FOkin User Guide

Géza I. Groma

Institute of Biophysics, Biological Research Centre, Eötvös Loránd Research Network, Szeged, Hungary

Email: groma.geza@brc.hu

Table of contents

Introduction

FOkin (**F**irst-**O**rder **kin**etics) is an object-oriented toolbox written in MATLAB to analyze kinetic data of first-order reaction systems. A detailed study on the problems FOkin can be applied for is published in [1].

Briefly, FOkin addresses the group elastic net problem (GENP) defined as

$$\text{minimize } \frac{1}{2}\sum_{k=1}^{p}\left\|\left(\mathbf{b}_k - \mathbf{A}_k\mathbf{x}_{*,k}\right)\right\|_2^2 + \lambda\left[\frac{1}{2}\left(1-\alpha\right)\sum_{k=1}^{p}\left\|\mathbf{x}_{*,k}\right\|_2^2 + \alpha\sum_{j=1}^{n}\left\|\mathbf{x}_{j,*}\right\|_2\right]. \qquad (1)$$

Here $\mathbf{b}_k$ is the m-vector of the experimental data, the element b_i of which is taken at the time t_i, corresponding to the k^{th} element of the vector of a group parameter $\mathbf{w} = \left(w_1, ..., w_p\right)$, typically the wavelength in a kind of spectroscopic measurement. The design matrix $\mathbf{A}_k$ is defined individually for each value of k. As required for a system of first-order reactions, in the simplest case for all k $\mathbf{A}_k = \mathbf{A}$, an $m \times n$ matrix with elements of

$$A_{ij} = \exp\left(-t_i / \tau_j\right) \;, \qquad (2)$$

where τ_j is an element of the n-vector $\boldsymbol{\tau}$, consisting of pre-defined time constants. In general the pure exponential term in Eq (2) is substituted for the analytical function of their convolution with a Gaussian with mean of t_0 and standard deviation of σ, describing the instrumental response function (IRF) of the measuring apparatus. The unknown distribution is represented by the $n \times p$ matrix $\mathbf{X}$, whose element x_{jk} corresponds to time constant τ_j and wavelength w_k, while $\mathbf{x}_{j,*}$ and $\mathbf{x}_{*,k}$ denote the j^{th} row and k^{th} column of $\mathbf{X}$, respectively. The GENP can be considered as a modified multiple elastic net problem, where the lasso term is substituted by a group-lasso term, ensuring correlation across the individual elements of each row of $\mathbf{X}$ (i.e., the elements corresponding to identical time constants but to different wavelength). Alternatively, if such a correlation is not required, the toolbox can solve the set of minimization problems defined separately for each k as

$$\text{minimize } \left\{\frac{1}{2}\left\|\left(\mathbf{b}_k - \mathbf{A}_k\mathbf{x}_{*,k}\right)\right\|_2^2 + \lambda\left[\frac{1}{2}\left(1-\alpha\right)\left\|\mathbf{x}_{*,k}\right\|_2^2 + \alpha\sum_{j=1}^{n}\left\|x_{j,k}\right\|_2\right]\right\}. \qquad (3)$$

Note that for the scalar operand in the last term of Eq (3) $\left\|x_{j,k}\right\|_2 \equiv \left|x_{j,k}\right|$ holds, hence the problems to solve are equivalent to a multiple elastic net problem (MENP), which turns to the simple lasso with $\alpha = 1$.

The toolbox implements two key tasks on the GENP/MENP:

(i) parameter estimation: with fixed values of the hyperparameters $\lambda > 0$ and $\alpha \in \left[0,1\right]$ it solves the minimization problem defined in Eq (1) for estimation of the distribution $\mathbf{X}$,

(ii) model selection: applying a machine learning procedure, it selects the optimal values of the hyperparameters for task (i), dictated merely by the data themselves.

Both tasks can be executed by an object of the FOkin class, applying its optimize() and select_model() methods, respectively. For technical reasons, the toolbox uses $\omega = 1 - \alpha$ for the second hyperparameter.

Required MATLAB version and toolboxes

The required version of MATLAB is 2019b or higher. Detailed tests were carried out on version 2020b.

