SimPel User Manual

Version 1.0.2

by

Stephan Rein

Contents

Preface

SimPel has been developed in the group of Prof. Dr. Stefan Weber at the University of Freiburg, Freiburg im Breisgau, Germany, during the last couple of years in course of developing GloPel, a software for advanced global analysis of PELDOR/DEER spectra.

This manual describes the general usage of SimPel and gives an overview of the maths behind that has been implemented.

If you use SimPel for your own research and publish results accordingly, please give credits citing the appropriate reference:

> Stephan Rein, Philipp Lewe, Susana L. Andrade, Sylwia Kacprzak, Stefan Weber
> Global analysis of complex PELDOR time traces
> *Journal of Magnetic Resonance*, submitted

A number of people have helped shaping SimPel and the ideas behind. First and foremost, Dr. Sylwia Kacprzak (now Bruker Biospin) was for years the driving force behind SimPel due to the need for advanced analysis of PELDOR/DEER data with limited signal-to-noise ratio. Dr. Till Biskup contributed ideas for programming and details of the implementation that make all the difference between a program useful for a larger audience and a simple in-house solution. And finally, without Prof. Dr. Stefan Weber and his continuous and continuing support, this work would not have been possible.

Freiburg, March 2018

Stephan Rein

1 Disclaimer

The software is provided "as is", without warranty of any kind, express or implied, including but not limited to the warranties of merchantability, fitness for a particular purpose and noninfringement. SimPel is distributed under a GPLv3 license.

2 Introduction

2.1 General Description

SimPel is a Python3-based simulation program for high-performance simulations of PELDOR/DEER traces. SimPel provides a user-friendly GUI based on the PyQt5 GUI framework.

Up to five Gaussian distributions with independent linear coefficients and standard deviations can be used for creating a distance distribution. The user-defined distance distribution is used to calculate the PELDOR/DEER trace. Many simulation options are available. In addition to the pure dipolar evolution signal of the PELDOR/DEER trace, a background function can be added as well as Gaussian (white) noise. Simulations can be saved and automatic figure export is supported (formats: png, tif, pdf, ...).

Configuration settings for simulation and plotting options can be adjusted, allowing the user, e.g. to create figures with own preferences.

SimPel can be used to systematically explore the impact of changes of the distance distribution on the PELDOR/DEER trace. Additionally, SimPel can be used to compare experimental time traces with simulations.

The Bruker BES3T file format (.DTA, .DSC) used by Xepr and Elexsys spectrometers is supported as well as text files for loading experimental data. Furthermore, SimPel can be used to create high quality figures for simulated time traces and distance distributions.

For PELDOR/DEER analysis, we recommend the program GloPel, which is also freely available and can be downloaded from `https://www.radicals.uni-freiburg.de/de/software/simpel`.

2.2 License

SimPel is made available free of charge and open source. SimPel is distributed under a GPLv3 (GNU General Public License). The full text of this license can be found in the file `LICENSE.txt` in the main directory of the SimPel source code.

2.3 Requirements

SimPel was developed using Python 3, and whereas there is good chance to get it to run with Python 2.x, this has not been tested and is not guaranteed.

Besides a reasonably recent Python installation, only some standard packages are required that should be either pre-installed on your system or fairly easy to install, namely:

- NumPy
- SciPy
- Matplotlib $\geq$ 2.0.0
- PyQt5

2.4 Installation Instructions

SimPel is available free of charge and open source (see license). We recommend everybody with a recent Python installation on their computers to download the source code and use it. In this case, no real installation is necessary, and starting SimPel is as simple as typing the following command into a terminal:

```
python3 SimPel.py
```

However, SimPel can be downloaded as executable binary for Windows, macOS and several Linux flavors.It was tested on various systems including Windows 7 (32-bit and 64-bit architecture), Windows 10 (32-bit and 64-bit architecture), macOS 10.12.6 (Sierra), and Ubuntu 16.04.

Please note that these versions are much harder to debug if something goes wrong, therefore, they are provided as is without any further support.

A note on how these binaries have been created: Thanks to PyInstaller, it is rather straightforward to create binary distributions of Python programs, both as directories with all the necessary files contained as well as single-file executables. Similar to cx_freeze, PyInstaller allows to define additional options in a "spec" file. This file is included in the sources available. To get a one-file binary, it was called simply as follows:

```
pyinstaller --onefile --noconsole setup.spec
```

1. Binaries for Linux have been built with Ubuntu 16.04 (Xenial Xerus) and a Python installation consisting of Python 3.5, matplotlib 2.1.1, PyQt 5.8.2, NumPy 1.13.3, SciPy 0.17.0

