Manual

A manual for 3-color continuous wave excitation photon trajectory analysis (linear models, Section 4) and customization guides for arbitrary models (Section 5).

Contents

1 Requirements

MATLAB ($\geq$ 2015a) The analysis program has been developed and tested on 2015a and newer versions.

CUDA ($\geq$ 8.0) `https://developer.nvidia.com/cuda-80-ga2-download-archive`. Not necessary if you don't utilize GPU.

In order to use pre-compiled MEX (`https://www.mathworks.com/help/matlab/matlab_external/introducing-mex-files.html`) analysis functions, you need MS windows (x64) and Visual C++ 2015 Redistributable (`https://visualstudio.microsoft.com/vs/older-downloads`). Otherwise, see Section 5 to compile the MEX functions on your system.

2 Installation

Simply unzip to extract files and add the file location to your MATLAB path (typically, it takes less than a minute). For the examples in the script files, copy functions in 'private' directory to one of your MATLAB path.

3 Input files

The photon trajectories need to be formatted in a specific way.

3.1 Photon trajectories

Photon trajectory data consists of a 4-column matrix (Fig. 1). The first column elements are bind to indices of photons. The second column elements are photon arrival times in 100 ns units. The third column corresponds to delay time from the laser pulse trigger signal, which is not used in the analysis of the data collected by continuous-wave mode laser excitation. The fourth column elements correspond to photon color or detection channel.

Put all photon trajectories of the same type (e.g., DA12, DA1, or DA2) into a single matrix. Save each matrix into a MAT file (e.g., DA12.mat, DA1.mat or DA2.mat). For the distinction of individual photon trajectory segments, an additional index array is required as follows.

3.2 Photon trajectory segment index

A photon trajectory segment is a continuous stream of photons emitted by a single molecule. The trajectory from the same molecule can also be separated into different segments when discontinuity occurs before photobleaching of one of the fluorophores such as photoblinking. The photon trajectory segment index array consists of the indices of the last photons of individual segments with 0 for the first element.

For example, the range of the i-th photon trajectory segment in the combined photon trajectory matrix is V(i)+1 - V(i+1) (index starts from 1 in MATLAB) (see Fig. 2).

4 Parameter Optimization

4.1 2-state Examples

Run **FRET3cCW_GUI** (Fig. 3) to test examples. The example includes

1. 2-state folding without acceptor blinking.
2. 2-state binding with acceptor blinking and bleaching.

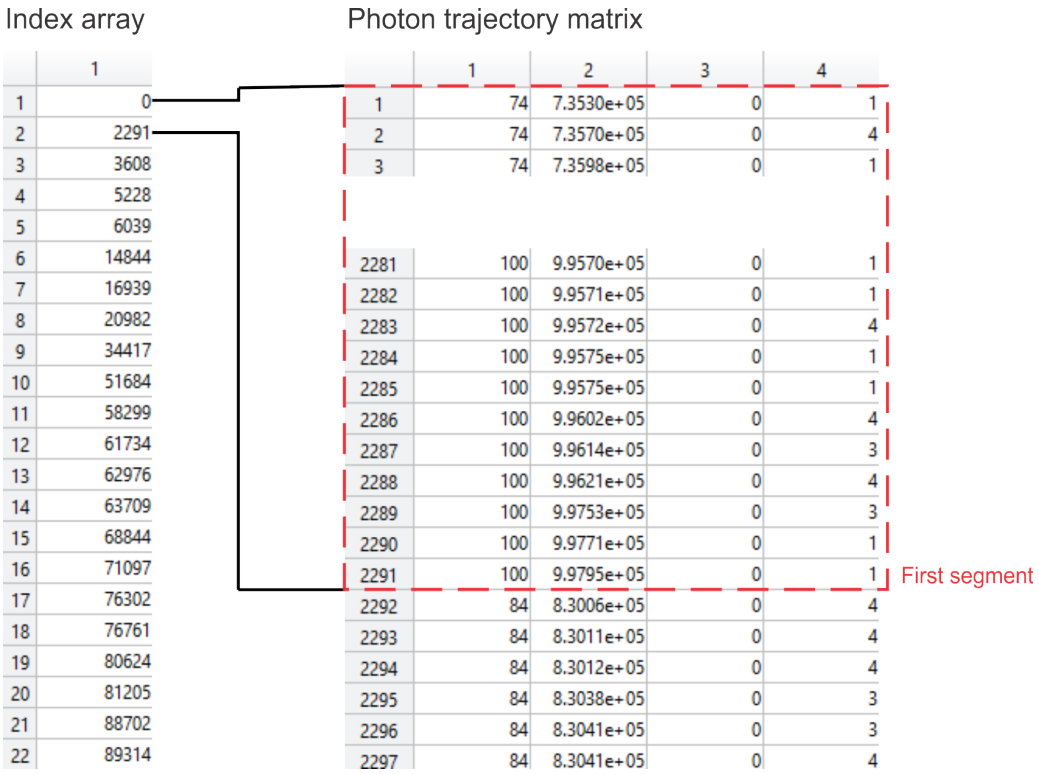
You can also analyze your own data with the same model by providing formatted photon trajectories and indices (Section 3).