MATLAB toolboxes required:

The Statistics and Machine Learning Toolbox is required for

bayesopt() in FOkin.select_model() and FOkin.calc_t0_fwhm()

cvpartition() in the KCV() private function of FOkin class.

The Curve Fitting Toolbox is required for

csaps() in FOkinDiscretized.showd() and FOkinDiscretized.showexp().

The Optimization Toolbox is required for

lsqnonlin() in FOkin.FOkinDiscretized.do_expfit().

The Parallel Computing Toolbox is not essential, it is required only for parallel processing i.e., if the FOkin.Options.num_par_workers property is set to nonzero for

parpool() and parfor() inFOkin.calc_t0_fwhm() and in the kCV() and RCVnv()private functions of FOkin class, which are invoked by FOkin.select_model().

Example programs

The following examples available in the *Examples* subdirectory illustrate all the major capabilities of the FOkin toolbox, by applying the algorithms and reproducing the figures and tables presented in [1]. Note, that due to the stochastic nature of cross-validation and Bayesian optimization, the values of the calculated data can be slightly different for every run.

example1.m – script applying Algorithm2 (based on the RCV(n_v) version of cross-validation) for the analysis of simulated data derived from a complex model of the bacteriorhodopsin photocycle.

example2.m – script for excluding distributed kinetics on the data analyzed by example1.m by model selection based on 10-fold CV (first step of Algorithm 3).

example3.m – script applying Algorithm2 for the analysis of experimental ultrafast fluorescence kinetic data measured on the coenzyme FAD.

example4.m - script for excluding distributed kinetics on the data analyzed by example3.m.

example5.m – script for analysis of simulated data with distributed kinetics by both RCV(n_v) and 10-fold CV.

example6.m – script demonstrating the differences in the results of model selections executed without cross-validation, with 10-fold CV and with RCV(n_v).

example7.m – script for analysis of simulated data with realistic noise.

example8.m – script for analysis of simulated data of Erlang distribution without and with exponential components.

example9.m – script for analysis of simulated data of second-order kinetics without and with exponential components.

create_bR_data.m – function called by example1.m, example2.m example6.m and example7.m.

create_distributed_data.m – function called by example5.m.

create_bR_data_with_real_noise.m – function called by example7.m.

create_Erlang_data.m – function called by example8.m.

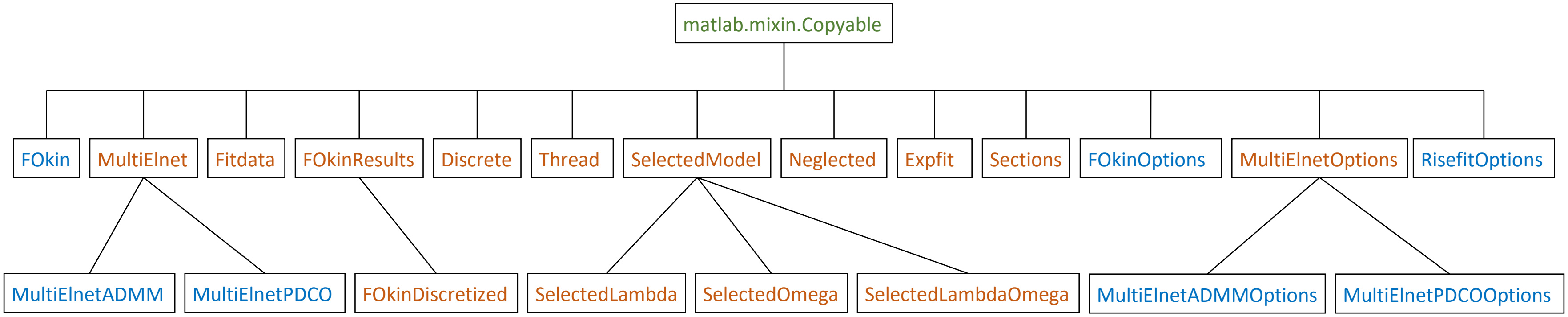
create_2nd_order_data.m – function called by example9.m.

create_raw_bR_data.m – function called by create_bR_data.m and create_bR_data_with_real_noise.m.

bR_spectral_data.mat, bR_rate_data.txt – input data files for create_raw_bR_data.m.