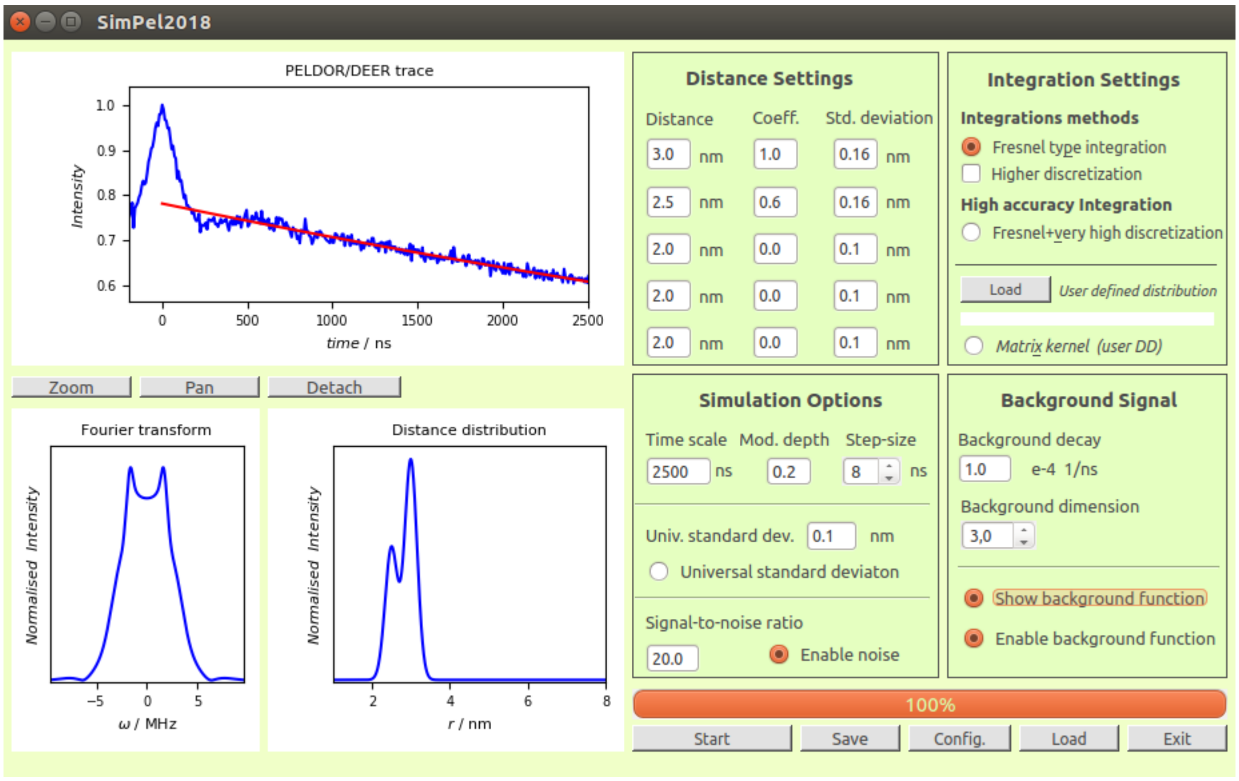
2. Binaries for macOS have been built with macOS 10.12.6 (Sierra) and a Python installation consisting of Python 3.5, matplotlib 2.0.0, PyQt 5.8.2

3. Binaries for Windows have been built with Windows7 (32-bit architecture) and a Python installation consisting of Python 3.5, matplotlib 2.1.2, NumPy (+MKL) 1.11.3, SciPy 0.19.0, PyQt 5.8.2

3 Using SimPel

3.1 Basic Options

When starting SimPel, the main GUI (Figure 1) opens. Default settings for all parameters are loaded when starting the program. The main GUI window as well as all additional windows are resizable and can be adjusted by the user.


SimPel2018

Distance Settings

Distance	Coeff.	Std. deviation
3.0 nm	1.0	0.16 nm
2.5 nm	0.6	0.16 nm
2.0 nm	0.0	0.1 nm
2.0 nm	0.0	0.1 nm
2.0 nm	0.0	0.1 nm

Integration Settings

Integrations methods
● Fresnel type integration
☐ Higher discretization
High accuracy Integration
○ Fresnel+very high discretization
Load *User defined distribution*
○ *Matrix kernel (user DD)*

Simulation Options

Time scale 2500 ns | Mod. depth 0.2 | Step-size 8 ns
Univ. standard dev. 0.1 nm
○ Universal standard deviaton
Signal-to-noise ratio 20.0
● Enable noise

Background Signal

Background decay 1.0 e-4 1/ns
Background dimension 3,0
● Show background function
● Enable background function

100%

Figure 1: Main GUI window of SimPel running on a Linux system (Ubuntu 16.04). Simulation of a bimodal distance distribution is shown. A background function (shown in red) was enabled. White noise was added to the simulated time trace.

Parameters such as central distances $r_{0,i}$, standard deviations σ_i, or linear coefficients c_i for up to five Gaussian functions can be set by the user. Additional simulation options are the time scale ($t_{\max}$), the PELDOR/DEER step-size and the modulation depth λ. The PELDOR trace is calculated according to Equation 1. A background function can be added to the PELDOR/DEER trace. The background dimension can be chosen between 1 and 6 in steps of 0.1. The background decay constant can be defined by the user. The background function is calculated according to Equation 4. White noise can be added by enabling the corresponding button. The signal-to-noise ratio can be given by the user. The simulation is carried out by pressing the “Start” button.

