Solution 2: Nonlinear regression (Lmfit package)

The second approach that could be used for the luminescence decay computing is nonlinear regression. The measured output signal $y(t)$ comprises convolution of luminescence $f(t)$ and excitation impulse $g(t)$:

$$y(t) = f(t) * g(t), \text{ or } y(t) = \int_0^t f(\tau) g(t-\tau) d\tau .$$

If we know the hypothesis model for $f(t)$ with set of some parameters, the corresponding cost function may be built and its minimization procedure could be performed. In our particular case

$f_{\text{model}}(t) = I_0 \exp(-t/t_0)$, with two parameters I_0 and t_0.

Therefore, the model for the output signal will be:

$$y_{\text{model}}(t) = \int_0^t I_0 \exp(-\tau/t_0)\, g(t-\tau) d\tau .$$

The objective is to derive parameters I_0 and t_0 in order to match $y_{\text{model}}(t)$ to $y(t)$. This is typical nonlinear regression problem. The benefit of this approach is obvious: avoiding noise and regularization like in Fourier transform case. Since no signal processing is performed, a considerable gain in accuracy is achieved.

Lmfit is a special extension package for Python – a high-level interface to non-linear optimization and curve fitting problems. It allows computing our problem by minimizing residual (i.e. data-model) array $\{y(t) - y_{\text{model}}(t)\}$.

More information could be found in lmfit github repository.

Unlike FFT/inverse FFT method, where full profile of the $f(t)$ curve is computed, regression works with exponential model $f_{\text{model}}(t)$ only (Fig. 1).

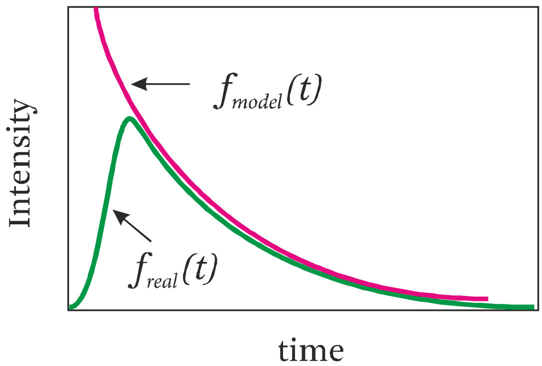

Fig. 1 – The real and modeled luminescence decay curves

This causes some inconveniences with time offsets. Simplified exponential model represents only decay and not the full profile of $f(t)$. Unless the entire curve is obtained, the input data must be preprocessed.

Time-wise, all the signals are biased. Attack times of $y(t)$ and impulse $g(t)$ are different and they are both measured at different times. It could be irrelevant for the conditions, mentioned above. However, due to a simplified model $f_{\text{model}}(t)$, one have to match all the signals to the time scale so that they all have the same starting point at zero seconds (Fig. 2).

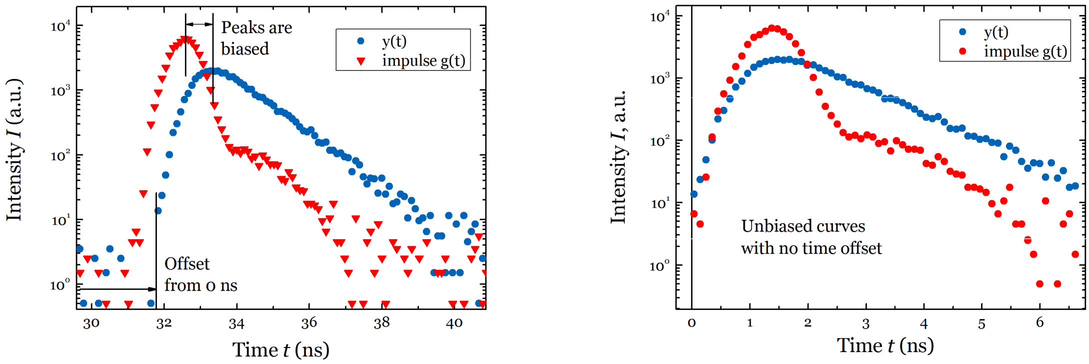

Fig. 2 – Measured (left) and preprocessed (right) input data

Another common problem in nonlinear regression is estimating errors. Asymptotic Standard Errors (ASE) *should not be used for nonlinear models*, since they underestimate the magnitude of the parameter uncertainties. The ASE are based on the "information matrix" and they ignore the off-diagonal elements. Probably the best alternative is using the **Bootstrap approach**. It provides multiple curve-fitting, seeding the data with randomly sampled residuals $\{y(t) - y_{\text{model}}(t)\}$. Histograms of the parameters are obtained as the result.

Finally, mathematical model $f_{\text{model}}(t)$ (or any arbitrary one) could be verified by performing the **Runs Test**. It does not validate the model itself, it simply says if the data and/or assumptions about the data are not consistent with the model.

More on this: doi:10.1016/S0091-679X(07)84024-6

The input data file "*preprocessed_BaF2_78nm.dat*" contains of three columns: time t, output $y(t)$ and excitation impulse $g(t)$. The script file "*Nonlinear_regression_BaF2.py*" performs nonlinear regression, Bootstrap, Runs Test, plotting curves, residuals, histograms of parameters I_0 and t_0, as well as printing the results to stdout.