FAD_data.mat – input data files for example3.m and example4.m.

Class hierarchy



The classes in blue are the ones the objects of which are most likely created directly by the user, while that in orange are mainly created internally. The code defining these classes is located in the FOkin directory and its +FOkin subdirectory, respectively. Accordingly, the classes falling under the second category must be referred within the FOkin namespace (e.g.,FOkin.MultiElnet).

All classes of the FOkin toolbox are subclassed from matlab.mixin.Copyable, which is subclassed from the handle class. This means that the objects of these classes are not copied by value but by reference, as demonstrated by the following snippet of code:

```
[data, time, wavelength] = create_bR_data(1.E-3, 10); % create test data
fokin = FOkin(data, time, wavelength); % create a FOkin object
fokin.options.signal_label % display the value of a property chain
opt =  fokin.options; % make a copy of the first property (by reference!)
opt.signal_label = '\DeltaA (rel)'; % assign a new value
fokin.options.signal_label % test it on the original object
```

```
ans = 

    'Fluorescence (rel)'


ans = 

    '\DeltaA (rel)'
```

On the other hand, shallow or deep copies by value can be created by matlab.mixin.Copyable.copy() (for details see the MATLAB documentation). The corresponding copy() methods of all classes of the FOkin toolbox create deep copy.

Class definitions

Only public properties and methods are listed. If not specified, the type of the properties is double.

The FOkin class

Description

The main class of the toolbox with properties defining a GENP/MENP and methods operating on that.

Superclass

matlab.mixin.Copyable

Properties

data – (read only) $m\times p$ matrix, copy of the data input argument of the constructor.

time – (read only) $m\times 1$ vector, copy of the time input argument of the constructor.

group_param – (read only) $1\times p$ vector, copy of the group_param input argument of the constructor.

weight– can be assigned by scalar, $m\times 1$ vector or $m\times p$ matrix, containing the weight of fitting at different points of time. A vector means equal weights for all columns of data, a scalar means equal values for all data points. Value of 1 means unweighted fitting. Default value is 1. Writing this property generates rebuilding of the object. Even if assigned by a scalar or column vector, the values are contained in the form of an $m\times p$ matrix.

t0 – scalar or $1\times p$ vector, containing the values (on the timescale of time) of the Gaussian describing the temporal IRF of the measuring device at the points of group_param. A scalar means equal values for all points. Default value is 0. Writing this property generates rebuilding of the object.

fwhm – scalar or $1\times p$ vector, containing the values (on the timescale of time) of FWHM of the Gaussian describing the temporal IRF of the measuring apparatus at the points of group_param. A scalar means equal values for all points. Zero can be specified for an instantaneous IRF. Default value is 0. Writing this property generates rebuilding of the object.

start – scalar or $1\times p$ vector, containing the index of time where the fitting starts for the points of group_param. A scalar means equal values for all points. Default value is 1. Writing this property generates rebuilding of the object.

name – char array describing the dataset. Default value is ''. Writing this property generates rebuilding of the object.

options – FOkinOptions object. Default value is a new instance of FOkinOptions. Writing this property generates rebuilding of the object.

optimizer – object of a subclass of FOkin.MultiElnet, implementing the do_optimize(), do_get_result(), do_reset(), numiter() and runtime() abstract methods. The currently available classes of this kind are FOkin.MultiElnetADMM and FOkin.MultiElnetPDCO. For a full GENP FOkin.MultiElnetADMM is required. FOkin.MultiElnetPDCO cannot handle the group-lasso penalty but solves the MENP considerably faster than FOkin.MultiElnetADMM. Default value is a new instance of FOkin.MultiElnetADMM. Writing this property generates rebuilding of the object.

tau – (read only) the vector of time constant calculated during the building of the object, consisting of logarithmically equidistant points. tau spans the same interval as time does, extended downward and upward by the number of decades specified in options.extension_lo and options.extension_hi, respectively. In addition, the vector is extended upward by an element with value Inf. The number of points in a decade is determined by options.n_tau_decade.