A user-defined distance can be loaded. From this distance distribution the PELDOR/DEER trace is calculated using a kernel matrix according to Equation 3. The distance distribution has to be provided as text file.

Experimental PELDOR/DEER time traces can be loaded and compared to simulations. SimPel accepts either a text-file format or a Bruker BES3T file format (.DTA, .DSC). Example for a comparison between an experimental dataset and a simulation is shown in Figure 2

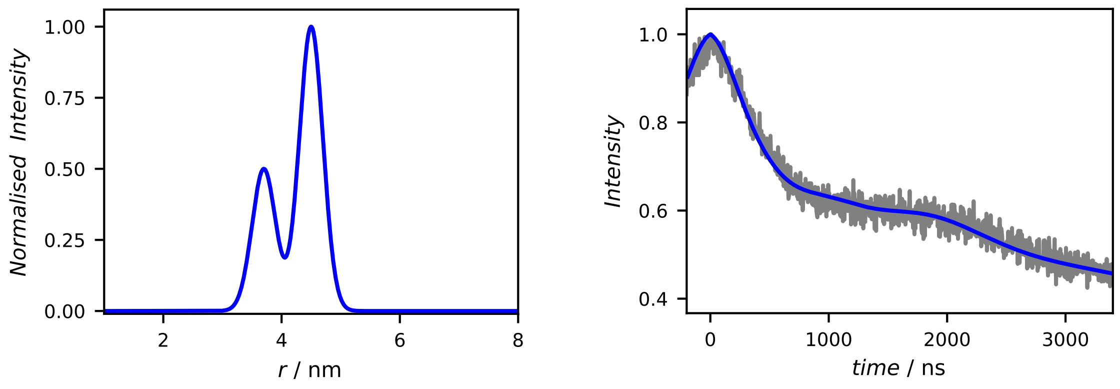


Figure 2: Comparison of a simulation (blue) and an experimental time trace (gray). SimPel can be used to get a first impression about the distance distribution of an experimental time trace. For detailed analysis, we recommend to use the program GloPel (`www.glopel.de`).

3.2 Integration Settings

The standard integration method of SimPel uses:

$$k = 19 + (\mathrm{round}(100\sigma_i)) + (20 + (\mathrm{round}(100\sigma_i))\%2$$

points for the discretization of a Gaussian function f_i in the interval $[r_{0,i} - 3\sigma_i\ ,\ r_{0,i} + 3\sigma_i]$. This is sufficient for most time traces. However, for some cases, especially if broad distance distributions are present in the small distance region, this is not sufficient and simulation noise is observed (Figure 3).

