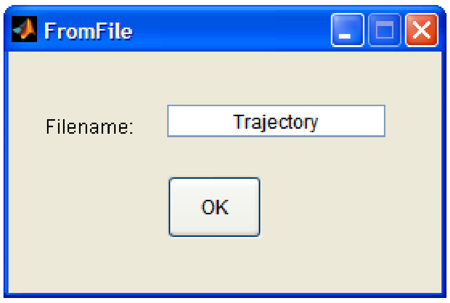

Filename: Trajectory

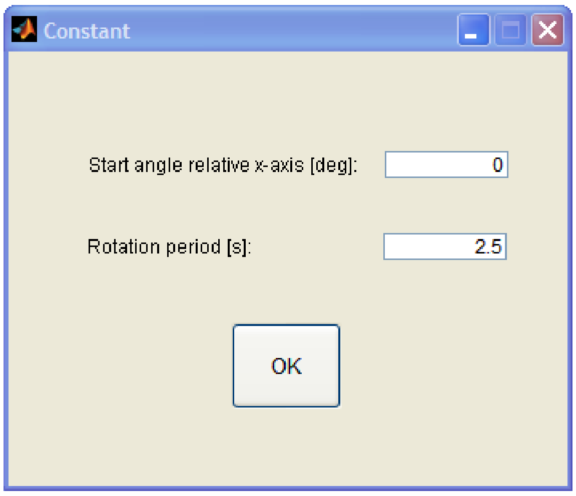
Figure 3.10 "Trajectory" pop-up menu when the option "From file" is chosen.

3.1.4.2 Rotation period

For the radar rotation period it is possible to choose between the following options:

1) Constant
2) Sinus

The option “Constant” simulates a radar with constant rotation period. The rotation period (T_{rot}) must be given together with the main lobe pointing angle (α_0) relative the x-axis at the scenario start time t_0 (Figure 3.11). See also Section 2.2.1.1.

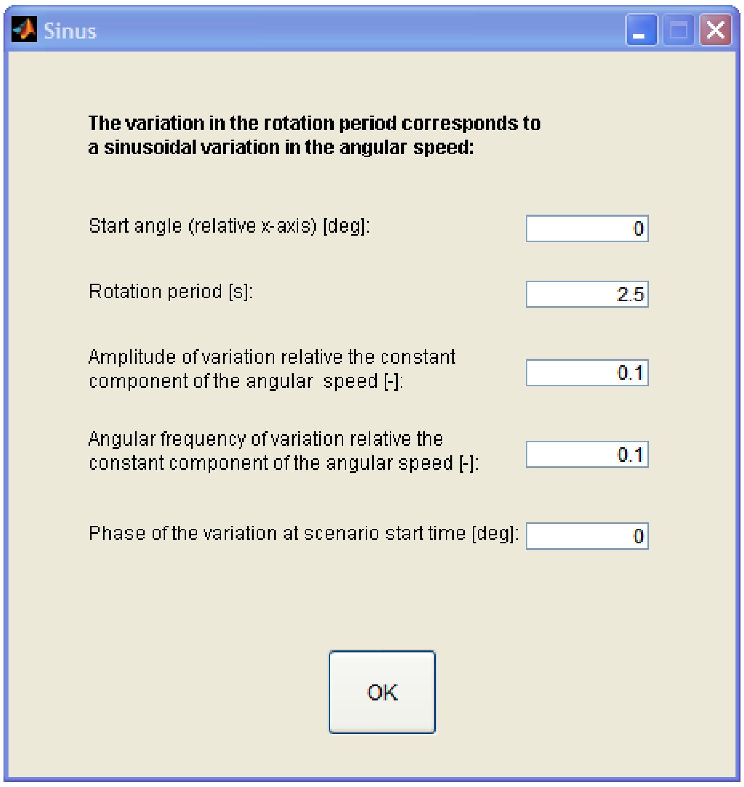


Start angle relative x-axis [deg]: 0

Rotation period [s]: 2.5

Figure 3.11 ”Rotation period” pop-up menu when the option “Constant” is chosen.

The option “Sinus” simulates a radar with sinusoidal varying angular speed. The mean rotation period (T_{rot}) must be given together with the main lobe pointing angle (α_0) relative the x-axis at the scenario start time t_0 (Figure 3.12). The amplitude (A) and angular frequency (s) of the variation relative to the constant component of the angular speed must also be given together with the phase (φ_0) of the variation at the scenario start time. See also Section 2.2.1.2.



The variation in the rotation period corresponds to a sinusoidal variation in the angular speed:

Start angle (relative x-axis) [deg]: 0

Rotation period [s]: 2.5

Amplitude of variation relative the constant component of the angular speed [-]: 0.1

Angular frequency of variation relative the constant component of the angular speed [-]: 0.1

Phase of the variation at scenario start time [deg]: 0

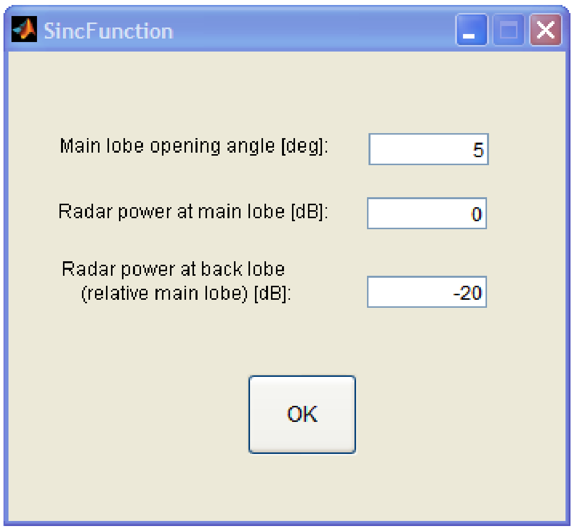
Figure 3.12 ”Rotation period” pop-up menu when the option “Sinus” is chosen.

3.1.4.3 Lobe pattern

For the radar lobe pattern it is possible to choose between the following options:

1) Sinc-function
2) From file

The option “Sinc-function” uses a modified sinc-function to describe the radar lobe pattern, see Figure 2.6 in Section 2.2.2.1. Main lobe opening angle (θ_{ML}), radar power (i.e., amplitude) at main lobe (P_{ML}), and radar power at back lobe (P_{BL}) must be given (Figure 3.13).



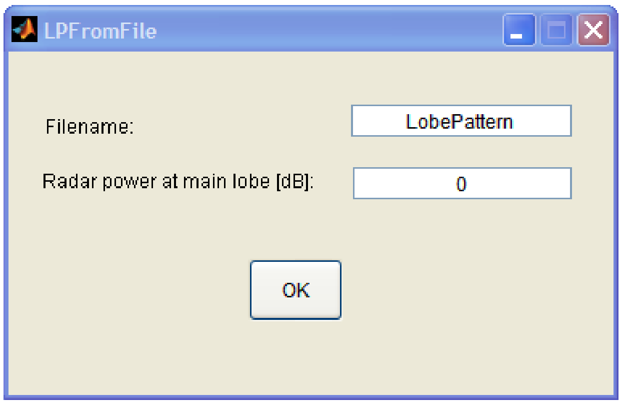
Main lobe opening angle [deg]: 5

Radar power at main lobe [dB]: 0

Radar power at back lobe (relative main lobe) [dB]: -20

Figure 3.13 ”Lobe pattern” pop-up menu when the option “Sinc-function” is chosen.

The option “From file” uses angles with corresponding lobe pattern amplitudes that are read from a text-file, to simulate the radar antenna’s lobe pattern. The file must be placed in the folder “Inputfiles” prior to the simulations and the content of the file must be arranged as follows: Column #1 contains the angles [deg] and column #2 contains the corresponding amplitudes [dB]. The amplitude values given in the file must be scaled so that the amplitude of the main lobe is equal to 0 dB. The name of the file together with the radar power at the main lobe (i.e., the strength of the radar) must be given in the corresponding pop-up menu (Figure 3.14).



Filename: LobePattern

Radar power at main lobe [dB]: 0

Figure 3.14 ”Lobe pattern” pop-up menu when the option “From File” is chosen.

3.1.4.4 PRI

For the radar PRI it is possible to choose between the following options:

1) Fixed
2) Stagger
3) Switched
4) Jitter

The option "Fixed" simulates a constant PRI for the radar, see Figure 2.7 in Section 2.2.3.1 for an example. The PRI must be given in the corresponding pop-up menu (Figure 3.15).

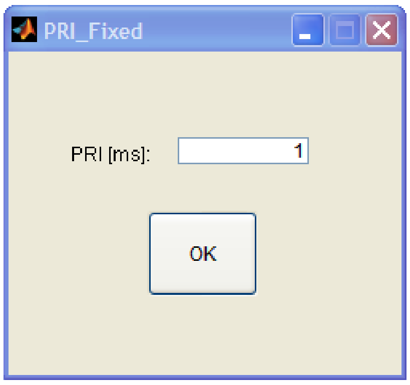


Figure 3.15 "PRI" pop-up menu when the option "Fixed" is chosen.

The option "Stagger" simulates stagger PRI for the radar, i.e., that the radar uses several PRIs in a fixed pattern, see Figure 2.8 in Section 2.2.3.2 for an example. The PRIs must be given in the corresponding pop-up menu (Figure 3.16).

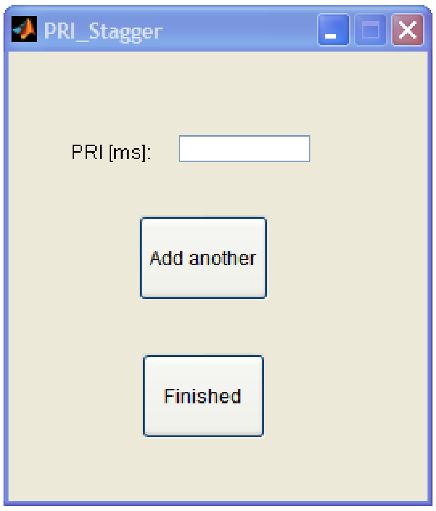


Figure 3.16 "PRI" pop-up menu when the option "Stagger" is chosen.

The option “Switched” simulates that the radar uses several PRIs in a fixed pattern, where each PRI may be repeated a certain number of times, see Figure 2.9 in Section 2.2.3.3 for an example. The PRIs must be given together with their corresponding number of repetitions (Figure 3.17).

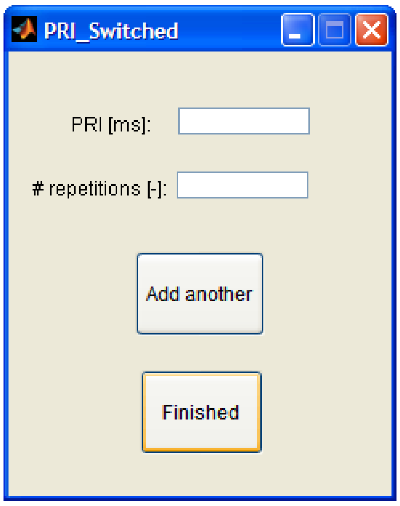
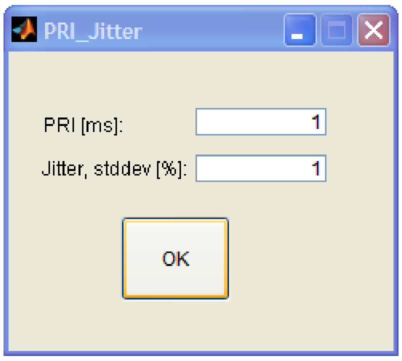


Figure 3.17 ”PRI” pop-up menu when the option “Switched” is chosen.

The option “Jitter” simulates a radar with jittered PRI, see Figure 2.10 in Section 2.2.3.4 for an example. The PRI mean value and standard deviation (jitter) must be given in the corresponding pop-up menu (Figure 3.18).


PRI_Jitter

PRI [ms]: 1

Jitter, stddev [%]: 1

Figure 3.18 ”PRI” pop-up menu when the option “Jitter” is chosen.

3.1.4.5   Frequency

For the frequency the options are the same as for the PRI, see Section 3.1.4.4.

3.1.4.6   Pulse width

For the pulse width the options are the same as for the PRI, see Section 3.1.4.4.

3.1.5 Add Sensor

Press the “Add Sensor” button in the program window (Figure 3.1) to add sensors to the scenario. A pop-up menu appears that asks for the characteristics of the sensor (Figure 3.19). The following information about the sensor must be given:

1) Trajectory
2) Saturation level
3) Detection probability
4) Amplitude error
5) TOA error
6) Frequency error
7) Pulse width error
8) AOA error

Each of these parameters will be described in more detail in the subsections below.

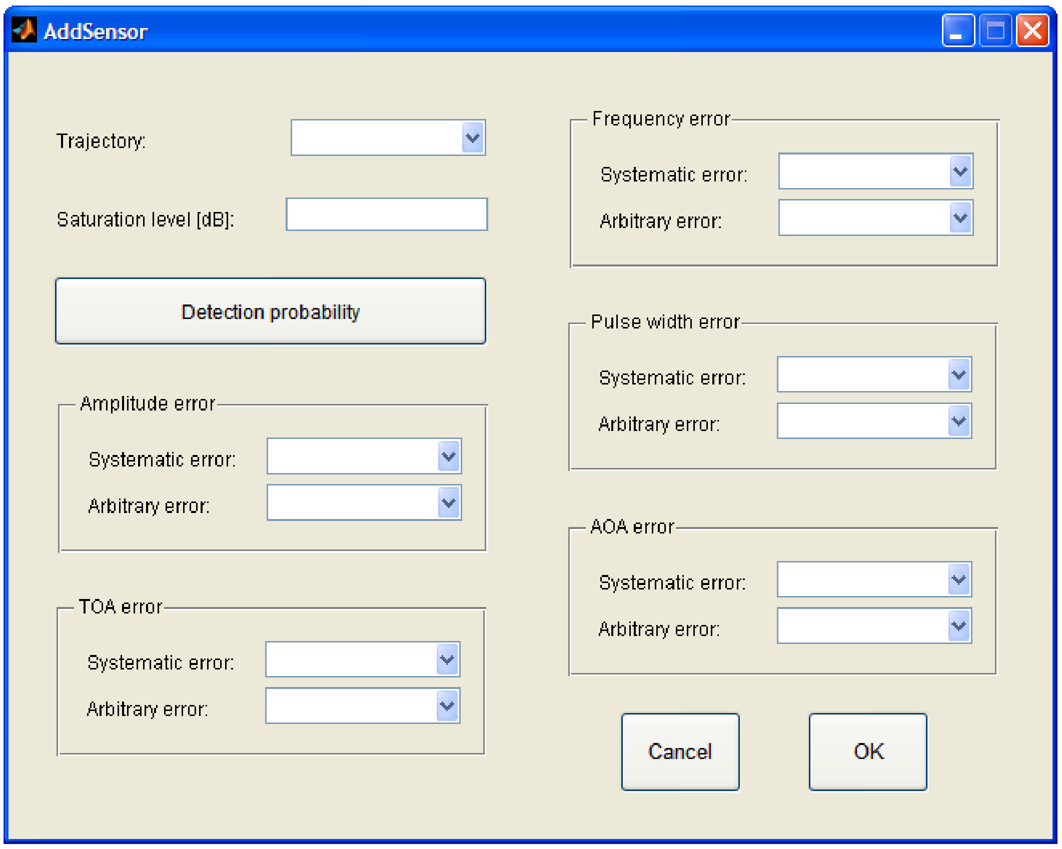


Figure 3.19 ”Add Sensor” pop-up menu.