A – (read only) the design matrices calculated during the building of the object. If the t0, fwhm and start properties are constants, A is a single $(m-\mathit{start})\times n$ matrix, otherwise it is an $1\times p$ cell array, containing matrices, where the size of A{k} is $(m-\mathit{start}\{k\})\times n$.

b – (read only) the values of the data property with maximum absolute value normalized to 1. If A is a single matrix, b is also a single $m\times p$ matrix. Otherwise, it is an $1\times p$ cell array, containing m-vectors.

bfit – (read only) the section of b (to be) participated in the fitting procedure. The structure of bfit is identical to b, but its size can be reduced, according to the corresponding value of the start property.

n, m, p – (read only) the values of n, m and p in Eq (1), respectively.

Methods

FOkin(data, time, group_param, weight, t0, fwhm, start, name) – constructor of the object. Each argument sets a property of identical name. Required arguments are data, time and group_param. For the values in the vector of time an arrangement close to logarithmically equidistant points is recommended. Unspecified arguments leave the default values of the corresponding properties.

optimize(warm) – executes the optimization task on the GENP/MENP defined by object. The actual optimization is carried out by the FOkin.MultiElnet object contained in the optimizer property.

Input arguments:

warm – logical, if true executes a warm restart with the results of the previous run. Default value is 0.

res = results(info) – gathers the results of the last optimization if any.

input arguments:

info – struct with fields of char arrays holding any kind of textual information, which will be copied to the previous field of res.info.

output arguments:

res – FOkin.FokinResults object containing the results of the optimization.

[meanMSPE, constraints, user_data] = CV (lambda, omega, method, warm) – executes cross-validation on the GENP/MENP defined by object.

input arguments:

lambda – scalar specifying the λ hyperparameter of the GENP/MENP.

omega – scalar specifying the ω hyperparameter of the GENP/MENP.

method – 'kCV' or 'RCVnv' for applying the *k*-fold CV or the RCV(n_v) version of cross-validation, respectively.

warm – logical, if true executes a warm restart in the optimization processes of the cross-validation with the results of the previous run. Default value is 0.

output arguments:

meanMSPE– mean of the Mean Squared Prediction Errors (MSPEs) over the specified cross-validation procedure.

constraints – dummy argument with value of [], used only to make an output signature required by the bayesopt() function in the Statistics and Machine Learning Toolbox of MATLAB.

user_data – struct of the following fields:

stdMSPE– standard deviation of the MSPEs.

support_size – (exists only if method is 'RCVnv') size of the support of the solution of GENP/MENP on the original complete dataset. (For the definition of support see the description of FOkin.SelectedOmega.average_support_size.)

selected = select_model(lambda, omega, maxiter, info, method, warm) – selects the optimal values of hyperparameters λ and ω for the GENP/MENP defined by the object.

input arguments:

lambda, omega – scalar or 2-vector. A scalar means that the corresponding λ or ω hyperparameter is kept fixed at that value. A vector specifies the limits of the interval within the hyperparameter is varied for finding the optimum. At least one of these arguments must be a vector. The Bayesian optimization algorithm implemented in the bayesopt() function in the Statistics and Machine Learning Toolbox of MATLAB and based on the CV() method is applied to select the hyperparameter(s) at the minimum value of the MSPE.

maxiter – positive integer specifying the number of objective function evaluations as stopping criteria (see 'MaxObjectiveEvaluations' in the name-value pair input arguments of bayesopt()).

info – struct with fields of char arrays holding any kind of textual information, which will be copied to the previous field of selected.info.

method, warm – arguments passed to the underlying CV() method.

output arguments:

selected – FOkin.SelectedOmega object if lambda is a scalar, FOkin.SelectedLambda object if omega is a scalar, FOkin.SelectedLambdaOmega object otherwise.