	1	2	3	4
1	74	7.3530e+05	0	1
2	74	7.3570e+05	0	4
3	74	7.3598e+05	0	1
4	74	7.3614e+05	0	1
5	74	7.3626e+05	0	4
6	74	7.3660e+05	0	4
7	74	7.3687e+05	0	1
8	74	7.3693e+05	0	4
9	74	7.3710e+05	0	1
10	74	7.3713e+05	0	4
11	74	7.3730e+05	0	3
12	74	7.3782e+05	0	1
13	74	7.3791e+05	0	1
14	74	7.3803e+05	0	1
15	74	7.3803e+05	0	1
16	74	7.3810e+05	0	4
17	74	7.3812e+05	0	1
18	74	7.3813e+05	0	1
19	74	7.3818e+05	0	1
20	74	7.3823e+05	0	4
21	74	7.3837e+05	0	4
22	74	7.3847e+05	0	1

Column1: Bin indices
Column2: Photon arrival time in 100 ns
Column3: Delay time (0 in CW analaysis)
Column4: Detection channel

1: Acceptor 2
3: Acceptor 1
4: Donor

Figure 1: Photon trajectory matrix format



	1
1	0
2	2291
3	3608
4	5228
5	6039
6	14844
7	16939
8	20982
9	34417
10	51684
11	58299
12	61734
13	62976
14	63709
15	68844
16	71097
17	76302
18	76761
19	80624
20	81205
21	88702
22	89314

	1	2	3	4
1	74	7.3530e+05	0	1
2	74	7.3570e+05	0	4
3	74	7.3598e+05	0	1
2281	100	9.9570e+05	0	1
2282	100	9.9571e+05	0	1
2283	100	9.9572e+05	0	4
2284	100	9.9575e+05	0	1
2285	100	9.9575e+05	0	1
2286	100	9.9602e+05	0	4
2287	100	9.9614e+05	0	3
2288	100	9.9621e+05	0	4
2289	100	9.9753e+05	0	3
2290	100	9.9771e+05	0	1
2291	100	9.9795e+05	0	1
2292	84	8.3006e+05	0	4
2293	84	8.3011e+05	0	4
2294	84	8.3012e+05	0	4
2295	84	8.3038e+05	0	3
2296	84	8.3041e+05	0	3
2297	84	8.3041e+05	0	4

Figure 2: Photon trajectory segmentation index and the corresponding segment in the photon trajectory matrix