3.1.5.1 Trajectory

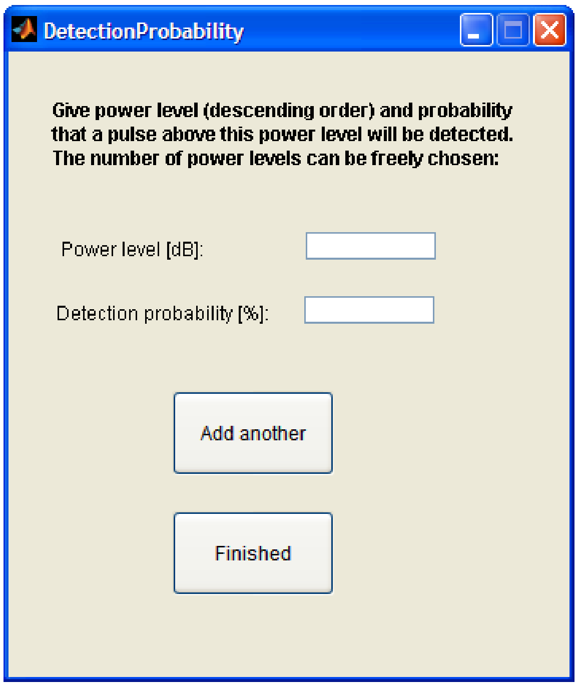
For the sensor trajectory the options are the same as for the radar trajectory, see Section 3.1.4.1.

3.1.5.2 Saturation level

Here the saturation level for the sensor is given.

3.1.5.3 Detection probability

The probability that a sensor detects a given pulse is assumed to be determined only by the pulse amplitude. To describe the detection probability, type pulse amplitude levels (descending order) together with corresponding detection probabilities in the detection probability pop-up menu (Figure 3.20). All pulses above a certain amplitude level (and at the same time below the previous higher amplitude level) will be detected with the corresponding detection probability. Pulses that are below the lowest given amplitude level are not detected.


DetectionProbability

Give power level (descending order) and probability that a pulse above this power level will be detected. The number of power levels can be freely chosen:

Power level [dB]:

Detection probability [%]:

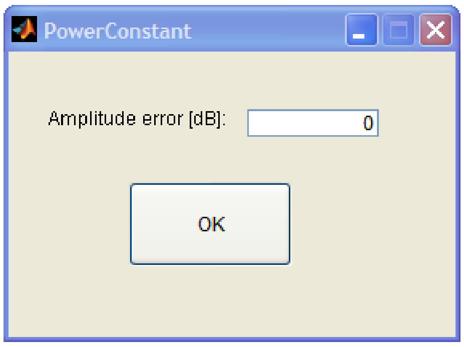
Figure 3.20 ”Detection probability” pop-up menu.

3.1.5.4 Amplitude error

Amplitude errors may be both systematic and arbitrary. The following options exist for systematic errors:

1) Constant
2) Linear
3) Sinus

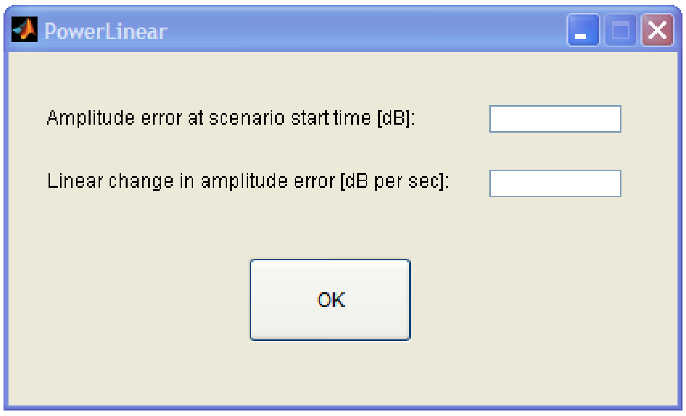
The option “Constant” simulates a constant systematic error. The value for the constant error must be given (Figure 3.21).


PowerConstant

Amplitude error [dB]: 0

Figure 3.21 "Systematic amplitude error" pop-up menu when the option "Constant" is chosen.

The option "Linear" simulates a systematic error that changes linearly with time (see also Section 2.3.2). The error (ΔP_0) at the scenario start time (t_0) must be given together with the rate of change (k) in the error (Figure 3.22).

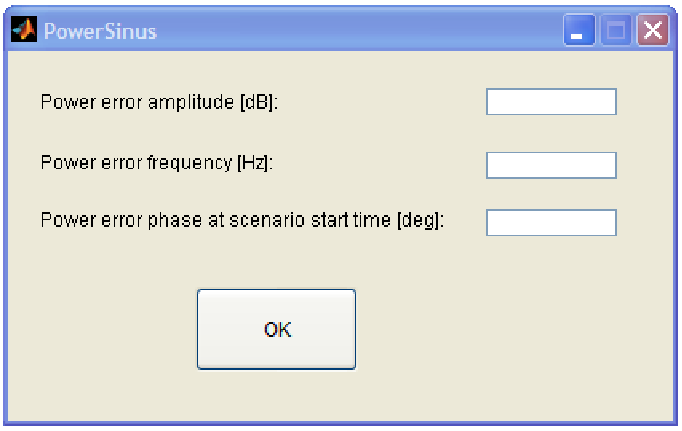

PowerLinear

Amplitude error at scenario start time [dB]:

Linear change in amplitude error [dB per sec]:

Figure 3.22 "Systematic amplitude error" pop-up menu when the option "Linear" is chosen.

The option "Sinus" simulates a systematic error that varies as a sinus with time (see also Section 2.3.2). The error amplitude (A), frequency (f), and phase (φ_0) at the scenario start time (t_0) must be given (Figure 3.23).


PowerSinus

Power error amplitude [dB]:

Power error frequency [Hz]:

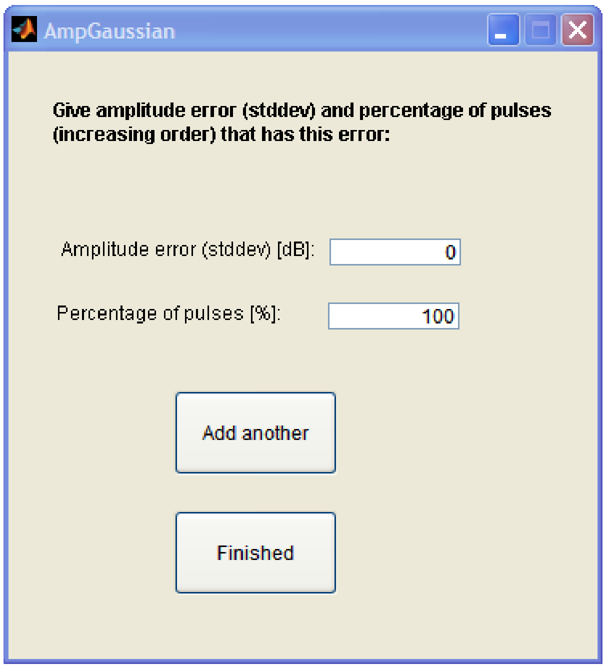
Power error phase at scenario start time [deg]:

Figure 3.23 "Systematic amplitude error" pop-up menu when the option "Sinus" is chosen.

For the arbitrary amplitude error the following option can be chosen:

1) Gaussian

The option “Gaussian” simulates an arbitrary error that has Gaussian distribution. The standard deviation of the amplitude error must be given together with the percentage of pulses that has this error. It is possible to set several amplitude error standard deviation levels. The total percentage must add up to 100%.


AmpGaussian

Give amplitude error (stddev) and percentage of pulses (increasing order) that has this error:

Amplitude error (stddev) [dB]: 0

Percentage of pulses [%]: 100

Figure 3.24 ”Arbitrary amplitude error” pop-up menu when the option “Gaussian” is chosen.

3.1.5.5 TOA error

The options for the TOA error are the same as for the amplitude error, see Section 3.1.5.4.

3.1.5.6 Frequency error

The options for the frequency error are the same as for the amplitude error, see Section 3.1.5.4.

3.1.5.7 Pulse width error

The options for the pulse width error are the same as for the amplitude error, see Section 3.1.5.4.

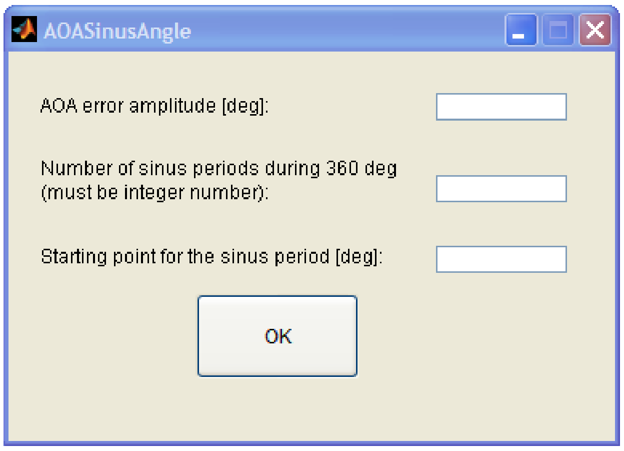
3.1.5.8 AOA error

The options for the AOA error are the same as for the amplitude error, see Section 3.1.5.4.

In addition there is an extra option for the systematic AOA error:

4) SinusAngle

This option simulates a systematic error with sinusoidal dependency on the direction that the sensor receives the signal from (see also Section 2.3.6). The sinusoidal AOA error amplitude (*A*) together with the number (*f*) of sinus periods per 360º and the reference angle AOA_{ref} must be given (Figure 3.25).


AOASinusAngle

AOA error amplitude [deg]:

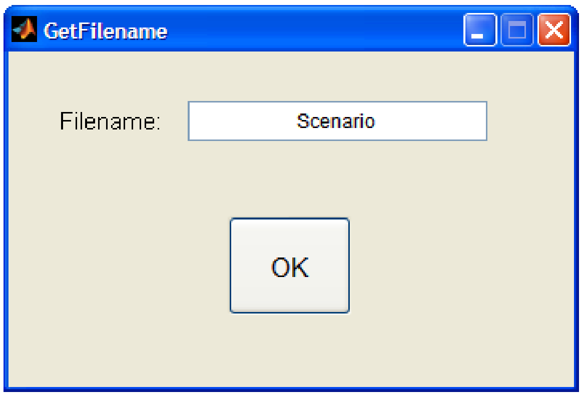
Number of sinus periods during 360 deg (must be integer number):

Starting point for the sinus period [deg]:

Figure 3.25 "Systematic AOA error" pop-up menu when the option "SinusAngle" is chosen.

3.1.6 Save Scenario

Pressing the "Save Scenario" button in the program window in Figure 3.1 saves the scenario. The scenario is saved automatically to the file "Scenario" in the folder "Scenario", which is used in the further simulations. In addition, a pop-up menu gives the possibility to save the scenario also under a different name for later use (Figure 3.26). This file is also placed in the folder "Scenario".


GetFilename

Filename: Scenario

Figure 3.26 "Save Scenario" pop-up menu.

3.2 Run simulations

By pressing the “Run Simulations” button in the program window shown in Figure 3.1, the PDWs are created based on the input scenario. The following parameters are calculated[1]:

1) Time of arrival (TOA)
2) Amplitude
3) Frequency
4) Pulse width (PW)
5) Angle of arrival (AOA)

Each parameter is written to separate files for each sensor. The following filenames are used:

- Sensor<*n*>_TOA
- Sensor<*n*>_Amplitude
- Sensor<*n*>_Frequency
- Sensor<*n*>_PW
- Sensor<*n*>_AOA

The second part of the filename shows which parameter is recorded in the file, while the first part of the filename shows which sensor has recorded this parameter (replace <*n*> with sensor number; 1, 2, …etc). For instance, the file “Sensor1_TOA” contains all the TOAs recorded by sensor 1.

In addition, information about which radar and which pulse the parameters belong to is recorded in two additional files:

- Sensor<*n*>_RadarNo
- Sensor<*n*>_PulseNo

All parameters are listed in the same order (increasing TOA), so that the first element of “sensor1_TOA” corresponds to the first element of “sensor1_RadarNo”, etc.

Radar and sensor positions are also calculated during the simulations and are recorded in the following files (x- and y-position separately):

- Radar<*n*>_X
- Radar<*n*>_Y
- Sensor<*n*>_X
- Sensor<*n*>_Y

All files are stored in the folder “Outputfiles”.

[1] See Section 2.3 for details on which factors are included in the calculation of each parameter.

3.3 Plot results

Results from the simulations can be plotted by choosing between the following options in the program window shown in Figure 3.1:

1) Plot Scenario
2) Amplitude vs TOA
3) PRI vs TOA
4) Frequency vs TOA
5) PW vs TOA
6) AOA vs TOA

The option "Plot Scenario" plots the positions of sensors and radars in the scenario, see Figure 4.1 in Chapter 4 for an example.

The option "Amplitude vs TOA" plots the amplitude of the received radar pulses as a function of TOA, see Figure 4.7 in Chapter 4 for an example. Pulses from different radars are marked by different colours, and each sensor is represented by a separate figure.

The option "PRI vs TOA" plots the PRI of the received radar pulses as a function of TOA, see Figure 4.9 in Chapter 4 for an example.

The option "Frequency vs TOA" plots the frequency of the received radar pulses as a function of TOA, see Figure 4.8 in Chapter 4 for an example.

The option "PW vs TOA" plots the pulse width of the received radar pulses as a function of TOA, see Figure 4.10 in Chapter 4 for an example.