[t0, fwhm, rel_params] = calc_t0_fwhm(maxiter) – estimates the value of the t0 and fwhm properties from the data themselves. The dependence of the elements of both t0 and fwhm on the group_param property is modeled by a spline function. The number of the knots and the allowed range of t0 and fwhm are defined in the options.n_t0_knots, options.n_fwhm_knots, options.t0_range and options.fwhm_range properties, respectively. The x components of the knots are distributed equidistantly over the range of group_param, while their y components are free parameters to be optimized by a Bayesian optimization algorithm in the same way as the select_model() method does. Alternatively, the options.fwhm_fixed property can specify an fwhm vector of fixed elements. Note that using this method is not recommended in a direct way but as incorporated in the FOkin.risefit() function, which can handle the same problem in a more sophisticated manner by adjustable control parameters.

input parameter:

maxiter – positive integer specifying the number of objective function evaluations as stopping criteria for the underlying Bayesian optimization algorithm.

The FOkinOptions class

Description

Specifies options for a FOkin object as assigned to its options property.

Superclass

matlab.mixin.Copyable

Properties

General options:

n_tau_decade – number of points in a decade of the tau property of the parent FOkin object.

Writing this property generates rebuilding of the FOkin object. Default value is 50.

extension_lo, extension_hi – number of decades by which the lower/higher limit of the tau property of the parent FOkin object are extended with respect to the first/last point of the time property of the FOkin object. Fractional and negative values are allowed. Writing this property generates rebuilding of the FOkin object. Default value is 1.

num_par_workers – number of required parpool workers in parfor loops in the methods of the parent FOkin object. Set to zero to run parfor as for. Set to Inf to use all the available workers. Nonzero numbers need the Parallel Computing Toolbox of MATLAB to be installed. Default value is 0.

parpool_spec – char array containing a valid profile name acceptable as the resources input argument of the parpool() function in the Parallel Computing Toolbox, called by the methods of the parent FOkin object. Ignored if num_par_workers is 0. Otherwise, see the documentation for parpool(). Default value is 'local'.

group_lasso– logical, applies only if the p property of the parent FOkin object is > 1. If true, the full GENP defined in Eq(1) will be solved by the object assigned to the optimizer property of the parent FOkin object. If false, MENP defined in Eq (3) will be applied separately to the proper subsets of the data. If the bfit property of the parent FOkin object is a matrix, these subsets are equivalent to its columns. If bfit is a cell array, they are equivalent to the vectors contained in its elements. Default value is 1.

signal_label – char array used in y labels of plots in visualization of the results for characterization of the data property of the parent FOkin object. Default value is 'Fluorescence (rel)'.

time_unit – char array used in x labels of plots and in table headers for the units of the time property of the parent FOkin object. Default value is 'ps'.

group_param_label – char array used in x labels of plots for characterization of the group_param property of the parent FOkin object. Default value is 'Wavelength (nm)'.

group_param_name – char array used in titles of plots for characterization of the group_param property of the parent FOkin object. Default value is 'wavelength'.

group_param_unit – char array used in titles of plots for the units of the group_param property of the parent FOkin object. Default value is 'nm'.

Options for cross-validation:

cv_nfold – specifies the number of folds in *k*-fold cross-validation. Default value is 10.

cv_nrep – specifies the number of repetitions in RCV(n_v). Default value is 1E4.

nc_rel – specifies the value of ${}^{n_c}\!/_{n}$ where n_c is the sample size of the construction subset and n is the total sample size in RCV(n_v). Default value is 0.9.

lsqminnorm_tol – Optional tolerance for the lsqminnorm() function of MATLAB applied in the RCV(n_v) algorithm. If [] (recommended), the tolerance is determined internally. Default value is [].

nonzero_limit – An element of the solution matrix $\mathbf{X}$ of a GENP/MEMP is considered nonzero if its absolute value is higher than this value. Applied in the following methods: FOkin.CV() (for RCVnv), FOkin.FOkinResults.discretize() and FOkin.SelectedOmega.count_support(). Suggested value: 0 for MultiElnetADMM and 1.E-6 for MultiElnetPDCO. Default value is 0.