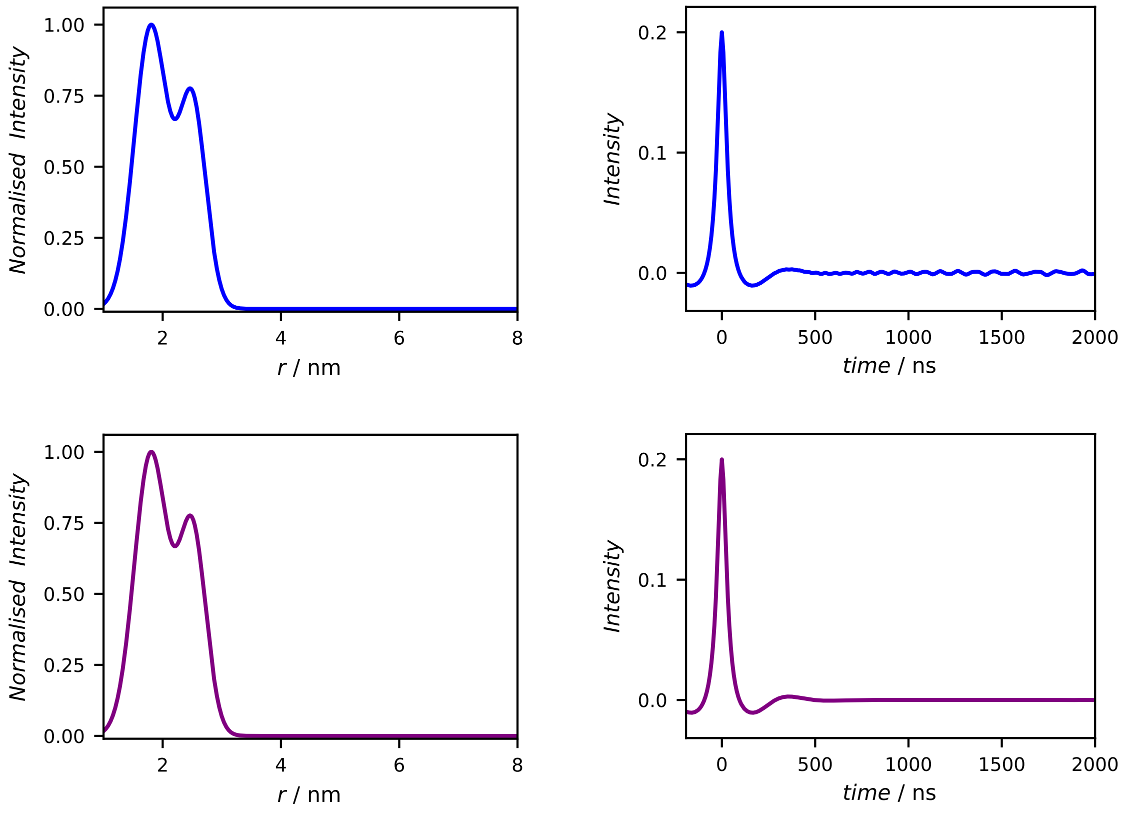
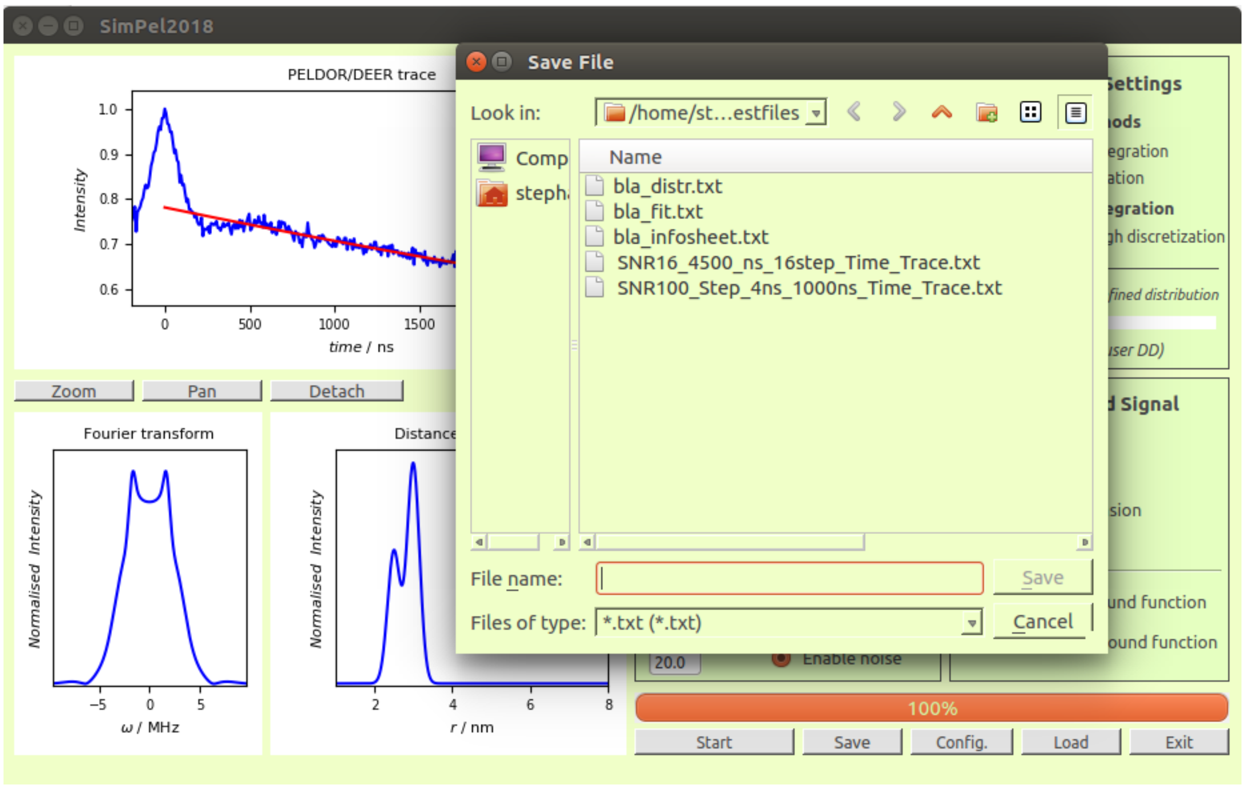


Figure 3: Comparison of standard integration settings (top, blue) and integration with higher discretization of the Gaussian functions (bottom, purple) used for simulation.

This simulation noise can be avoided by enabling the "Higher discretization" check-box (three times higher discretization rate) or by checking/enabling "Fresnel+very high discretization" button (five times higher discretization rate). These should be sufficient for acquiring noise-free simulations of all possible distance distributions.

3.3 Data Output and Figure Export

Once a simulation was carried out, the result can be saved and the figures exported. If the "Save" button is clicked, a file manager opens and the files can be saved at the location defined by the user (Figure 4). The PELDOR/DEER trace, the distance distribution, and the Fourier transform are saved as text files. Additionally, an information sheet is automatically created, containing all simulation options (Script 1). The information sheet contains all information to reproduce the simulation. Naturally, white noise cannot be reproduced exactly as it is created via a pseudo-random number generator.



bla_distr.txt
bla_fit.txt
bla_infosheet.txt
SNR16_4500_ns_16step_Time_Trace.txt
SNR100_Step_4ns_1000ns_Time_Trace.txt

Figure 4: File manager opens when the Save button is clicked. All output files are saved as text-files.

If not disabled in the configurations (for configuration settings see next section) the figures are automatically saved as Portable Network Graphics (png) and as vectorized Portable Document Format (PDF) files.