The option "AOA vs TOA" plots the AOA of the received radar pulses as a function of TOA, see Figure 4.11 in Chapter 4 for an example.

4 Example

We will in this chapter demonstrate the use of the PDW simulator and discuss the resulting output. Figure 4.1 shows the scenario to be used. The scenario consists of two stationary sensors and three radars, where radar 1 is stationary, radar 2 moves along a circle arc with a constant speed of 5 m/s, and radar 3 moves along a straight line with a constant speed of 8 m/s.

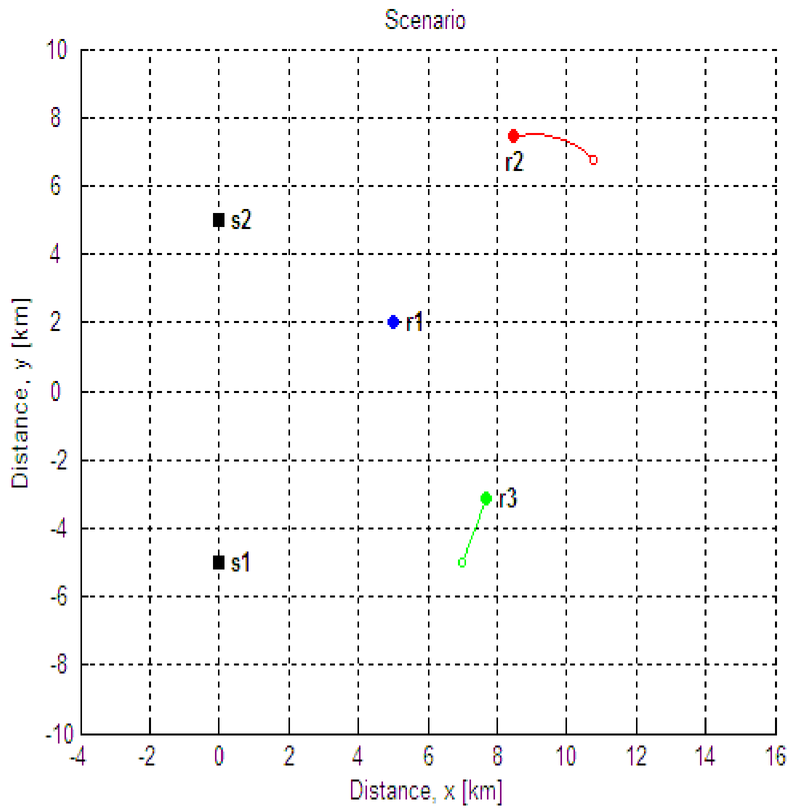
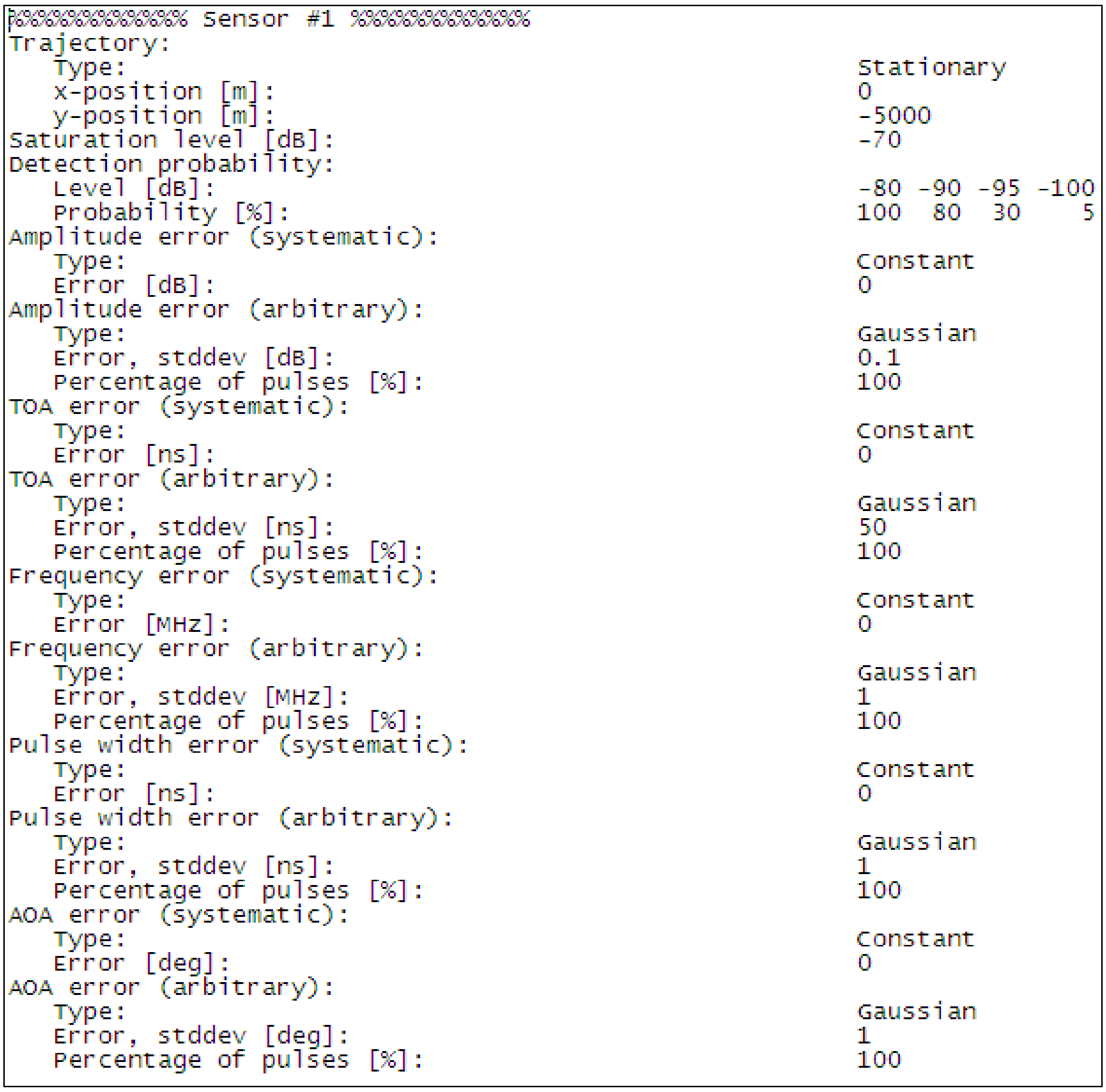


Figure 4.1 Scenario example with two stationary sensors (s1 and s2) and three radars (r1, r2, and r3).

Figure 4.2 and Figure 4.3 show the input parameters for sensors 1 and 2. We see that sensor 2 has somewhat better pulse detection probability and makes more precise measurements than sensor 1.

Figure 4.4-Figure 4.6 show the input parameters for radars 1-3. Radar 2 is the most powerful of the three radars with a maximum pulse amplitude that lies 5 dB above that of radar 3 and 10 dB above that of radar 1. The radars have rotation periods between 2 s and 3 s and main lobe opening angles between 1° and 5°. Frequencies are in the range 9.2-9.6 GHz and pulse widths are in the range 1.2 µs - 1.8 µs. The radars use different PRI patterns with PRIs in the range 0.67-0.9 ms.

Figure 4.7-Figure 4.11 show plots of the resulting output parameters from the PDW simulator, such as pulse amplitude, frequency, PRI, pulse width, and AOA. The plots are shown for a selected time interval of 1 s.

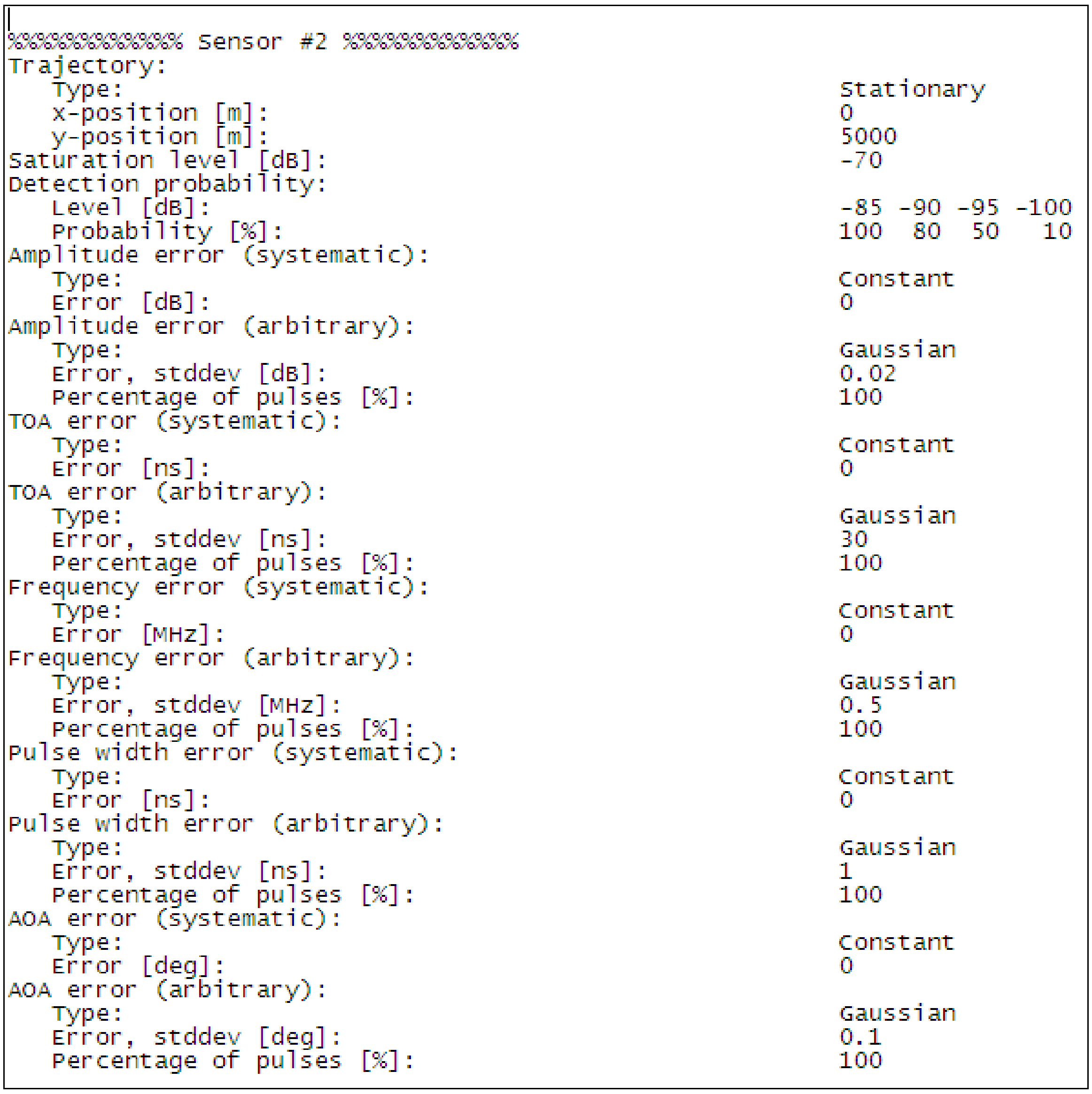


```
%%%%%%%%%%%% Sensor #1 %%%%%%%%%%%%
Trajectory:
   Type:                                                  Stationary
   x-position [m]:                                        0
   y-position [m]:                                        -5000
Saturation level [dB]:                                    -70
Detection probability:
   Level [dB]:                                            -80  -90  -95  -100
   Probability [%]:                                       100   80   30     5
Amplitude error (systematic):
   Type:                                                  Constant
   Error [dB]:                                            0
Amplitude error (arbitrary):
   Type:                                                  Gaussian
   Error, stddev [dB]:                                    0.1
   Percentage of pulses [%]:                              100
TOA error (systematic):
   Type:                                                  Constant
   Error [ns]:                                            0
TOA error (arbitrary):
   Type:                                                  Gaussian
   Error, stddev [ns]:                                    50
   Percentage of pulses [%]:                              100
Frequency error (systematic):
   Type:                                                  Constant
   Error [MHz]:                                           0
Frequency error (arbitrary):
   Type:                                                  Gaussian
   Error, stddev [MHz]:                                   1
   Percentage of pulses [%]:                              100
Pulse width error (systematic):
   Type:                                                  Constant
   Error [ns]:                                            0
Pulse width error (arbitrary):
   Type:                                                  Gaussian
   Error, stddev [ns]:                                    1
   Percentage of pulses [%]:                              100
AOA error (systematic):
   Type:                                                  Constant
   Error [deg]:                                           0
AOA error (arbitrary):
   Type:                                                  Gaussian
   Error, stddev [deg]:                                   1
   Percentage of pulses [%]:                              100
```

Figure 4.2 Input parameters for sensor 1.

Figure 4.7a) and b) show the pulse amplitude as a function of TOA for pulses received by sensor 1 and sensor 2 respectively. The main lobe from each of the three radars passes sensor 2 at approximately the same time. Short after, the main lobe of radar 2 passes sensor 1 followed somewhat later by radar 3 and radar 1. The radars are differently placed relative to the sensors (thereby seeing a different angular distance between the sensors) and they have different rotation periods. Radar 2 has the shortest rotation period (2 s), followed by radar 1 (2.5 s) and radar 3 (3 s).

Even though radar 2 is the most powerful radar, we see that its maximum amplitude as detected by sensor 1 is lower than that of radar 3 which has the highest detected maximum amplitude of all three radars at sensor 1 (Figure 4.7a). The reason for this is that radar 2 is placed at a much longer distance from sensor 1 than radar 3. At sensor 2 the situation is opposite. Here the detected maximum amplitude of radar 3 is the lowest of all three radars. This is because radar 3 is placed at the longest distance from sensor 2.

