SoftwareX 7 (2018) 259–262

Contents lists available at ScienceDirect

SoftwareX

journal homepage: www.elsevier.com/locate/softx

Software Update

Update (v1.2) to *DLTPulseGenerator*: A library for the simulation of lifetime spectra based on detector-output pulses

Danny Petschke *, Torsten E.M. Staab

University Wuerzburg, Department of Chemistry, LCTM Roentgenring 11, D-97070 Wuerzburg, Germany

ARTICLE INFO

Article history:
Received 15 June 2018
Received in revised form 22 June 2018
Accepted 22 June 2018

ABSTRACT

In the last update of *DLTPulseGenerator* library (*v1.1*), we realised the simulation of distributed *specific lifetimes* as can be found for the lifetimes of positrons (PALS) in porous materials due to the pore size distribution.

However, in this update *v1.2*, the *DLTPulseGenerator* library was modified to allow the simulation of lifetime spectra consisting of non-*Gaussian* distributed and linearly combined Instrument Response Functions (IRF), since a *Gaussian* shaped instrumental response of a lifetime spectroscopy setup more likely represents an approximation as it reflects the experimentally obtained results. Eventually, this provides an improved modeling of the experimental instrumental response and, finally, leads to a more realistic simulation of lifetime spectra.

Code metadata

Current code version	*v1.2*
Permanent link to code/repository used of this code version	https://github.com/ElsevierSoftwareX/SOFTX-D-17-00077
Legal Code License	*BSD-3-clause*
Code versioning system used	github
Software code languages, tools, and services used	C/C++ and python.
Compilation requirements, operating environments & dependencies	**OS:** Microsoft Windows **compilation requirements (for DLTPulseGenerator.h/.cpp only):** should work with any C++ compiler (has to provide C++11 standard) – recommended: MS-VSCompiler (at least version 2013) **dependencies for example C++ project - AppDLTPulseGenerator:** Microsoft Visual Studio 2015 **dependencies for C++ wrapper in python - pyDLTPulseGenerator.py:** ctypes-library **dependencies for example project in python - pyDLTPulseGeneratorApp.py**: matplotlib, NumPy • recommended software for validation: **DDRS4PALS software v1.04 has implemented the DLTPulseGenerator-library (v1.2)**. A simple xml-file serves as input: https://github.com/dpscience/DDRS4PALS (follow the instructions (wiki) on GitHub to start a simulation)
If available Link to developer documentation/manual	a ***readme.md*** *file* can be found on **github**: https://github.com/dpscience/DLTPulseGenerator/blob/master/README.md
Support email for questions	danny.petschke@uni-wuerzburg.de

1. Introduction and significance

In *DLTPulseGenerator* v1.0 [1] and v1.1 [2], the uncertainties of the ideal lifetime (dt_{ideal}) due to the influences of the setup components, i.e. the instrumental response, were simulated using a *Gaussian* distribution function for the Photo Detection Systems A and B

DOI of original article: http://dx.doi.org/10.1016/j.softx.2018.04.002.

* Corresponding author.

E-mail addresses: danny.petschke@uni-wuerzburg.de (D. Petschke), torsten.staab@uni-wuerzburg.de (T.E.M. Staab).

https://doi.org/10.1016/j.softx.2018.06.003

Table 1
Modifications of struct DLTSetup: The variables PDSUncertaintyA, PDSUncertaintyB and MUUncertainty in DLTPulseGenerator v1.0 [1] and v1.1 [2] were replaced by the structs irfA, irfB (struct type DLTIRF_PDS) and irfMU (struct type DLTIRF_MU) to accomplish the simulation of non-Gaussian distributed and linearly combined IRFs.

struct DLTSetup	DLTPulseGenerator v1.0/1.1	DLTPulseGenerator v1.2
	PDSUncertaintyA (double)	irfA (struct type DLTIRF_PDS)
	PDSUncertaintyB (double)	irfB (struct type DLTIRF_PDS)
	MUUncertainty (double)	irfMU (struct type DLTIRF_MU)

(PDS - A/B), and the Measure Unit (MU). Since the resulting Instrument Response Function (IRF) mostly consists of a contribution significantly deviating from the Gaussian shape, this approximation hardly reflects the experimental results in lifetime spectroscopy methods such as Fluorescence Lifetime Spectroscopy (FLS) and Positron Annihilation Lifetime Spectroscopy (PALS). However, this effect is well known in spectroscopic methods of other fields of research, such as X-ray or neutron powder diffraction, where the peak shape is often described by a *Pseudo-Voigt* profile (*Voigt* profile approximation). This is used for cases where neither a pure *Gaussian* nor a *Lorentzian/Cauchy* distribution function leads to an appropriate fit [3–5]. In PALS, the *Gaussian* approximation is commonly used since the analytical solution exists for a convolution of a *Gaussian* with an *exponential* distribution function, as it was first shown by Kirkegaard and Eldrup in 1972 [6]. Therefore, the least-square fitting can be applied to retrieve the *specific lifetimes* and its corresponding intensities of the lifetime spectrum [6–12].

To investigate the influences on the spectra analysis using non-*Gaussian* distributed and linearly combined Instrument Response Functions, modifications of struct *DLTSetup* were applied.