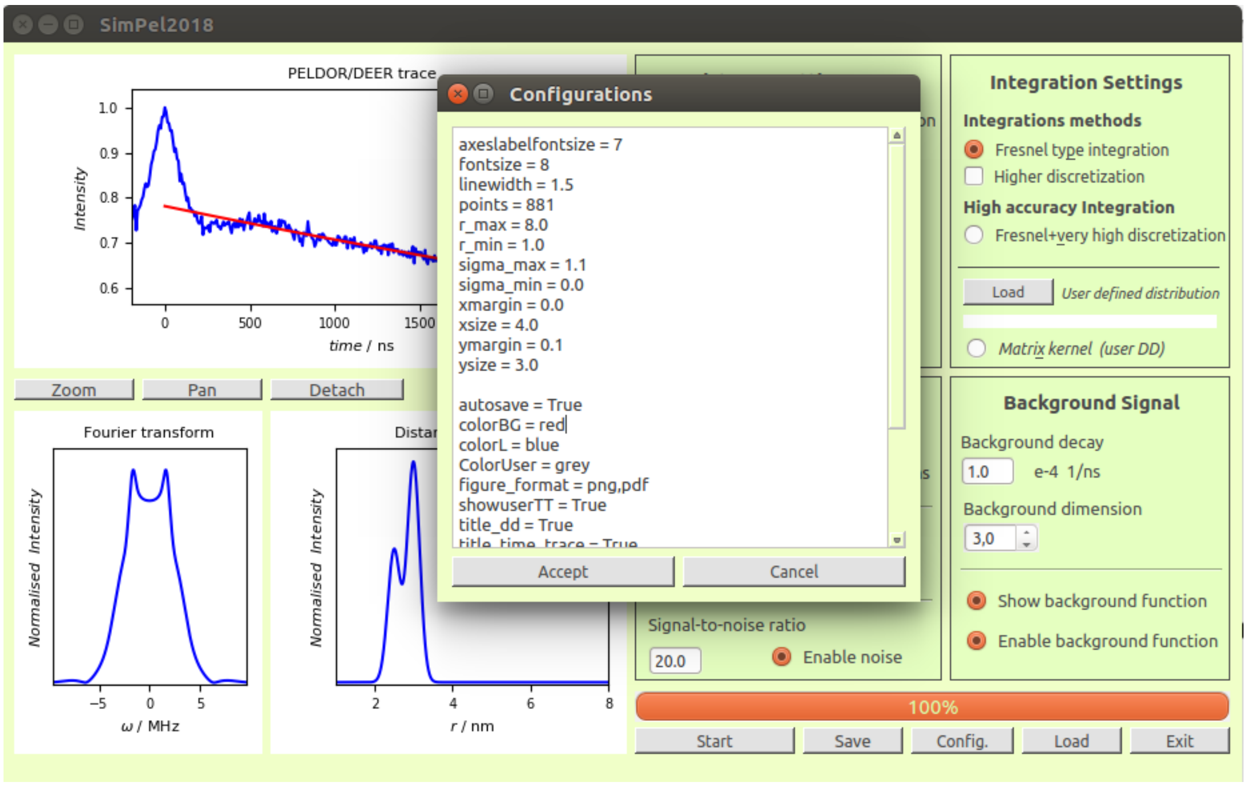
Script 1: Example for an automatically aved information sheet.

```
Infosheet for PELDOR/DEER Simulation. Original Path:
/home/stephan/Python_SimPel/Test_output_1

Simulated with SimPel2018: 1.0.0

Analyzed at: 2018-03-16 11:00:44

*************Simulation Parameter*****************

Distance / nm    Coefficient    sigma / nm
1.8000           1.0000           0.2800
2.5000           0.6000           0.2300
2.0000           0.0000           0.1000
2.0000           0.0000           0.1000
2.0000           0.0000           0.1000


PELDOR/DEER step size: 8.0 ns

PELDOR/DEER time scale: 2000.0 ns

PELDOR/DEER modulation depth: 0.2

Gaussian noise enabled: False

Background function enabled: False

Integration method: Fresnel integrals
```

3.4 Configuration Settings

Many standard settings of SimPel can be changed either dynamically directly in the GUI or as default by creating a configuration file.

The current configuration can be opened and changed when running SimPel by pressing the "Config" button. If new configurations are set and the "Accept" button (see Figure 5) is clicked, the new settings are adapted immediately and a new simulation is carried out.



```
axeslabelfontsize = 7
fontsize = 8
linewidth = 1.5
points = 881
r_max = 8.0
r_min = 1.0
sigma_max = 1.1
sigma_min = 0.0
xmargin = 0.0
xsize = 4.0
ymargin = 0.1
ysize = 3.0

autosave = True
colorBG = red
colorL = blue
ColorUser = grey
figure_format = png,pdf
showuserTT = True
title_dd = True
title_time_trace = True
```

● Fresnel type integration
○ Fresnel+very high discretization
Background decay: 1.0 e-4 1/ns
Background dimension: 3,0
● Show background function
● Enable background function
Signal-to-noise ratio: 20.0 ● Enable noise
100%

Figure 5: Current configurations of SimPel. They can be easily altered and accepted.