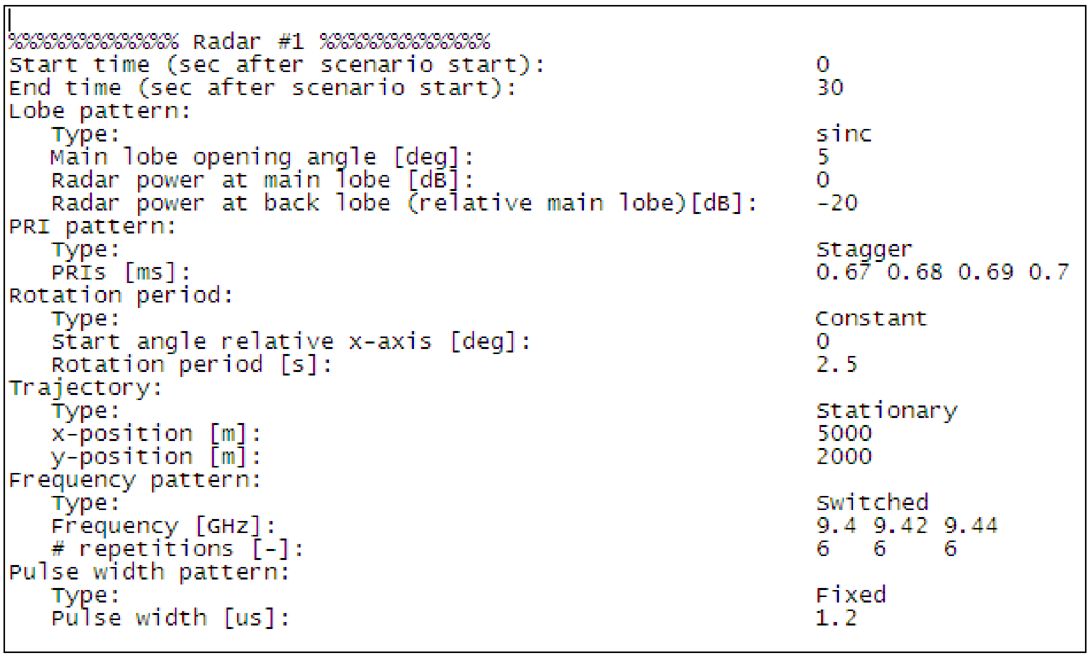


```
%%%%%%%%%%%% Sensor #2 %%%%%%%%%%%%
Trajectory:
   Type:                                                  Stationary
   x-position [m]:                                        0
   y-position [m]:                                        5000
Saturation level [dB]:                                    -70
Detection probability:
   Level [dB]:                                            -85 -90 -95 -100
   Probability [%]:                                       100  80  50   10
Amplitude error (systematic):
   Type:                                                  Constant
   Error [dB]:                                            0
Amplitude error (arbitrary):
   Type:                                                  Gaussian
   Error, stddev [dB]:                                    0.02
   Percentage of pulses [%]:                              100
TOA error (systematic):
   Type:                                                  Constant
   Error [ns]:                                            0
TOA error (arbitrary):
   Type:                                                  Gaussian
   Error, stddev [ns]:                                    30
   Percentage of pulses [%]:                              100
Frequency error (systematic):
   Type:                                                  Constant
   Error [MHz]:                                           0
Frequency error (arbitrary):
   Type:                                                  Gaussian
   Error, stddev [MHz]:                                   0.5
   Percentage of pulses [%]:                              100
Pulse width error (systematic):
   Type:                                                  Constant
   Error [ns]:                                            0
Pulse width error (arbitrary):
   Type:                                                  Gaussian
   Error, stddev [ns]:                                    1
   Percentage of pulses [%]:                              100
AOA error (systematic):
   Type:                                                  Constant
   Error [deg]:                                           0
AOA error (arbitrary):
   Type:                                                  Gaussian
   Error, stddev [deg]:                                   0.1
   Percentage of pulses [%]:                              100
```

Figure 4.3 Input parameters for sensor 2.

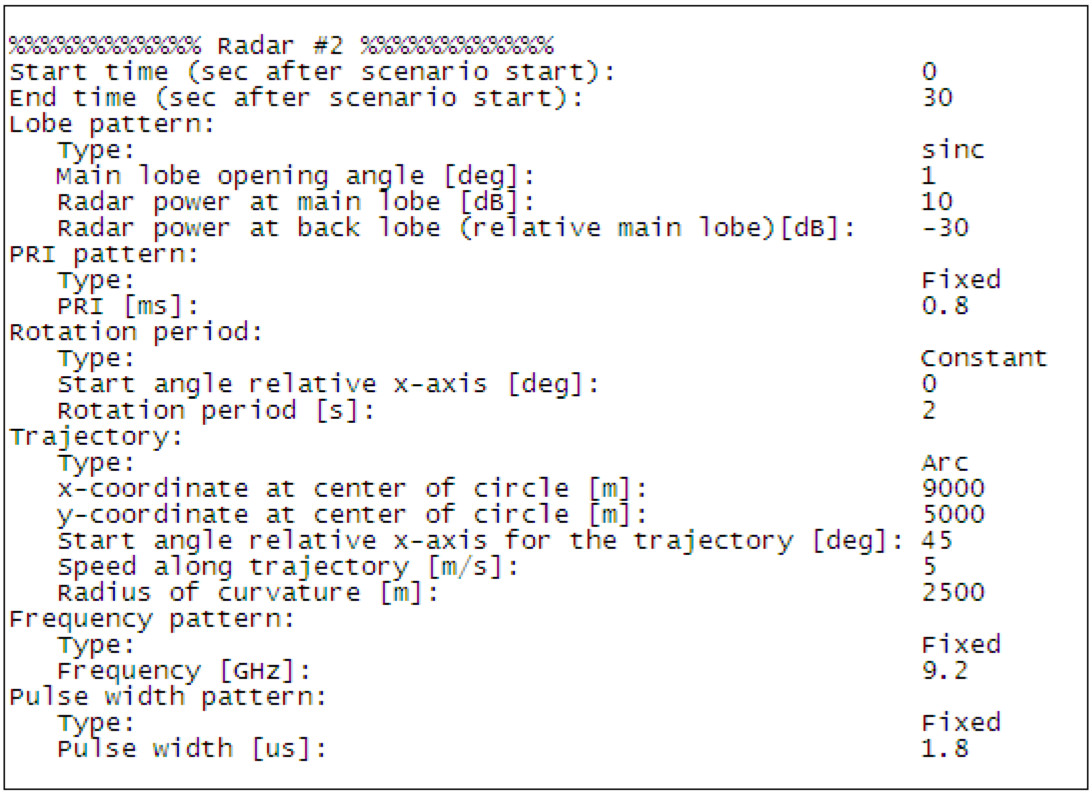
Figure 4.7c) shows a selected time interval from Figure 4.7b) with pulses received by sensor 2 during 100 ms between TOA=21.0 s and TOA=21.1 s. It is here possible to see single pulses, and we see that in the central parts of the three main lobes all pulses are detected. However, in the side lobes several pulses are missing. The amplitude measurements have arbitrary errors with standard deviation 0.1 dB for sensor 1 and 0.02 dB for sensor 2.

Figure 4.8a) and b) show the pulse frequency as a function of TOA for pulses received by sensor 1 and sensor 2 respectively. We notice that more pulses are detected by sensor 1 than by sensor 2 for radar 3. This is because the signal from radar 3 is stronger at sensor 1 than at sensor 2 (see Figure 4.7). For radars 1 and 2 the situation is opposite; more pulses are detected by sensor 2. In addition to receiving stronger signal from radars 1 and 2 than sensor 1, sensor 2 also has better ability to detect pulses (see Figure 4.2 and Figure 4.3).



```
%%%%%%%%%%%% Radar #1 %%%%%%%%%%%%
Start time (sec after scenario start):                  0
End time (sec after scenario start):                    30
Lobe pattern:
   Type:                                                sinc
   Main lobe opening angle [deg]:                       5
   Radar power at main lobe [dB]:                       0
   Radar power at back lobe (relative main lobe)[dB]:   -20
PRI pattern:
   Type:                                                Stagger
   PRIs [ms]:                                           0.67 0.68 0.69 0.7
Rotation period:
   Type:                                                Constant
   Start angle relative x-axis [deg]:                   0
   Rotation period [s]:                                 2.5
Trajectory:
   Type:                                                Stationary
   x-position [m]:                                      5000
   y-position [m]:                                      2000
Frequency pattern:
   Type:                                                Switched
   Frequency [GHz]:                                     9.4  9.42  9.44
   # repetitions [-]:                                   6    6     6
Pulse width pattern:
   Type:                                                Fixed
   Pulse width [us]:                                    1.2
```

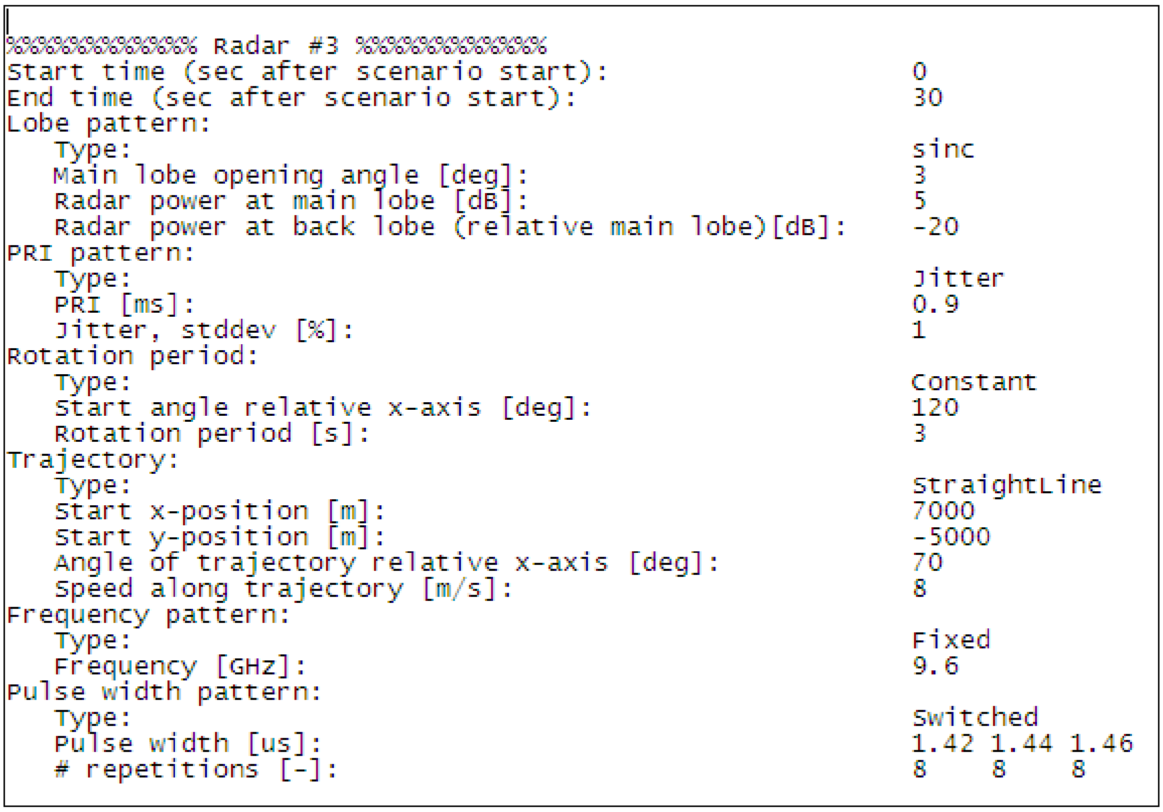
Figure 4.4 Input parameters for radar 1.



```
%%%%%%%%%%%% Radar #2 %%%%%%%%%%%%
Start time (sec after scenario start):                      0
End time (sec after scenario start):                        30
Lobe pattern:
   Type:                                                    sinc
   Main lobe opening angle [deg]:                           1
   Radar power at main lobe [dB]:                           10
   Radar power at back lobe (relative main lobe)[dB]:       -30
PRI pattern:
   Type:                                                    Fixed
   PRI [ms]:                                                0.8
Rotation period:
   Type:                                                    Constant
   Start angle relative x-axis [deg]:                       0
   Rotation period [s]:                                     2
Trajectory:
   Type:                                                    Arc
   x-coordinate at center of circle [m]:                    9000
   y-coordinate at center of circle [m]:                    5000
   Start angle relative x-axis for the trajectory [deg]:    45
   Speed along trajectory [m/s]:                            5
   Radius of curvature [m]:                                 2500
Frequency pattern:
   Type:                                                    Fixed
   Frequency [GHz]:                                         9.2
Pulse width pattern:
   Type:                                                    Fixed
   Pulse width [us]:                                        1.8
```

Figure 4.5 Input parameters for radar 2

Figure 4.8c) shows a selected time interval from Figure 4.8b) with pulses received by sensor 2 during 100 ms between TOA=21.0 s and TOA=21.1 s. We see that radars 2 and 3 have constant frequencies of 9.2 GHz and 9.6 GHz respectively, while radar 1 uses a switched frequency pattern of [6 x 9.40   6 x 9.42   6 x 9.44] GHz. The frequency measurements at sensor 2 have arbitrary errors with standard deviation 0.5 MHz.



```
%%%%%%%%%%%% Radar #3 %%%%%%%%%%%%
Start time (sec after scenario start):                  0
End time (sec after scenario start):                    30
Lobe pattern:
   Type:                                                sinc
   Main lobe opening angle [deg]:                       3
   Radar power at main lobe [dB]:                       5
   Radar power at back lobe (relative main lobe)[dB]:   -20
PRI pattern:
   Type:                                                Jitter
   PRI [ms]:                                            0.9
   Jitter, stddev [%]:                                  1
Rotation period:
   Type:                                                Constant
   Start angle relative x-axis [deg]:                   120
   Rotation period [s]:                                 3
Trajectory:
   Type:                                                StraightLine
   Start x-position [m]:                                7000
   Start y-position [m]:                                -5000
   Angle of trajectory relative x-axis [deg]:           70
   Speed along trajectory [m/s]:                        8
Frequency pattern:
   Type:                                                Fixed
   Frequency [GHz]:                                     9.6
Pulse width pattern:
   Type:                                                Switched
   Pulse width [us]:                                    1.42 1.44 1.46
   # repetitions [-]:                                   8    8    8
```

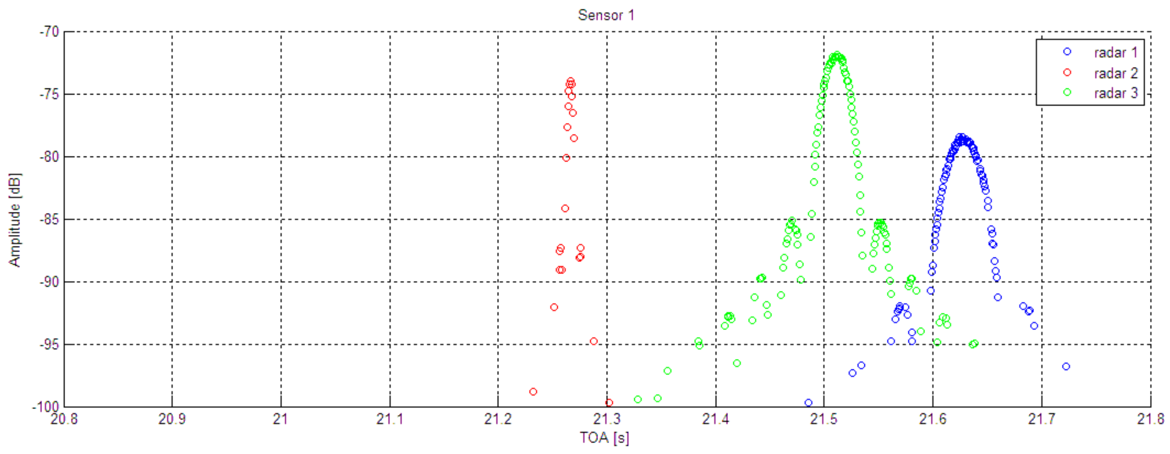
Figure 4.6   Input parameters for radar 3.