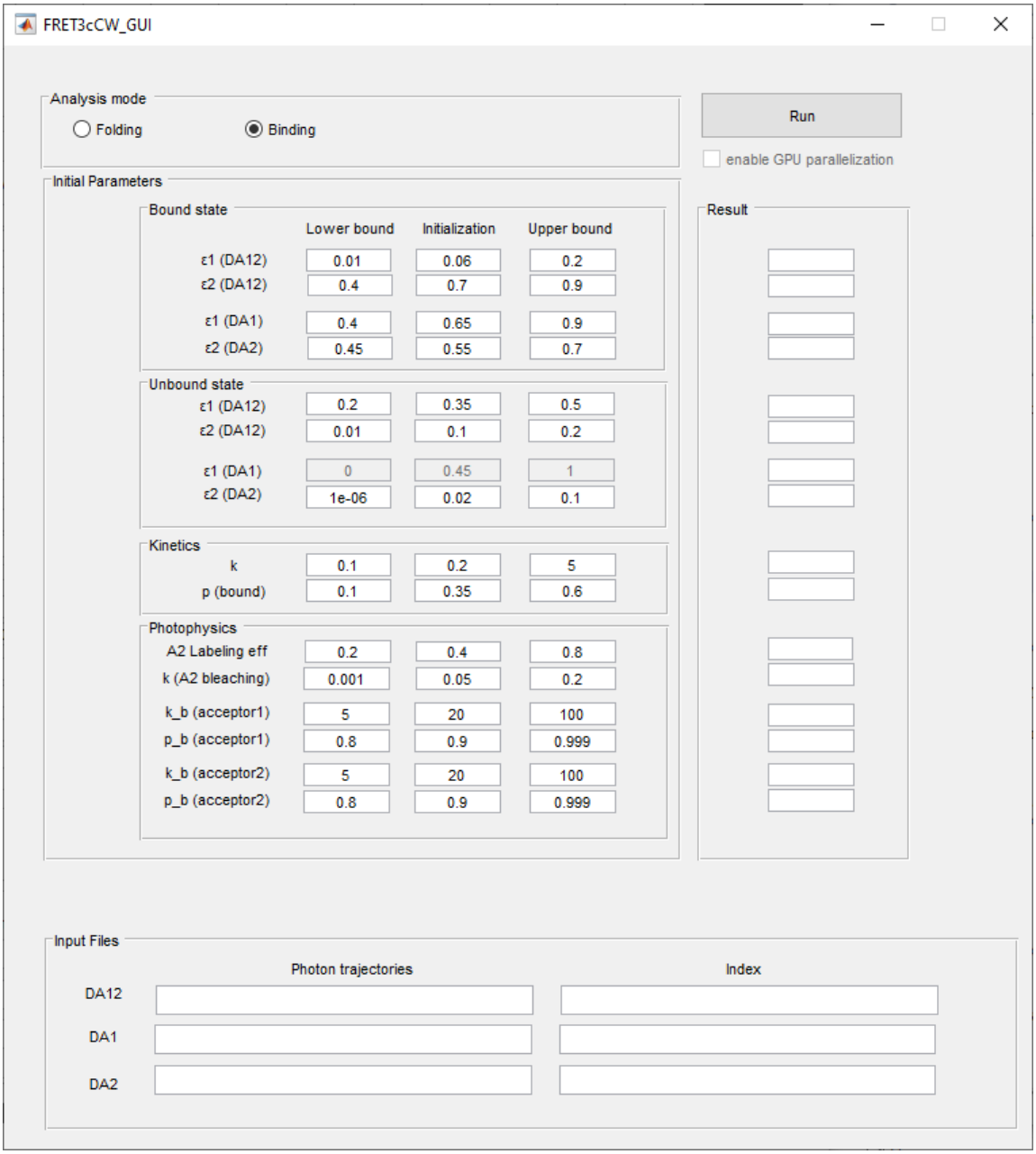


Figure 3: A GUI program for examples and 2-state model analysis

4.2 Folding

For the analysis of folding data, you can call one of the following functions.

- mlhrateeiglingen3cAlexABIC.m
- mlhrateeiglingen3c1Ex_G.m

These functions can be used for any linear kinetics model such as two-state, three-state, etc. As you can guess from the name, the first function can also be used for the alternating laser excitation (ALEX) data (please see **FRET3cCW_Folding.m** and **mlhrateeiglingen3cAlexABIC.m** for more details).

The second function (**mlhrateeiglingen3c1Ex_G.m**) does exactly the same job as the first function but utilizes GPU parallelization.

4.2.1 Input parameters

initparams Initial guess of the parameters.

LUbounds Lower and upper bounds of the parameters.

burstbint3r Photon trajectory matrix.

cumindex Photon trajectory segmentation index.

indexone Number of segments.

fixed Set value to 1 to fix fitting parameters.

stateid The type of a segment. Can have three values. 1: DA1A2 (3-color), 2: DA1, 3: DA2.

exid The type of excitation. 1: donor excitation, 2: acceptor excitation.

chans Detection channels (Acceptor 2, Acceptor 1, Donor).

denatConc Concentration of denaturant. Reserved for future usage.

The following two parameters are for GPU parallelization

maxThread Number of total CPU threads. One thread will be dedicated to GPU communication.

GPU_load A portion of photon trajectories to be loaded on GPU.

After optimization, these two functions return the result parameters. For more details with examples, please see **FRET3cCW_Folding.m**. Expected run time is less than 5 minutes.

4.3 Binding

For the analysis of binding data, you can call the following functions.

- mlhrateeiglingen3c1ExC.m
- mlhrateeiglingen3c1ExABIC.m

These functions can be used for any linear kinetics model, but the index of the unbound state should be specified. For example, for the three-state model with an on-pathway intermediate state (i.e., B – I – U), ubid = 3 (see the input parameters below). The above functions can also be used for folding experiments (only for continuous wave data, please see **FRET3cCW_Binding.m**, **mlhrateeiglingen3c1ExC.m** and **mlhrateeiglingen3c1ExABIC.m** for more details). The second function includes acceptor blinkings.

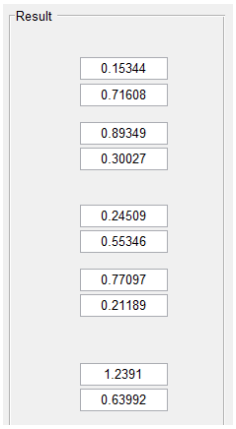


Figure 4: The expected output for the folding demo.