2. Changes in the software architecture — modifications of struct DLTSetup

The simulation of non-*Gaussian* distributed and linearly combined Instrument Response Functions (IRF) for the Photo Detection System (PDS) and the Measure Unit (MU) is accomplished by replacing the variables ***DLTSetup::PDSUncertaintyA***, ***DLTSetup::PDSUncertaintyB*** and ***DLTSetup::MUUncertainty*** with the new structs *DLTIRF_PDS* and *DLTIRF_MU*, as listed in Table 1.

Each struct ***irfA/irfB*** (***irfMU***) of struct type *DLTIRF_PDS* (*DLTIRF_MU*) consists of a set of max. five weighted (I_i) distribution functions f_i representing the final IRF of PDS A/B (MU) (Fig. 1):

$$\mathrm{IRF}\,(t) = \sum_{i=1}^{5} I_i f_i(t) \text{ with } \sum_{i=1}^{5} I_i \equiv 1. \tag{1}$$

The variables used to model the distribution functions (f_i) are defined in the structs ***irfXPDS*** (***irfXMU***) of struct type *DLTIRF*, where *X* relates to the index *i*.

- **struct DLTIRF**

The number of linearly combined distribution functions for ***irfA/irfB*** and ***irfMU*** is controlled by setting the variable ***enable*** (type: bool). The considered weighting (I_i, Eq. (1)) is defined by the variable ***intensity*** ($0 \le I_i \le 1$), where the sum of all enabled intensities must be equal to one. Setting ***enable*** to *false* results in a zero weighted component ($I_i \equiv 0$, Eq. (1)) and is equivalent to setting the corresponding struct ***irfXPDS*** (***irfXMU***) to the value *IGNORE_DLTIRF*.

Furthermore, three types of distribution functions (*enum DLTDistributionFunction::Function*) are provided (variable ***functionType***):

I. *Gaussian* distribution[1] (**DLTDistributionFunction::Function::GAUSSIAN**)

II. *Cauchy/Lorentzian* distribution[1] (**DLTDistribution Function::Function:: LORENTZIAN_CAUCHY**)

III. *Log-Normal* distribution[1] (**DLTDistributionFunction::Function::LOG_NORMAL**).

The peak location indicating parameters[1] are specified by the variable ***relativeShift*** (in nanoseconds), whereas the normalisation/scaling parameters[1] of the distribution functions are given by the variable ***uncertainty*** (in nanoseconds). Variable ***param*** is reserved for future purposes and can serve as parameter for additional or custom distribution functions, where two parameters are insufficient for the definition (see: modifications in struct DLTSimulationInput [2]).

By enabling only one component ***irfXPDS*** (***irfXMU***) of each struct ***irfA***, ***irfB*** and ***irfMU*** and, moreover, setting the ***functionType*** to **DLTDistributionFunction::Function::GAUSSIAN**, the functionality is equivalent to those of *DLTPulseGenerator* v1.0 [1] and v1.1 [2].

Table 2
IRF: Pseudo-Voigt profile: Listing of the simulation input (left column, variables of struct type DLTIRF) and resulting model fit output (right column). The Pseudo-Voigt based IRF is realised using a linear combination of a Gaussian (Eq. (1)) and Lorentzian/Cauchy (Eq. (2)) distribution function.

Pseudo-Voigt	$\mathbf{IRF_1}$ (simulation input)	$\mathbf{IRF_1}$ (model fit output)
functionType	**::Function::GAUSSIAN**	
uncertainty : σ [ps]	85.0	87.08 ± 2.10
intensity	0.8	0.755 ± 0.080
	$\mathbf{IRF_2}$ (simulation input)	$\mathbf{IRF_2}$ (model fit output)
functionType	**::Function:: LORENTZIAN_CAUCHY**	
uncertainty : s [ps]	85.0	64.91 ± 3.46
intensity	0.2	0.245 ± 0.053

Table 3
Left column: Simulated specific lifetimes and its intensities used for the lifetime spectrum simulation as displayed in Fig. 2a. Right column: Retrieved output by applying the reconvolution approach using DLTReconvolution software [14]. The results of the reconvolution fit indicate an excellent agreement.

	simulation (input)	reconvolution fit (output)
τ_1 [ps]	260.0	260.7 ± 0.6
I_1	0.4	0.406 ± 0.016
τ_2 [ps]	1500.0	1499.9 ± 1.3
I_2	0.6	0.594 ± 0.012

3. Illustrative example — verification of the validity and functionality

For the verification of the validity and functionality of this new feature, a lifetime spectrum consisting of two *specific lifetimes* (see Fig. 2a and Table 3) has been simulated by using the *Pseudo-Voigt* profile

$$V_p\,(t|\eta,\ \sigma,\mathrm{s}) = \ \eta G\,(t|\sigma) + (1-\eta)\,L\,(t|\mathrm{s})\,,\quad 0 \le \eta \le 1 \tag{2}$$

as an example for a linearly combined IRF, consisting of a *Gaussian* (**DLTDistributionFunction::Function::GAUSSIAN**)

$$\mathbf{IRF_1}:\ G\,(t|\mu=0,\ \sigma) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left\{-\frac{t^2}{2\sigma^2}\right\} \tag{3}$$

[1] All formulas and parameter definitions are described in [2].