Figure 4.9 shows the PRI as a function of TOA for pulses received by sensors 1 and 2. We see that radar 2 has a constant PRI of 0.8 ms, while radar 3 has a jittered PRI of 0.9 ms with standard deviation 9 µs (1%). Radar 1 uses stagger PRI with PRIs of [0.67 0.68 0.69 0.70] ms. The PRI pattern for radar 1 appears clean and regular for the central part of the main lobe where all pulses are detected (between TOA=21.01 s and 21.06 s), see Figure 4.9c). For the side lobe (between TOA=21.08 s and 21.10 s) several pulses are missing, and the pattern becomes more disrupted.

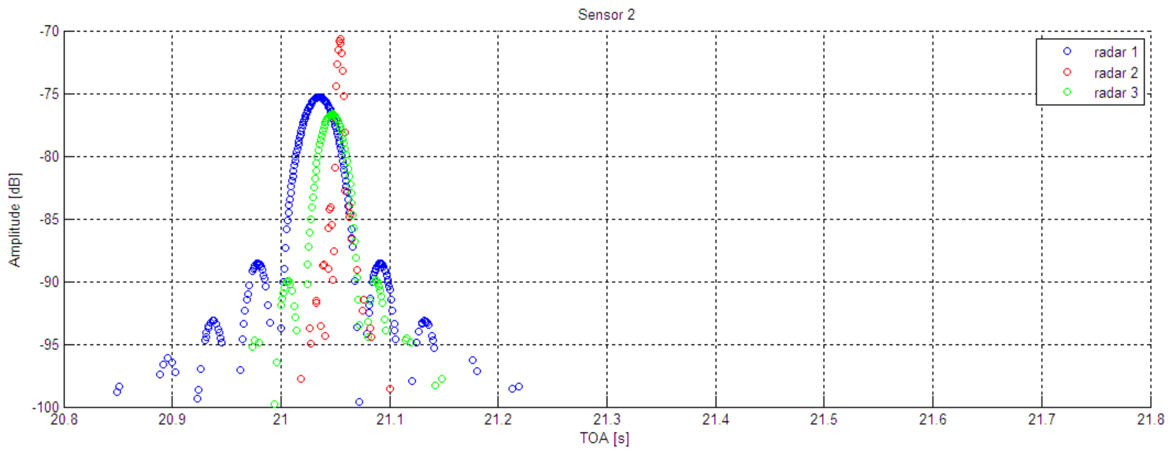
Figure 4.10 shows the pulse width as a function of TOA for pulses received by sensors 1 and 2. We see that radars 1 and 2 have constant pulse widths of 1.2 µs and 1.8 µs respectively. Radar 3 uses a switched pulse width pattern with pulse widths of [8 x 1.42   8 x 1.44   8 x 1.46] µs. The pulse width measurements have arbitrary errors of 1 ns (both sensors).

Figure 4.11 shows the angle of arrival as a function of TOA for pulses received by sensors 1 and 2. We see that the AOA is approximately -31°, 10°, and -54° for radars 1, 2, and 3 respectively, as measured by sensor 2 (Figure 4.11b and c). At sensor 1 the corresponding numbers are 54°, 48°, and 2° (Figure 4.11a). The measurement errors have standard deviation 1° at sensor 1 and 0.1° at sensor 2. This difference is clearly visible in the figure, where the spread in the measurements are seen to be much larger for sensor 1 (Figure 4.11a).

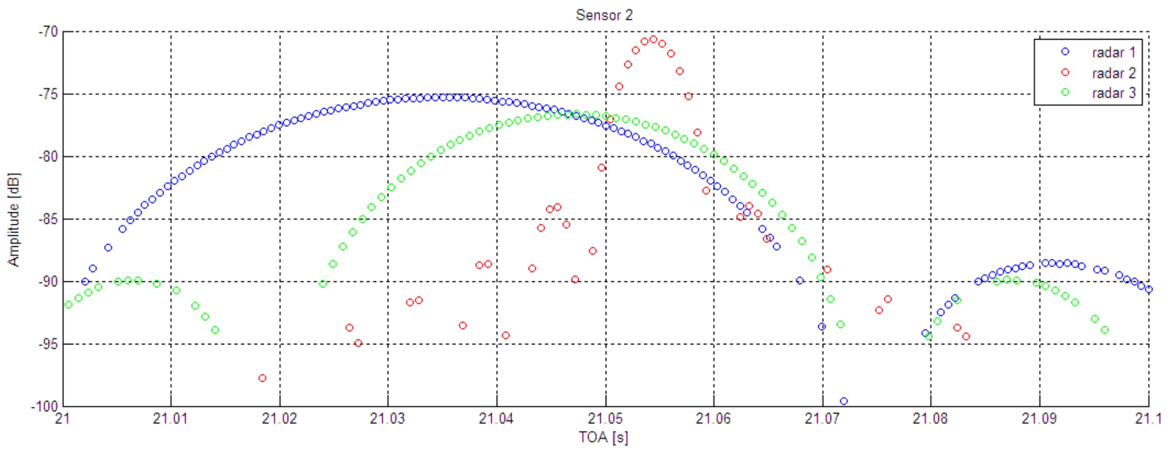
We see that for all parameters (amplitude, frequency, PRI, pulse width, AOA) the results look reasonably realistic. The PDW simulator could therefore be a valuable supplement to real data when developing and testing new pulse handling algorithms.



a)

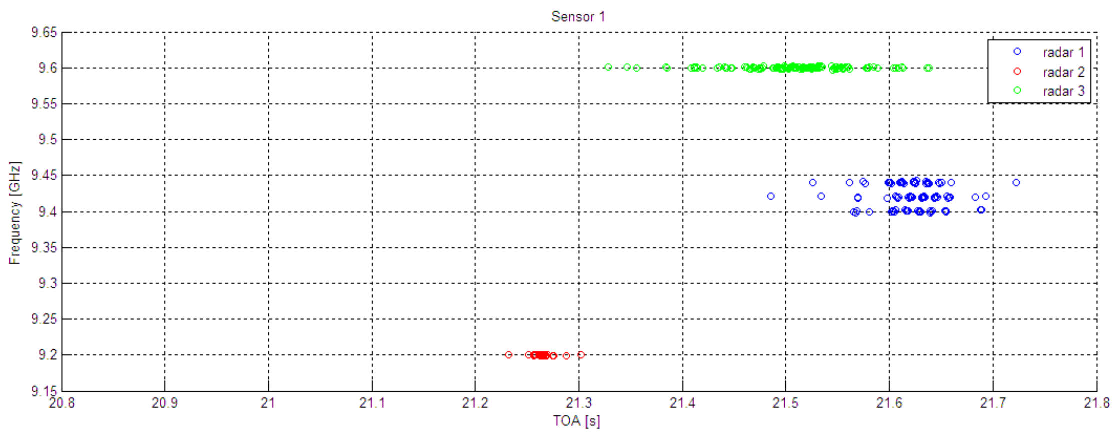


b)

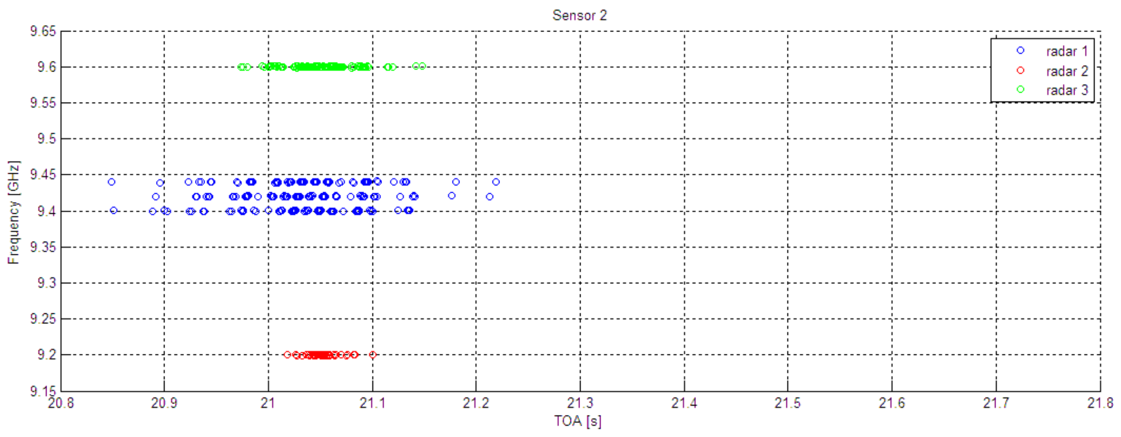


c)

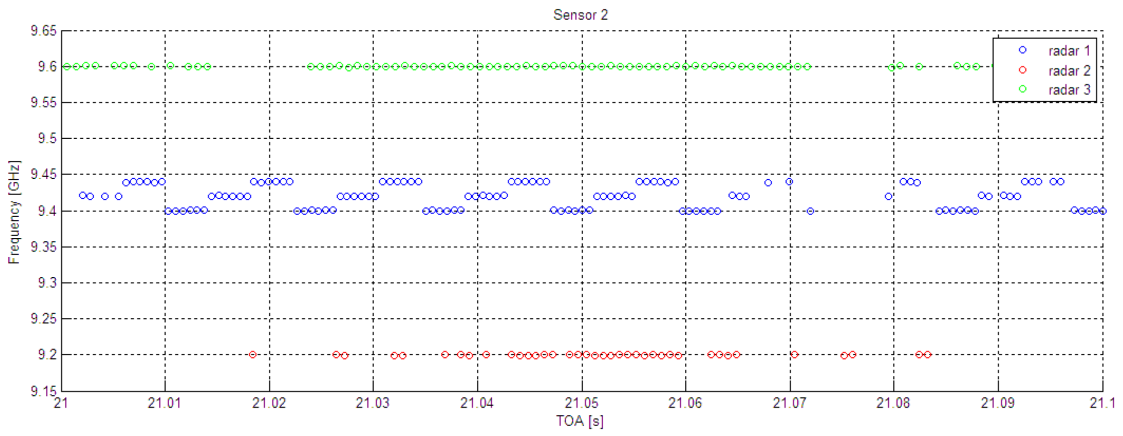
Figure 4.7 Amplitude as a function of TOA for pulses received during a time interval of 1 s by a) sensor 1 and b) sensor 2. In c) is shown a selected time interval from b) with pulses received by sensor 2 during 100 ms between TOA=21.0 s and TOA=21.1 s.



a)

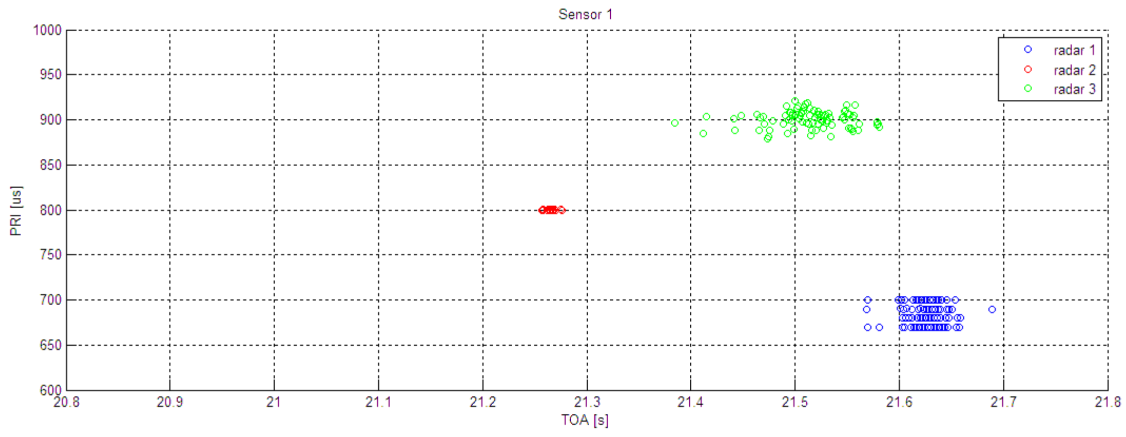


b)

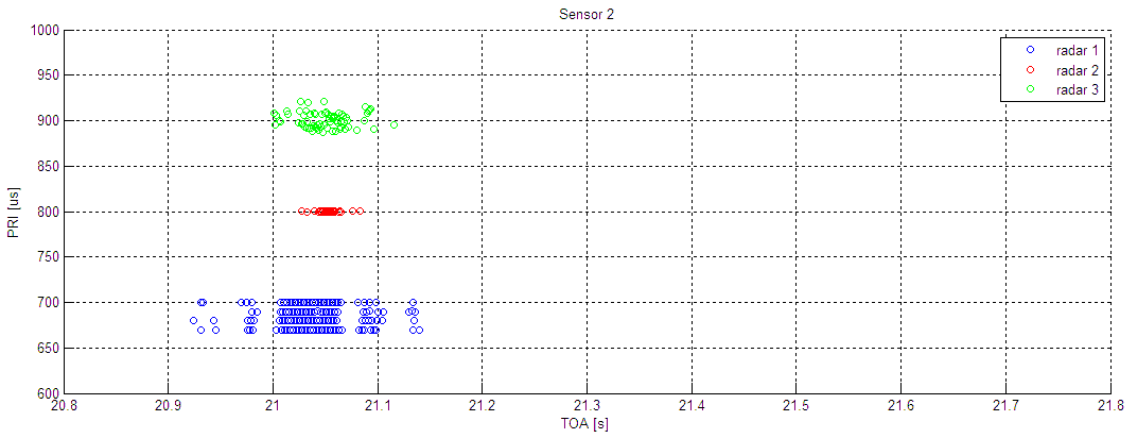


c)