4.3.1 Input parameters

initparams Initial guess of the parameters.

LUbounds Lower and upper bounds of the parameters.

burstbint3r Photon trajectory matrix.

cumindex Photon trajectory segment index.

indexone Number of segments.

isfastbinding The experiment type. 1: binding experiment, 0: folding experiment.

isA2bleach 1 if Acceptor2 bleaches or blink in the bound state, transition kinetics is included in the model. 2 if detailed balance is broken for bleaching or blinking processes.

ubid State index for the unbound state.

stateidML The type of a segment. Can have 2 values. 1: DA1A2/DA1, 3: DA2.

chans Detection channels (Acceptor 2, Acceptor 1, Donor).

fparam Leakages and FRET efficiencies for acceptor dark states.

For more details with examples, please see **FRET3cCW_Binding.m**. Expected run time is less than 10 minutes.

5 Custom models

To implement arbitrary model, you need to build your own function. For this purpose you need additional requirements:

GSL library For parameter optimization and linear algebra.

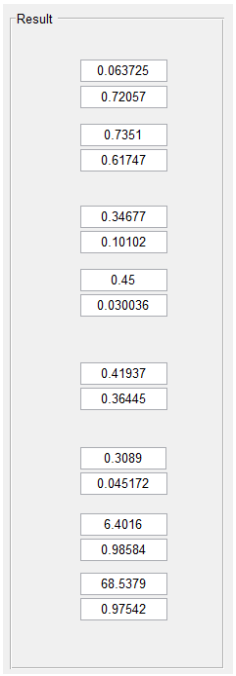


Figure 5: The expected output for the binding demo.

MATLAB library To call C/C++ functions in MATLAB.

Visual studio 2015 Or any C/C++ compiler compatible with MATLAB.

5.1 Requirements

5.1.1 GSL library

The current version of the analysis functions were built with GSL version 2.4. You can download GSL from `https://www.gnu.org/software/gsl/`. For Windows OS, you can use a pre-built version for Visual Studio (`https://www.bruot.org/hp/libraries/`) or you can build GSL using Visual Studio `https://github.com/BrianGladman/gsl.git`.

5.1.2 MATLAB library

MATLAB library for MEX can be found at *YOUR_MATLAB_PATHWAY* /extern/lib. General C++ MEX Application guide can be found at `https://www.mathworks.com/help/matlab/cpp-mex-file-applications.html`

5.1.3 Visual Studio 2015

The current version of the analysis functions were built on Visual Studio 2015 (`https://visualstudio.microsoft.com/vs/older-downloads/`). We provide the visual studio solution for your convenience.

5.2 Customization

Unzip the project file. You can find the source codes in "src" directory and MATLAB wrapper functions in "mWrapper" directory.

5.2.1 Visual Studio 2015

Modify the library paths in "x64MexPropertySheet.props". There are two paths to be modified

C:\Program Files\MATLAB\R2018b This is your MATLAB2018 path.

C:\gsl\lib This is your GSL library path.

Replace these paths to your MATLAB and GSL installation paths.

5.2.2 Step-by-step guide for the installation and the path setting

1. Install MATLAB 2018 (for other MATLAB versions, please check MEX application documents for the corresponding version).
2. Install Visual studio 2015 community version from `https://visualstudio.microsoft.com/vs/older-downloads/`
3. Install CUDA 8.0 from `https://developer.nvidia.com/cuda-80-ga2-download-archive`
4. Install GSL library for Visual Studio 2015 from `https://www.bruot.org/hp/libraries/`
5. Modify paths in "x64MexPropertySheet.props" as described above.

5.2.3 How to introduce a new model

To introduce your own model, you need to check the following functions at least.

analysisThread This is the function for likelihood evaluation. Here you need to define your rate matrix and the way you evaluate the overall likelihood (including photon colors and number of states).

input_from_matlab This function assign MATLAB input double arrays into C/C++ variables. Also, there are some pre-defined constants for certain models.

init_mlh_vars This function initializes variables. Depending on a model vectors and matrices dimension need to be adjusted.

init_mlh_vars_GPU init_mlh_vars GPU version.

5.2.4 MATLAB version dependencies in Visual Studio

For MATLAB 2018 (and maybe newer versions), MW_NEEDS_VERSION_H needs to be defined in the headers for Visual Studio. And MEXFUNCTION_LINKAGE needs to be used to define the gateway function. In older versions, you can use __declspec(dllexport) to define the gateway function.