Alternatively, user defined configuration setting can be saved in a file called `SimPel.conf`. This configuration file needs to be located in the folder containing the executable or the source code. An example of a user defined `SimPel.conf` is shown in Script 2 and the corresponding changes on the figures are demonstrated in Figure 6.

Script 2: Example for a user defined configuration file (`SimPel.conf`).

```
#User defined default configurations
r_max = 10
r_min = 2
t_min = 0
fontsize = 9.0
Lcolor = green
title_dd = False
title_time_trace = False
yticks_dd = True
ylabel_time_trace = $Form\ \ factor$
```

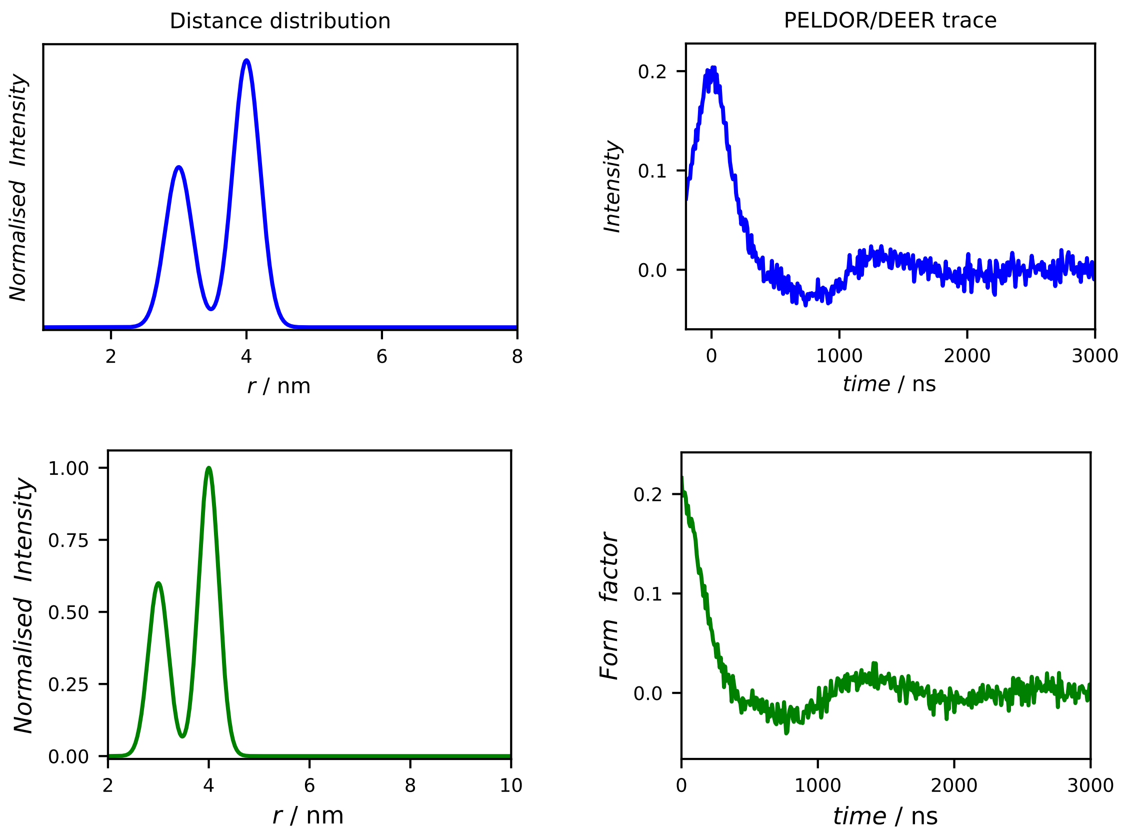
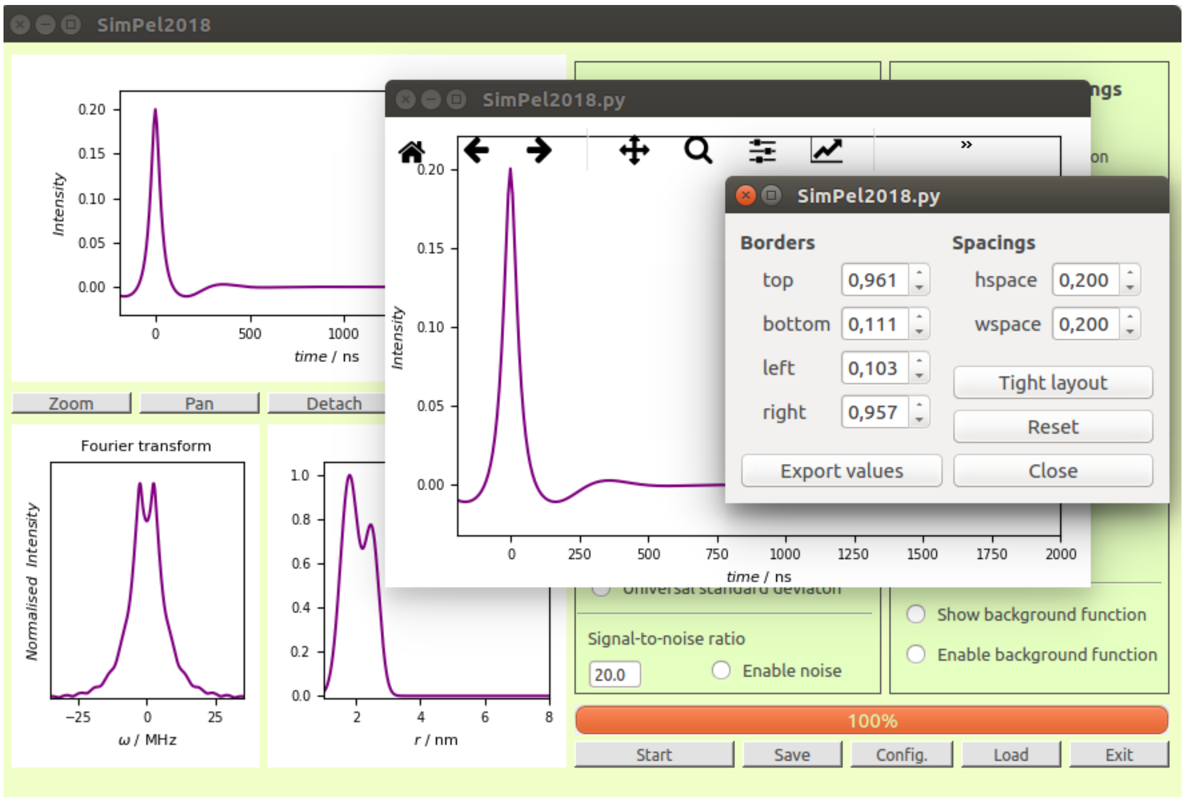