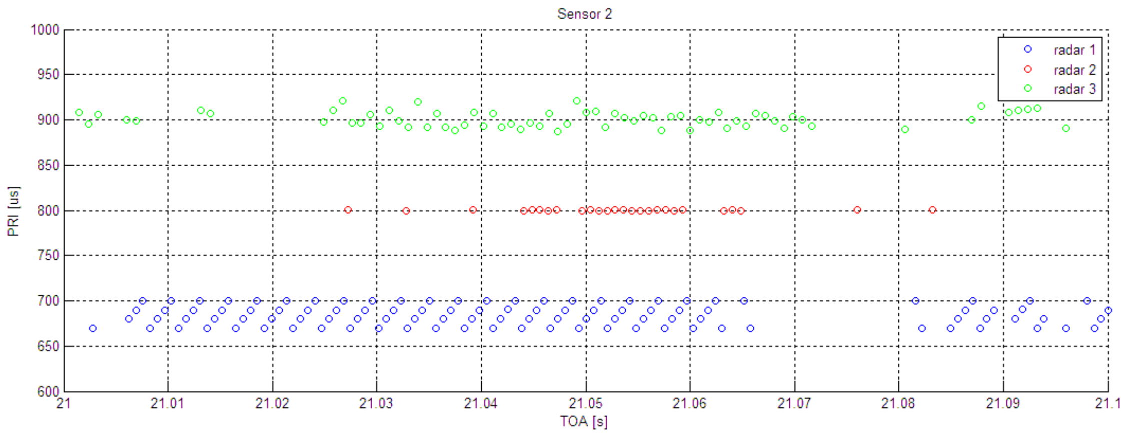
Figure 4.8 Frequency as a function of TOA for pulses received during a time interval of 1 s by a) sensor 1 and b) sensor 2. In c) is shown a selected time interval from b) with pulses received by sensor 2 during 100 ms between TOA=21.0 s and TOA=21.1 s.



a)

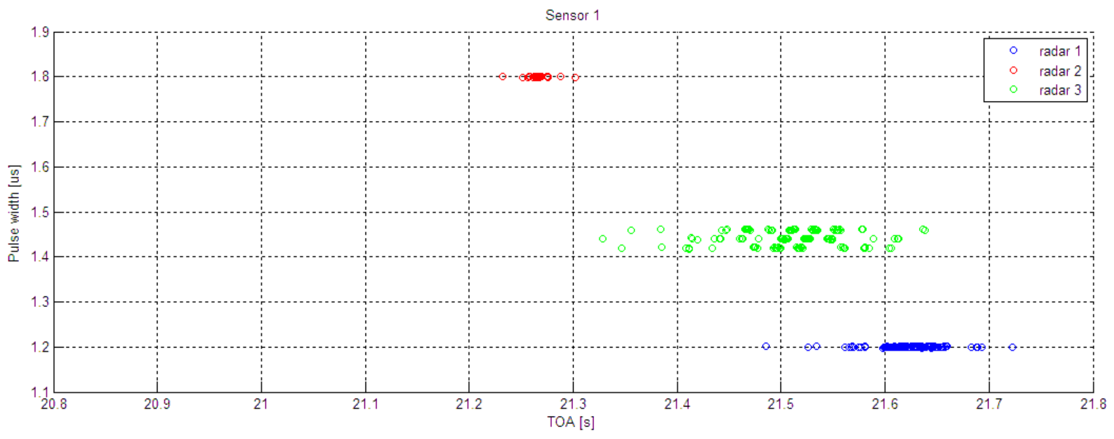


b)

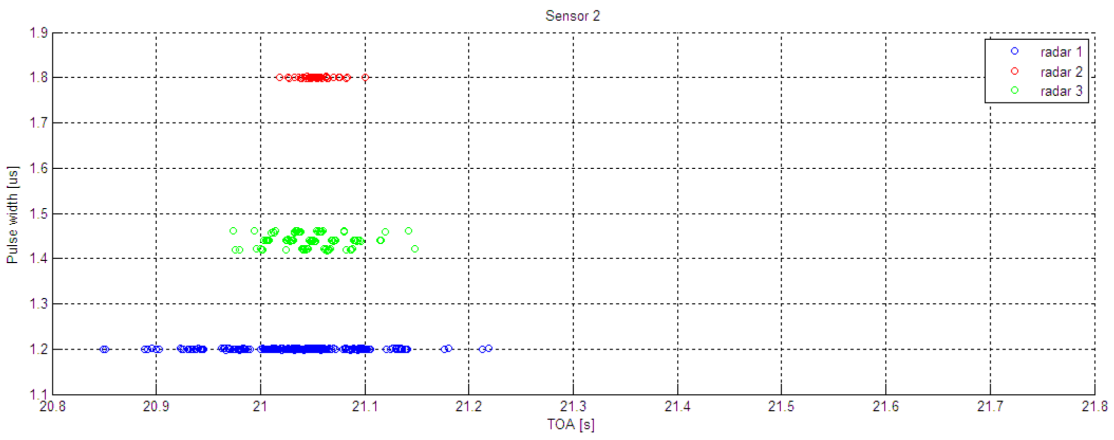


c)

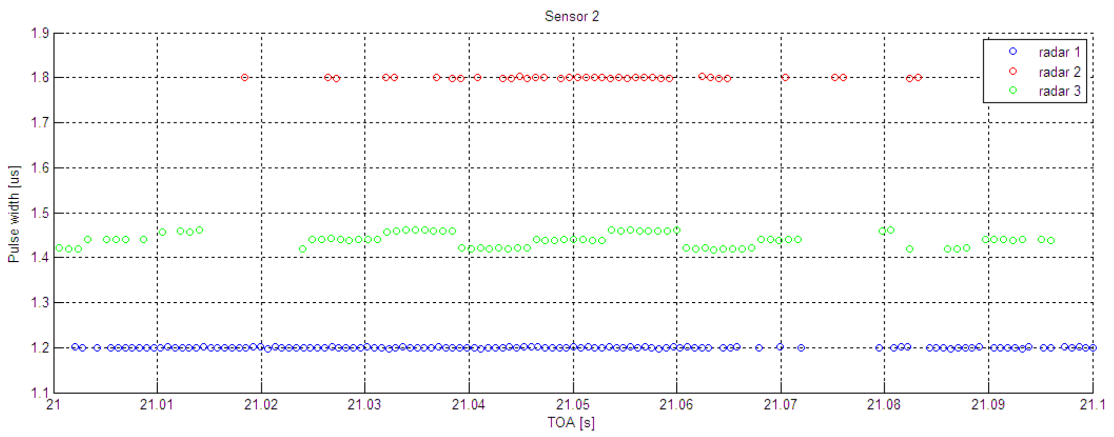
Figure 4.9 PRI as a function of TOA for pulses received during a time interval of 1 s by a) sensor 1 and b) sensor 2. In c) is shown a selected time interval from b) with pulses received by sensor 2 during 100 ms between TOA=21.0 s and TOA=21.1 s.



a)

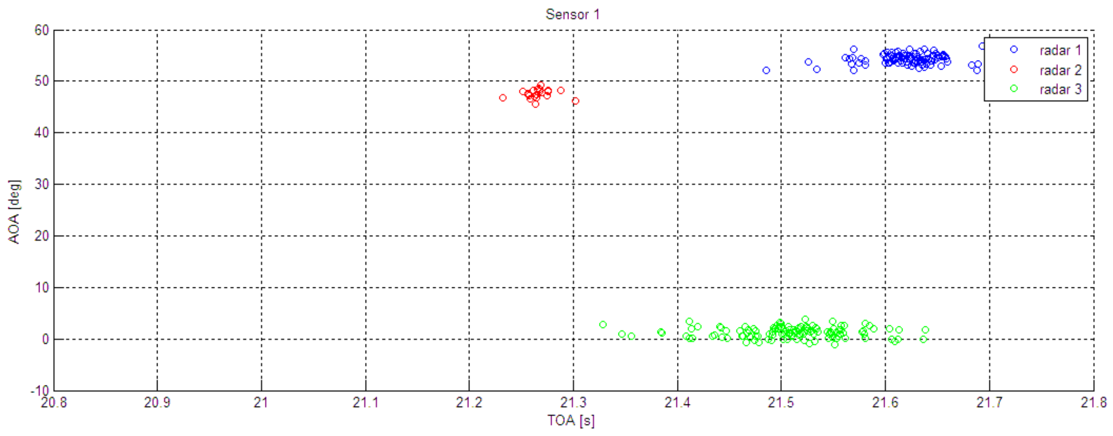


b)

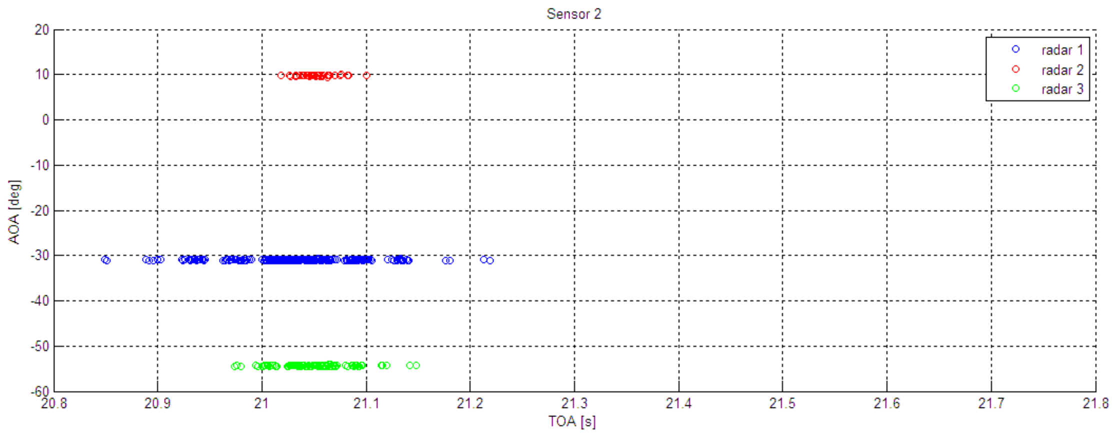


c)

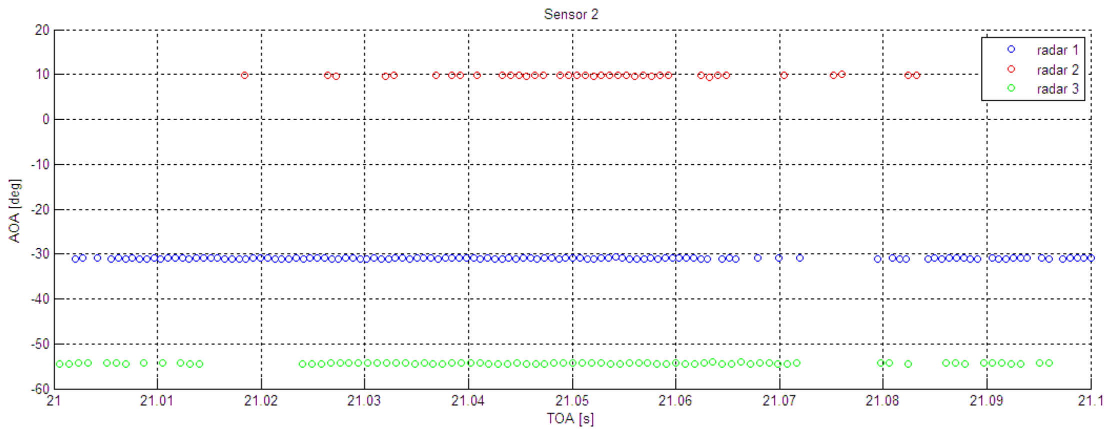
Figure 4.10 Pulse width as a function of TOA for pulses received during a time interval of 1 s by a) sensor 1 and b) sensor 2. In c) is shown a selected time interval from b) with pulses received by sensor 2 during 100 ms between TOA=21.0 s and TOA=21.1 s.



a)



b)



c)

Figure 4.11 AOA as a function of TOA for pulses received during a time interval of 1 s by a) sensor 1 and b) sensor 2. In c) is shown a selected time interval from b) with pulses received by sensor 2 during 100 ms between TOA=21.0 s and TOA=21.1 s.

5 An application – Testing LINE Deinterleaving

The goal of this chapter is to show how data generated by the PDW-simulator can be used efficiently to test and optimize the LINE deinterleaver. Testing a deinterleaver is an excellent application of simulated PDW data. It is particularly applicable to deinterleaving because a model solution to the deinterleaving problem is directly available, which is a rare luxury. This gives great opportunities to optimize, test and compare algorithms, which is discussed in further detail in Chapter 2 of [1].

5.1 The LINE deinterleaver

LINE (**LI**ttle **N**avigation Radar **E**SM) was an activity at FFI that culminated in the spring of 2012 at Unified Vision 2012 (UV12), in which two experiment sensors were used to track ships by their navigation radars at Ørlandet. This was accomplished by combining the difference in the rotation phase of the radar (based on the strength of the signal received at each ESM-sensor) with the difference in the time of arrival (TDOA) of pulses at the two ESM-sensors. The two methods gave several geographical curves (arcs and half hyperbolas), such that the radar would be located at an intersection between the curves.

The LINE deinterleaver (described in Appendix A of [1]) was developed, with the goal of deinterleaving data with the following constraints:

- The work will be focused on navigation radars
- Those radars tend to have a high degree of jitter
- Those radars frequently do not have additional PRI patterns, like stagger, switched PRI or sliding PRI
- The parameters available to the deinterleaver are TOA and amplitude.
- There are relatively few radars present at once.

A deinterleaver was developed, which takes as its input some number (typically 500) of consecutive pulses and then group together the pulses that came from the same radar and groups separately pulses from different radars. This was tested in a realistic scenario during UV12 with good results, measured by the utility and reliability that the deinterleaver provided to client programs.

5.2 Clustering

As discussed in [1], it is useful to consider deinterleaving in the more general context of clustering, and to develop generic evaluation criteria, rather than specialized criteria for deinterleaving. Here, the problem of clustering N different data points into some number of clusters is considered as a binary classification problem. Every data point may or may not be within the same cluster as each of the other data points. Hence, for every pair of data points, one may ask whether both points in that pair were grouped together or not, in the result. For each pair of points, this has either a positive or negative answer. One may ask the same question about every pair of points with respect to the model solution. This puts every pair of points into one of the following four categories[2]:

1. Correctly grouped together (True Positives, TP)
   - They are in the same result cluster and solution cluster
2. Incorrectly grouped together (False Positives, FP)
   - They are in the same result cluster, but different solution clusters
3. Incorrectly not grouped together (False Negatives, FN)
   - They are in different result clusters, but the same solution cluster
4. Correctly not grouped together (True Negatives, TN)
   - They are in different result clusters and solutions clusters.