Options for FOkin.FOkinResults.discretize() (see details there)**:**

min_feature_gap – minimum number of points in a gap required for separating two features in the solution. Default value is 1.

thread_row_region – maximum number of rows to step up or down for finding the next point of a thread. Default value is 20.

min_thread_length – a minimum allowed length of a thread. Default value is 3.

thread_colors – color order used in FOkin.FOkinDiscretized.showd() and FOkin.FOkinDiscretized.showexp(). Default value is

```
[0   0   1      % blue
0.39 0.83 0.07  % green
1    0    0     % red
0.06 1    1     % cyan
1    0    1     % magenta
0.93 0.69 0.13  % gold
0.64 0.08 0.18]; % brown.
```

smooth_tau, smooth_val – Smoothness of the splines defining the continuous line across the time constants and the amplitudes, respectively, on a scale of (0-1). 0: a constant equal to the mean of the values. 1: the smoothest spline crossing all points. Default values are 1.E-5 and 1.E-3, respectively.

Options for controlling the execution of the bayesopt() function in the Statistics and Machine Learning Toolbox of MATLAB, applied in the FOkin.select_params() and FOkin.calc_t0_fwhm() methods. The optional name-value input argument pairs of bayesopt() are formed from the property names (without the leading 'bo_') and their values, respectively. See the corresponding MATLAB documentation for details. A NaN value forces using the default MATLAB value. 'MaxObjectiveEvaluations' is not included here as it obtains value directly from an input argument of the methods.

bo_AcquisitionFunctionName – char array, default value is 'expected-improvement-plus'.

bo_IsObjectiveDeterministic – logical, default value is 0.

bo_ExplorationRatio – default value is 0.5.

bo_GPActiveSetSize – default value is 1E3 (for a slower but more precise optimization).

bo_UseParallel – logical, default value is 0.

bo_ParallelMethod – char array, default value is 'clipped-model-prediction'.

bo_MinWorkerUtilization – default value is NaN.

bo_MaxTime – default value is Inf.

bo_NumSeedPoints – default value is 4.

bo_XConstraintFcn – function handle, default value is [].

bo_ConditionalVariableFcn – function handle, default value is [].

bo_NumCoupledConstraints – default value is 0.

bo_AreCoupledConstraintsDeterministic – logical array, default value is NaN.

bo_CoupledConstraintTolerances – vector, default value is NaN.

bo_Verbose – 0, 1 or 2, default value is 1.

bo_OutputFcn – function handle or cell array of function handles, default value is {}.

bo_SaveFileName – char array, default value is 'BayesoptResults.mat'.

bo_SaveVariableName – char array, default value is 'BayesoptResults'.

bo_PlotFcn – A function handle, cell array of function handles, or 'all', default value is {@FOkin.FOkinPlotObjectiveModel, @plotAcquisitionFunction}.

bo_InitialX – table, default value is NaN.

bo_InitialObjective – vector, default value is [].

bo_InitialConstraintViolations – matrix, default value is [].

bo_InitialErrorValues – vector, default value is [].

bo_InitialUserData – cell array, default value is {}.

bo_InitialObjectiveEvaluationTimes – vector, default value is [].

bo_InitialIterationTimes – vector, default value is [].

Options for FOkin.calc_t0_fwhm():

n_t0_knots – number of knots in the spline defining the values of output argument vector t0. Default value is 3.

t0_range – (2-vector) allowed range of t0 relatively to its automatically estimated value. Default value is [0.5, 1].

n_fwhm_knots – number of knots in the spline defining the values of output argument vector fwhm. Applies only if the fwhm_fixed property is []. Default value is 1.

fwhm_range – (2-vector) allowed range of fwhm in its own units. Applies only if fwhm_fixed is []. Default value is [35, 50].

fwhm_fixed – vector of the fixed values of fwhm. If [] fwhm will be approximated by a spline. Default value is [].

The FOkin.MultiElnet class

Description

Common superclass of classes to solve the GENP/MENP with abstract methods of do_optimize(), do_get_result(), do_reset(), numiter() and runtime().

Superclass

matlab.mixin.Copyable

Subclasses

MultiElnetADMM, MultiElnetPDCO

Properties

name – char array describing the object. Default value is ''.