Figure 6: Comparison of standard configurations of SimPel (top, blue) and user-defined default configurations as set in Script 2. Many settings were changed compared to the standard configuration (the fontsize was increased, the distance range and color were changed, ...).

3.5 Detaching Figures

If wished, the plots can be detached. This enables additional options to modify or configure the plots. If new simulations are carried out or other things are changed, e.g. noise is added, the figures are updated automatically.



SimPel2018.py

Borders

top	0,961
bottom	0,111
left	0,103
right	0,957

Spacings

hspace	0,200
wspace	0,200

Figure 7: Main GUI window of SimPel running on a Linux system (Ubuntu 16.04). Detached figures can be modified separately and saved afterwards.

4 Algorithms for the Calculation of PELDOR/DEER Traces

4.1 PELDOR/DEER Trace Simulation Kernel

SimPel calculates a PELDOR/DEER time trace $V(t)$ using:

$$
\begin{aligned}
&\boldsymbol{V} = \lambda \sum_{i=1}^{5} c_i \sum_{k=1}^{j} I(r_{i,k}) S(\omega_{i,k}, t) \\
&\text{with} \quad j = 19 + (\text{round}(100\sigma_i)) + (20 + (\text{round}(100\sigma_i))\%2, \\
&\text{with} \quad I(r_{i,k}) = \frac{1}{N_i} \exp\left[-\frac{1}{2}\left(\frac{r_{i,0} - r_{i,k}}{\sigma_i}\right)^2\right], \\
&\text{with} \quad N_i = \sum_{k=1}^{j} \exp\left[-\frac{1}{2}\left(\frac{r_{i,0} - r_{i,k}}{\sigma_i}\right)^2\right], \\
&\text{with} \quad S(\omega_{i,k}) = \frac{\cos\left(\omega_{i,k}|t|\right)}{z_{i,k}} \cdot \text{FresnelC}(z_{i,k}) + \frac{\sin\left(\omega_{i,k}|t|\right)}{z_{i,k}} \cdot \text{FresnelS}(z_{i,k}), \\
&\text{with} \quad z_{i,k} = \sqrt{\frac{6\,\omega_{i,k}|t|}{\pi}}, \\
&\text{with} \quad \omega_{i,k} = \frac{52.04 \cdot 2\pi}{r_{i,k}^3}, \\
&\text{with} \quad \sum_{k=1}^{5} c_i = 1 \quad \text{and} \quad c_i \in \mathbb{R}_0^+ \ .
\end{aligned}
\tag{1}
$$

λ is the given modulation depth, $r_{i,0}$ are the five given central distances, σ_i are the given standard deviation, c_i are the linear coefficients and t is the time vector running from -200 ns to the user-defined value of $t_{\max}$ with a user-defined step size. The linear coefficients are renormalized to their sum being equal to one. It is therefore not necessary to pay attention to the normalization of the linear coefficients. The normalization factor for a Gaussian function $\left(\frac{1}{\sigma_i\sqrt{2\pi}}\right)$ is not used. Instead the Gaussian functions are normalized to their sum. This is more exact, as different standard deviations cause different discretization rates for different standard deviations. The PELDOR kernel integral is not evaluated directly to avoid numerical integration. Instead, the integral is reformulated as a sum of Fresnel integrals (FresnelC and FresnelS, see Equation 2). The Fresnel integrals can be evaluated in vectorized form by passing a multidimensional array to the SciPy function `fresnel` (scipy.special.fresnel). Indeed, the full PELDOR/DEER trace calculation shown in Equation 1 is carried out without a loop but by passing a 3-dimensional array (i, k, t) to the Fresnel function.