Counting the number of pairs in each of the above categories, and associating these counts with the category labels, TP, FP, FN and TN, the following additional measures can be derived:

- TP_{Max}=TP+FN is the largest possible value of TP for a given problem.
- TN_{Max}=TN+FP is the largest possible value of TN for a given problem.
- Sensitivity = TP / TP_{Max}
- Specificity = TN / TN_{Max}
- Accuracy = (TP + TN) / (TP_{Max} + TN_{Max})

It is worth noting that, for any clustering problem, there is a trivial solution, grouping all inputs together, giving a sensitivity of 1 (optimal) and another trivial solution, assigning a separate cluster to each data point, giving a specificity of 1. Thus one cannot measure clustering well with only one of these criteria. One needs to either consider both or combine the criteria in some way. Accuracy is one such combination, which makes the assumption that false positives/negatives are equally bad. In this chapter, specificity and sensitivity will be observed together.

Counting all the pairs of points explicitly, is a tedious process that requires the algorithm to visit some point at least $N \times (N-1)/2$ times. Calculating TP, FP, FN and TN can be done much more efficiently, as described in [1].

5.3 Trying the LINE deinterleaver on simulated data

An example scenario of PDW-simulations with three radars and two sensors, was given in Chapter 4, and the LINE deinterleaver will now be applied to a part of the input from that scenario. In particular, the data from sensor 1 will be studied. 67885 pulses were received at sensor 1. The value 220 was added to all amplitudes to make them all positive, as required by the Line PDW data type, in which amplitude was specified as an 8 bit unsigned integer. For similar reasons, the TOA values were scaled up by a factor of 10000, although these values have been

[2] The rest of Section 5.2 consists mostly of direct quotations from [1].

scaled back in the data plots. The deinterleaver works best on relatively small data sets at a time, so the focus will be on the first 5000 pulses, which are shown in Figure 5.1.

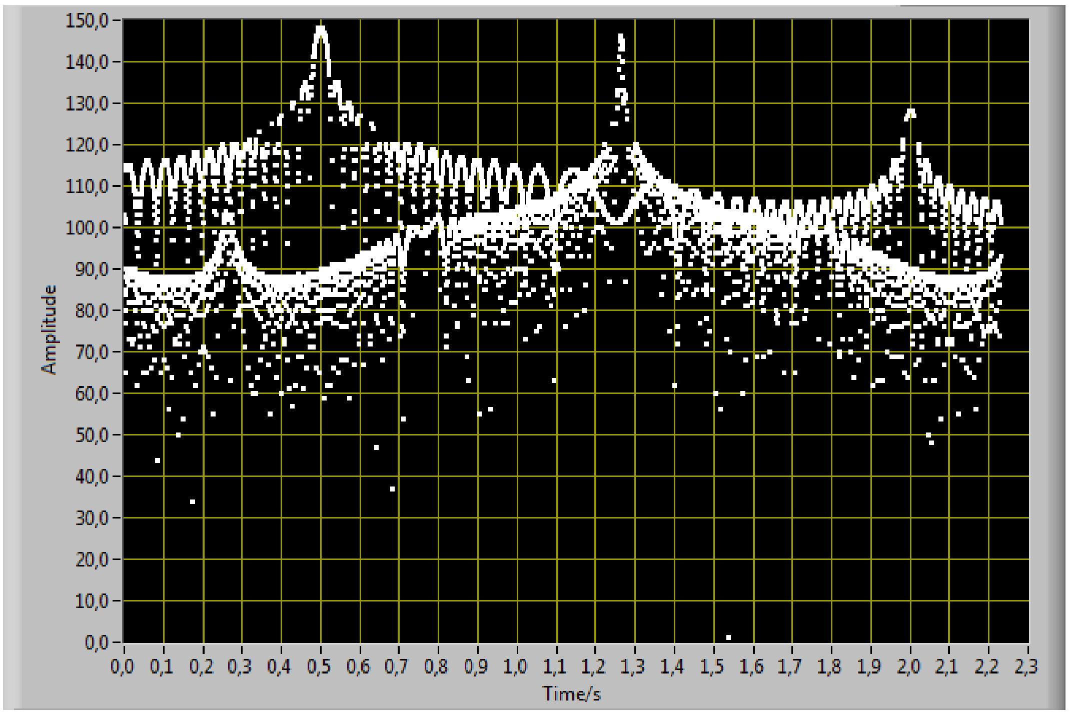


Figure 5.1 5000 pulses of interleaved data from the PDW-simulator.

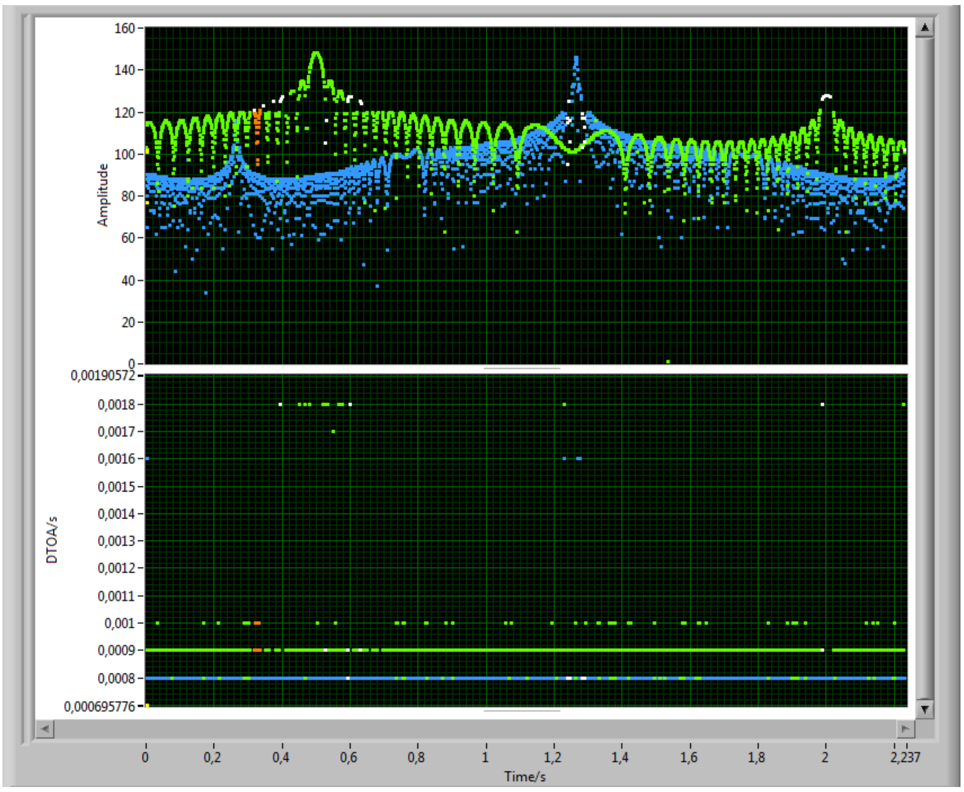


Figure 5.2 A first deinterleaving result for the inputs shown in Figure 5.1. DTOA in the figure is the same as ΔTOA.

When applying the LINE deinterleaving algorithm with some suitably chosen parameters, as described in Appendix A of [1], to the above data, it gives the output shown in Figure 5.2.

Visual inspection of the tendencies in the Time-Amplitude part of the plot (top part of the figure) suggests that most pulses have indeed been grouped together correctly, although the dash of orange points should have probably been grouped with the green points. However, such visual inspection does not exploit the luxury of having a model solution, as provided by the PDW-simulator.

Measuring the specificity and sensitivity of the deinterleaving result in Figure 5.2 gives the approximate values 0.992 and 0.970, respectively, meaning that 99.2% of all pairs that should have been grouped together were grouped together and 97.0% of all pairs that should not have been grouped together were not grouped together.

Having laid the ground work for assessing results, one can make things a little more interesting by modifying free algorithm parameters. One important parameter is the Relative Jitter Tolerance, J, which determines how much the ΔTOA[3] of consecutive pulses is allowed to vary within a group. Setting this parameter to 0 would require the ΔTOA to be exactly the same throughout each group. Setting it to 1 would allow variation of 100%. The optimal value is somewhere between these two extremes, but may be different for each dataset. Figure 5.3 shows the specificity and

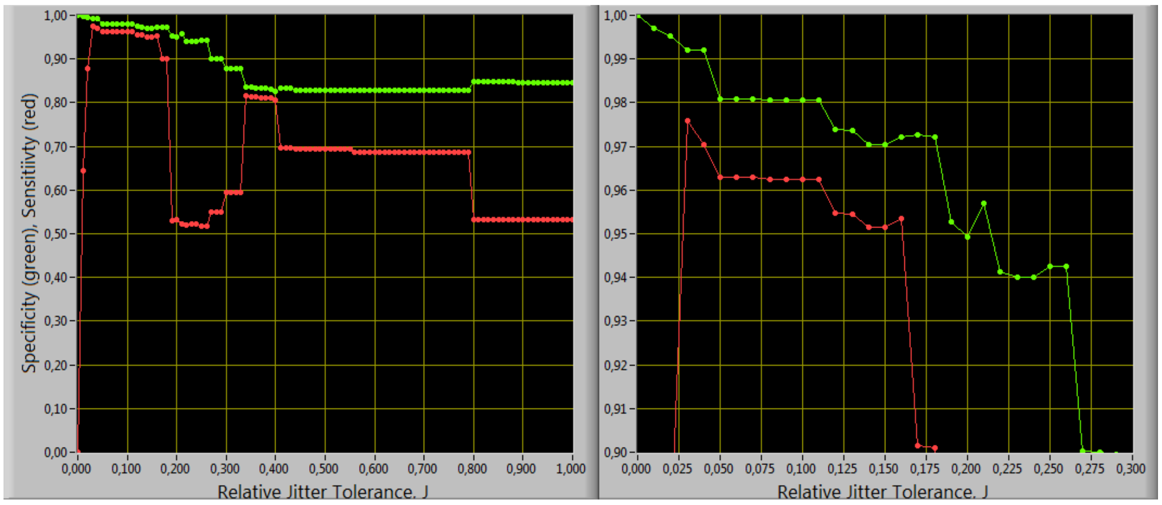


Figure 5.3 Specificity and sensitivity, measured for deinterleaving with different J-values. The best values for optimizing (maximizing) both criteria seem to be in the approximate range 0.01-0.20.

[3] ΔTOA of two consecutive pulses is the positive difference between the TOA values of the two pulses. If the two pulses come from the same radar, then, provided all emitted pulses were detected by the sensor, ΔTOA is the PRI of that radar at that point in time. Note that this is a different meaning of ΔTOA, compared with that of the previous chapters, where ΔTOA denoted the error in the measured TOA. The letter Δ is commonly used in mathematics to denote a value change or difference. Sticking with this convention in these two different contexts seems superior to introducing a new identifier just to make them look different. The current usage is exclusive to this chapter.

sensitivity, given J values uniformly spread between 0 and 1. Unsurprisingly, J=0 gives excellent specificity but terrible sensitivity. Since the ΔTOA varies a little throughout the data, J=0 cannot group any data together.

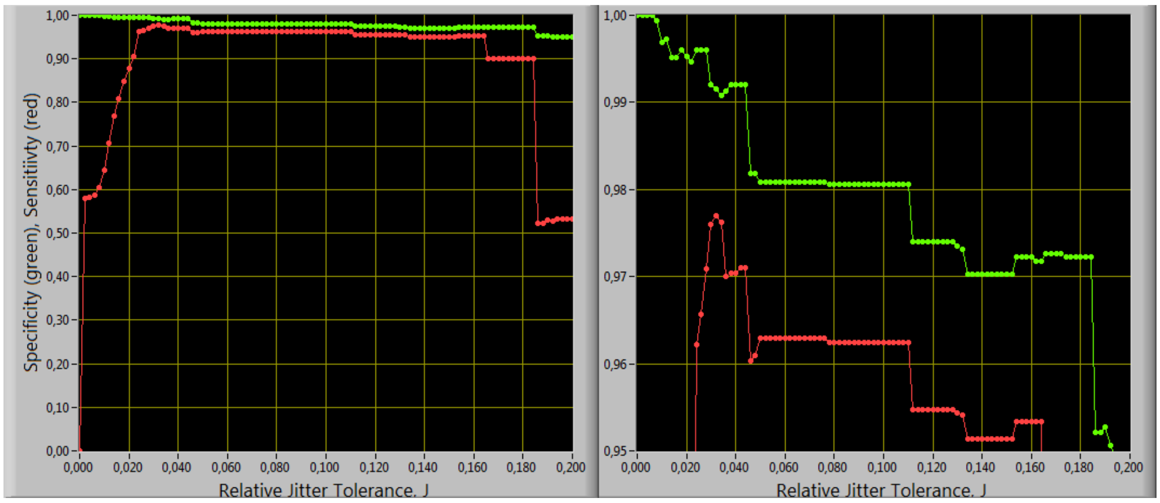


Figure 5.4 Looking closer at the interval J ∊ (0.01, 0.20), although the differences are becoming rather small within most of the interval, there is definitely a close to global peak for both curves at a little more than 0.03.

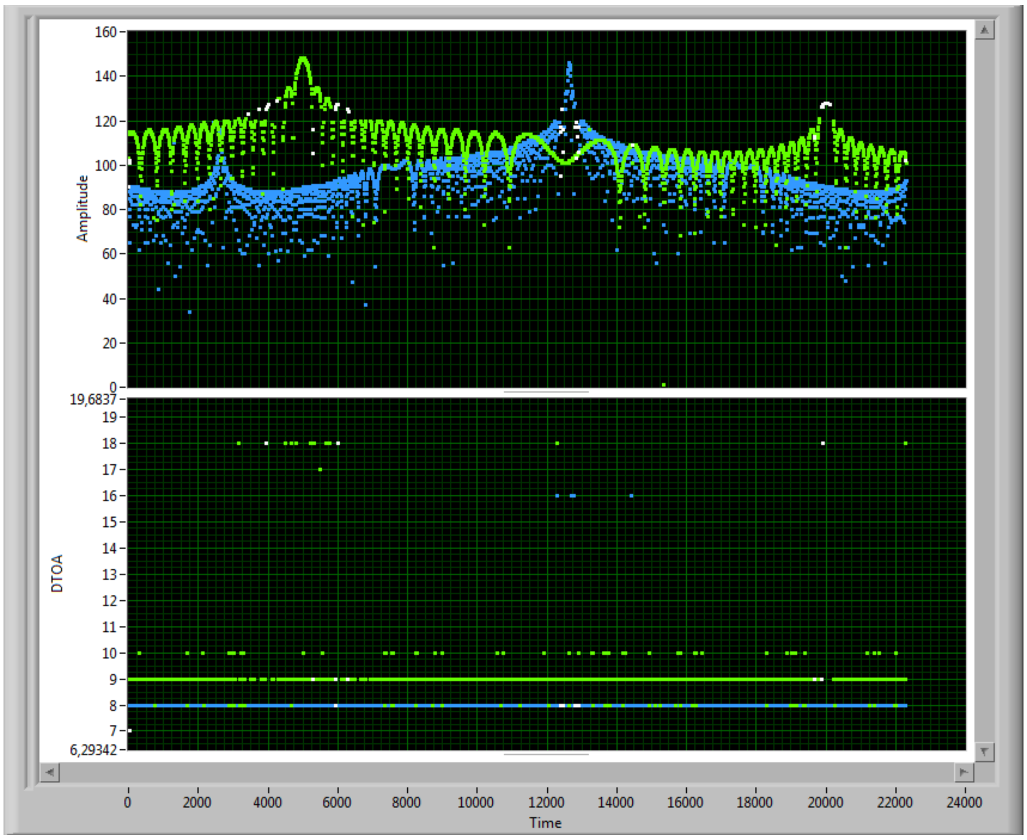


Figure 5.5 An optimal result, with specificity=0.992 and sensitivity=0.977 was found at J=0.031. The result also gave an improved visual result, as seen when comparing this figure to Figure 5.2.

The specificity drops pretty rapidly as the J is increased, whilst the sensitivity has a global peak within J ∈ (0.01, 0.20). Figure 5.4 shows how the specificity and sensitivity vary in that interval.

It seems that the peak in both specificity and sensitivity may be within the interval J ∈ (0.028, 0.034), thus the process is repeated for that interval, and J=0.031 gives a good compromise between specificity=0.992 and sensitivity=0.977. This also gives an improved visual result, shown in Figure 5.5, where the orange pulses from Figure 5.2 have been grouped with the green pulses, as they should.

5.4 Generalizing to different data from the same PDW-simulation

The model solution to the deinterleaving has helped optimize the parameter J, for these particular data. It is now interesting to see if the optimized parameter remains a good choice for different data from the same PDW-simulation. So far, the pulses considered have been pulses 0 through 5000 in the data. Trying J=0.031 on pulses 5001 through 10000 gives specificity=0.996 and sensitivity=0.981, i.e. an even better result than for the first 5000 pulses. Trying the same for the next 5000 pulses, and so on (progressing 5000 pulses per iteration) gives sensitivities and specificities as given in Figure 5.6. The specificity does not vary much. It is mostly between 0.99 and 1, once dropping to about 0.98. The sensitivity varies much more. In most of the cases (11 out of 14) it is in the range (0.94, 1), but in the remaining cases, it is around 0.84 or (in one case) 0.78.

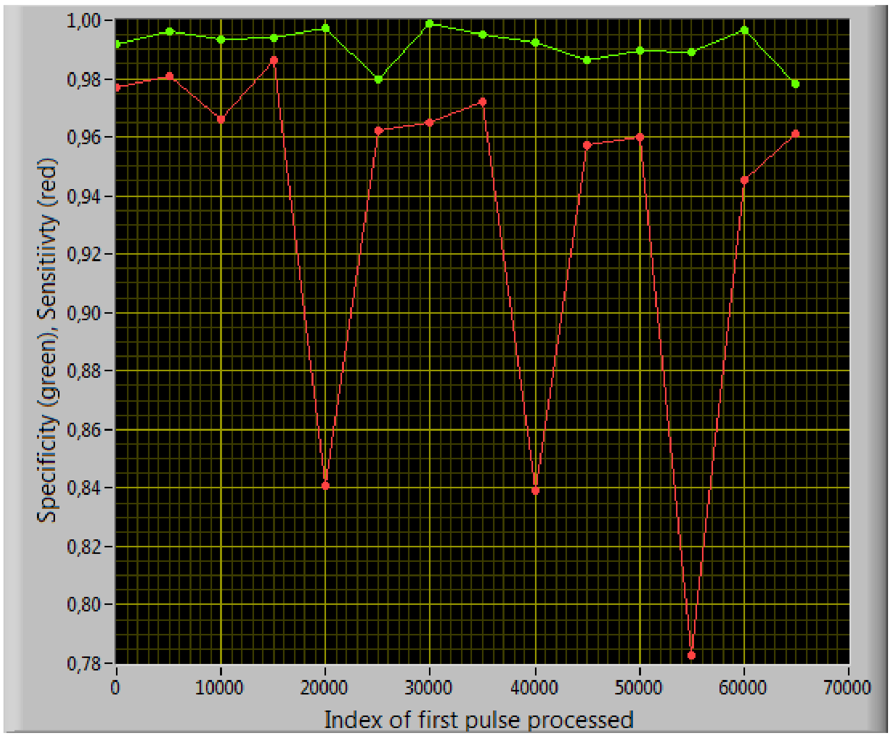


Figure 5.6 Specificity and sensitivity after deinterleaving different chunks of pulses from the current data set. 5000 pulses were processed, deinterleaved and evaluated at a time, before progressing to the next 5000 pulses, continuing this process for a total of 50000 pulses.

It is worth taking a look at the cases that fail, to find out why so many points fail to get grouped together. Figure 5.7 shows the result from deinterleaving pulses 55000 through 60000, which gave the lowest sensitivity of 0.78. The figure shows that about 800 pulses (the purple ones) have been put in a separate group, even though they should clearly be grouped with the green pulses, judging from the continuity both in amplitude and ΔTOA.

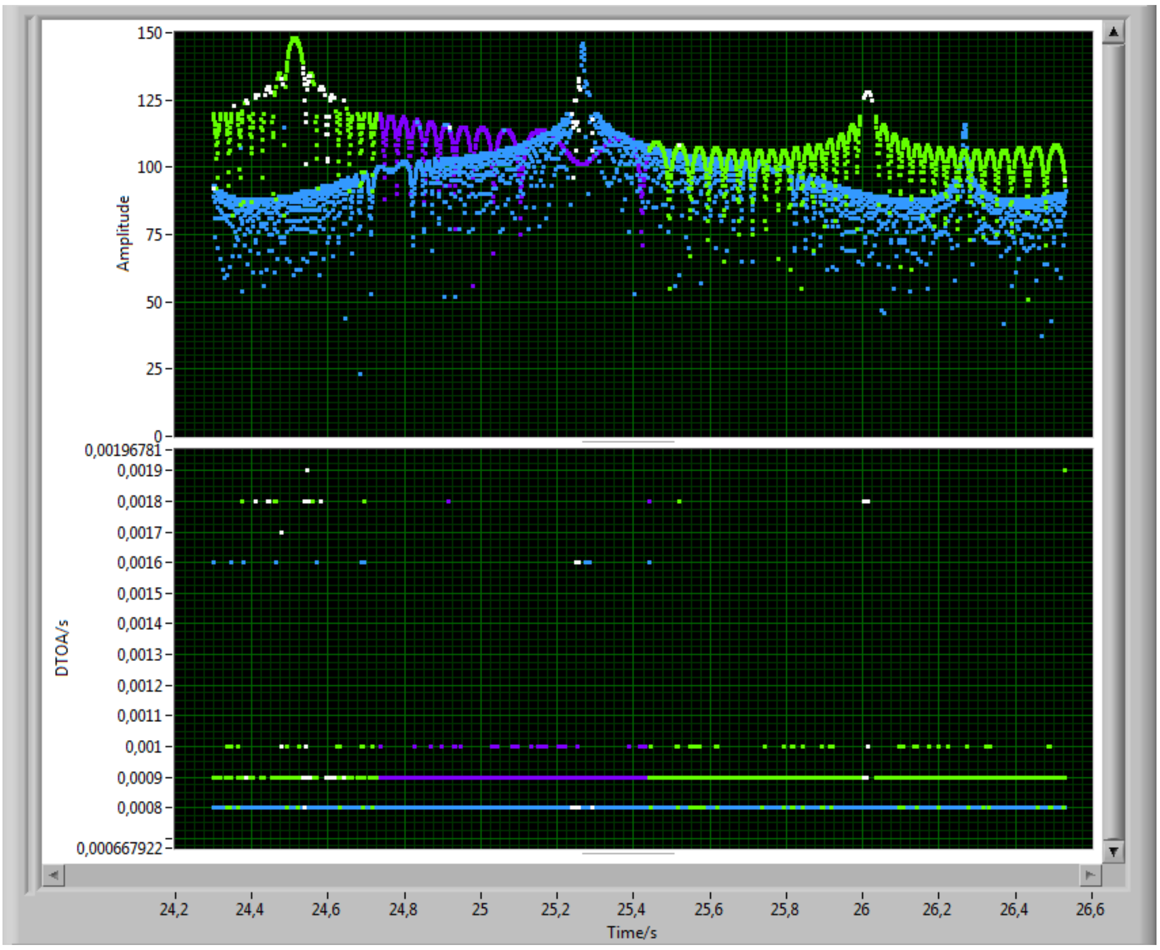


Figure 5.7 Pulses 55000 through 60000 were deinterleaved with the lowest performance and a sensitivity of approximately 0.78.

Now may be a good moment to reevaluate the performance measures. The deinterleaving error in Figure 5.7 (the purple pulses that should be green) seems a relatively small mistake, being simply a discontinuity in the cluster membership in two places along the time axis. If the purple pulses were distributed arbitrarily, then the sensitivity and specificity would remain exactly the same, even though this would probably lead to greater problems for a client program. Thus, depending on the application of the deinterleaving result, these performance measures may not always be appropriate. It will not be discussed further in this context, but is worth keeping in mind when evaluating clustering results.

5.5 Beyond this example

So far, the deinterleaver has been applied to PDW simulator data in order to optimize one parameter with respect to a particular data set. An extension of this would be to optimize the same parameter with respect to different data sets, for example a set of representative datasets that are relevant in preparation for certain real scenarios.

After that, one may change the focus to the other parameters of the deinterleaving algorithm. The LINE deinterleaving algorithm is described in [1], and so far the parameter J (Relative Jitter Tolerance) of this algorithm has been considered. One may do similar work with the parameters M (Missing Pulses Tolerance), S (Pulses Skipped) and N (Sequence Length), or a combination of these. Much of the work has also been manual so far, in part because no effort has been made towards combining the two quality measures. A simple way to combine specificity and sensitivity is to optimize the sum, or a weighted sum. Alternatively, one could optimize the accuracy. Having decided which measure to use, one can start automating searches for optimal parameter combinations.

6 Summary

We have developed a first version of a Pulse Descriptor Word (PDW) simulator with a user-friendly interface in Matlab. The simulator produces PDWs with information about time of arrival (TOA), amplitude, frequency, pulse width, and angle of arrival (AOA), based on information about scene geometry and radar and sensor properties.

Applying the PDW simulator to a test scenario has shown that the simulator produces quite realistic output. The PDW simulator was also used to test a deinterleaving algorithm – the LINE Deinterleaver. The benefits of controlling the input scenario, as well as knowing the absolute truth about which pulse belongs to which radar, was clearly seen for this type of application. We conclude that the PDW simulator could be a valuable additional tool when developing and testing pulse handling algorithms.

The current version of the PDW simulator software simulates rotating radars in 2 dimensions on a flat Earth. Future versions of the software could be expanded to operate in 3 dimensions and to use an elliptical Earth. Obstacles could be added to the scenario, and integration of the software with 3D maps could open up for the possibility to create realistic geographic scenarios. Non-rotating radars and additional parameters and measurements such as crystal frequency and Doppler could also be included. The user interface could be upgraded to further facilitate the use of the PDW simulator.

Appendix A PDW simulator algorithm

The algorithm for the PDW simulator can be summarized as follows:

<u>Procedure GeneratePDWs:</u>

1) For each radar #m:
   ProcessRadar(#m)

2) Arrange the pulses received (from several radars) at each sensor, in order according to their time of arrival.

3) Write the following pulse information to file (in the form of separate tables for each parameter and each sensor, see Section 3.2):

   a. TOA
   b. Amplitude
   c. Frequency
   d. Pulse width
   e. AOA
   f. Radar number
   g. Pulse number

4) Write radar and sensor positions to file.

<u>Procedure ProcessRadar(#m):</u>

1) Generate table with pulse emitting times for radar #*m*, based on the given PRI type and values (Section 2.2.3).

2) Generate tables with *x*- and *y*- positions for radar #*m* at pulse emitting times, based on the given information about the radar movement (Section 2.1).

3) Generate table with angles between the radar main lobe pointing direction and the *x*-axis at pulse emitting times for radar #*m*, based on the given information about the rotation period and start pointing direction for the main lobe (Section 2.2.1).

4) For each sensor #n:
   ProcessSensor(#n)

Procedure ProcessSensor(#n):

1) Generate tables with *x*- and *y*- positions for sensor #*n* at pulse emitting times, based on the given information about the sensor's movements (Section 2.1).

2) Generate table with angles between the radar-sensor line and the *x*-axis at pulse emitting times for radar #*m* and sensor #*n*.

3) Generate table with angles between the radar-sensor line and the radar main lobe at pulse emitting times for radar #*m* and sensor #*n*.

4) Generate table with the distance between radar #*m* and sensor #*n* at pulse emitting times.

5) Generate table with the amplitude of the received pulses at sensor #*n* for all pulses from radar #*m* (Equation (2.15)).

6) Determine which pulses from radar #*m* that are detected at sensor #*n*, based on the given detection probabilities for the sensor (Section 2.3.1).

7) Determine the time of arrival (TOA) for pulses from radar #*m* at sensor #*n* (Equation (2.19)).

8) Determine the frequency for pulses from radar #*m* at sensor #*n* (Equation (2.21)).

9) Determine the pulse width for pulses from radar #*m* at sensor #*n* (Equation (2.22)).

10) Determine the angle of arrival (AOA) for pulses from radar #*m* at sensor #*n* (Equation (2.23)).

References

[1] Opland E J (2013): “Clustering Evaluation for Deinterleaving”, FFI-rapport 2013/00567