$$\begin{aligned} S(\omega_\mathrm{D}, t) &= \int_0^{\pi/2} \cos\Big((1 - 3\cos^2\phi)\,\omega_\mathrm{D}\,t\Big) \sin\phi \,\mathrm{d}\phi \\ &= \frac{\cos\left(\omega_\mathrm{D} t\right)}{z} \underbrace{\int_0^z \cos\left(\frac{\pi}{2}\,u^2\right)\mathrm{d}u}_{\equiv \mathrm{FresnelC}(z)} + \frac{\sin\left(\omega_\mathrm{D} t\right)}{z} \underbrace{\int_0^z \sin\left(\frac{\pi}{2}\,u^2\right)\mathrm{d}u}_{\equiv \mathrm{FresnelS}(z)} \\ &= \frac{\cos\left(\omega_\mathrm{D} t\right)}{z} \cdot \mathrm{FresnelC}(z) + \frac{\sin\left(\omega_\mathrm{D} t\right)}{z} \cdot \mathrm{FresnelS}(z)\;, \quad \text{with } z = \sqrt{\frac{6\,\omega_\mathrm{D} t}{\pi}} \end{aligned} \tag{2}$$

4.2 PELDOR/DEER Trace Kernel Matrix

If a user-defined distance distribution $\boldsymbol{P} = r_i$ is loaded and the PELDOR/DEER trace has to be calculated, a matrix kernel $\boldsymbol{K}$ is used:

$$\begin{aligned} & V(t) = \boldsymbol{K}\boldsymbol{P}, \\ & \text{with} \quad \boldsymbol{K} = K_{ji} = \frac{\cos\left(\omega_i |t_j|\right)}{z_i} \cdot \mathrm{FresnelC}(z_i) + \frac{\sin\left(\omega_i |t_j|\right)}{z_i} \cdot \mathrm{FresnelS}(z_i), \\ & \text{with} \quad z_i = \sqrt{\frac{6\,\omega_i |t_j|}{\pi}}, \\ & \text{with} \quad \omega_i = \frac{52.04 \cdot 2\pi}{r_i^3}, \end{aligned} \tag{3}$$

The calculation of the kernel matrix $\boldsymbol{K}$ is carried our by passing a 2-dimensional array to the Fresnel function of SciPy. This allows a highly efficient calculation of the kernel matrix.

4.3 Adding a Background Function

A background function can be "added" to the PELDOR kernel function, resulting in the new function $V_b(t)$:

$$\begin{aligned} & V_b(t) = (V(t) + 1 - \lambda)B(t), \\ & \text{with} \quad B(t) = \exp\left[-k|t^{d/3}|\lambda\right] \end{aligned} \tag{4}$$

k is the user-defined background decay constant and d is the user-defined background dimension. The function $V_b(t)$ is subsequently normalized to its maximum. The simulation function without background decay is kept as variable. It is therefore not necessary to rerun a simulation in case the background settings are changed.

4.4 Adding White Noise

The noise used is statistical (uncorrelated) noise, having a probability density function (PDF) equal to that of a normalized Gaussian function. It can be described as additive white Gaussian noise (AWGN). The noise is added to get a noisy time trace $V_n(t)$:

$$\begin{aligned} & V_n(t) = V(t) + n, \\ & \text{with} \quad n = \frac{\lambda}{\text{SNR}}\, p_n(x), \\ & \text{with} \quad p_n(x) = \left(\frac{1}{\sigma_n\sqrt{2\pi}}\right) \exp\left[-\frac{1}{2}\left(\frac{x}{\sigma_n}\right)^2\right] \end{aligned} \tag{5}$$

This implies a mean value of 0. SNR is the given signal-to-noise ratio and σ_n is set to one. The noise-free simulation is kept as variable. Therefore, it is not necessary to rerun a simulation when removing the noise again or changing the signal-to-noise level.

5 Full configuration file (`SimPel.conf`)

Script 3: Complete configuration file of the default settings of SimPel

```
# Simulation Settings
r_max = 8
r_min = 1
sigma_max = 1.1
sigma_min = 0.0
points = 881
t_min = -200

# Figure Settings
fontsize = 9
axeslabelfontsize = 8
xmargin = 0.0
ymargin = 0.1
xsize = 4
ysize = 3
colorL = blue
colorBG = red
ColorUser = grey
linewidth = 1.5
showuserTT = True
yticks_time_trace = True
yticks_dd = False
title_time_trace = True
title_dd = True

# Individual Axes Labels (Latex Math Environment Syntax)
ylabel_time_trace = $Intensity$
xlabel_time_trace = $time\ \mathrm{/ \ ns}$
ylabel_dd = $Normalised\ \ Intensity$
xlabel_dd =$r\ \mathrm{/ \ nm}$

#Save figures
autosave = True
figure_format =png,pdf
```