A – the design matrices of the GENP/MENP as described for FOkin.A. Default value is [].

b – the data of the GENP/MENP as described for FOkin.bfit. Default value is [].

lambda – the $\lambda > 0$ hyperparameter of the GENP/MENP. Default value is 1E.4.

alpha – the $\alpha \in [0,1]$ hyperparameter of the GENP/MENP. Default value is 1.

omega – the value of $\omega = 1 - \alpha$. On reading, it is calculated from the property alpha. On writing $\omega \in [0,1]$ must be satisfied and the value of alpha is assigned to $\alpha = 1 - \omega$.

lambda1 – (read only) the value of $\lambda_1 = \lambda\alpha$.

lambda2 – (read only) the value of $\lambda_2 = \lambda(1-\alpha)$.

options – object of FOkin.MultiElnetOptions. If an object of any subclass of FOkin.MultiElnetOptions is assigned to this property, all subsequent assignments will require an object of the same subclass. Default value is FOkin.MultiElnetOptions.

supports_group – (read only) logical, if true the object supports the group-lasso penalty (GENP), if false it supports only the lasso penalty (MENP).

results – (read only) results of optimization, i.e., the $\mathbf{X}$ matrix for GENP or a column of that for MENP.

Methods

MultiElnet(A, b) – constructor of the object. Each argument sets a property of identical name.

optimize(warm) – executes optimization as described for FOkin.optimize(warm). The actual optimization is executed by the protected do_optimize() method, required to be implemented for fully functional subclasses.

reset() – resets the object to its initial state as done by the constructor. The actual resetting is executed by the protected do_reset() method, required to be implemented for fully functional subclasses.

The MultiElnetADMM class

Description

A fully functional optimizer implementing all abstract methods of FOkin.MultiElnet and supporting the group-lasso penalty. The implementation of the do_optimize() protected method applies the Alternating Direction Method of Multipliers (ADMM) [2]. The corresponding code is a modified version of the group_lasso.m function publicly available from [3].

Superclass

FOkin.MultiElnet

Properties

No extra property is defined beyond that in FOkin.MultiElnet.

Methods

MultiElnetADMM(A, b) – constructor of the object with arguments identical to that of FOkin.MultiElnet(A, b)

numiter = numiter() – returns the number of iterations executed by the optimize() method.

runtime = runtime() – returns the runtime of the optimize() method in seconds.

diagnose() – executes graphically visualized diagnosis on the convergence of the optimization, see the code in do_optimize.m and diagnose.m and read [2] for details. If the value of the DIAGNOSE property of the MultiElnetADMMOptions object assigned to the options property of an object of this class is > 1 the method is invoked automatically during the execution of the optimize() method and can be called also after the termination of that. Otherwise optimize() does not collect the information needed for this method.

The MultiElnetPDCO class

Description

A fully functional optimizer implementing all abstract methods of FOkin.MultiElnet but not supporting the group-lasso penalty. The implementation of the do_optimize() protected method applies the Primal-Dual interior method for Convex Objectives (PDCO) [4]. The underlying pdco.m function is an unmodified copy of that publicly available from [5].

Superclass

FOkin.MultiElnet

Properties

No extra property is defined beyond that in FOkin.MultiElnet.

Methods

MultiElnetPDCO(A, b) – constructor of the object with arguments identical to that of FOkin.MultiElnet(A, b)

numiter = numiter() – returns the number of iterations executed by the optimize() method.

runtime = runtime() – returns the runtime of the optimize() method in seconds.

The FOkin.MultiElnetOptions class

Description

Specifies options for a MultiElnet object as assigned to its options property.

Superclass

matlab.mixin.Copyable

Subclasses

MultiElnetADMMOptions, MultiElnetPDCOOptions

Properties

DIAGNOSE – possible values: 0, 1, 2 specifying the level of diagnosis. Applies only if the parent object is a subclass of FOkin.MultiElnet, defining a method to do that. Default value is 0.

The MultiElnetADMMOptions class

Description

Specifies options for a MultiElnetADMM object as assigned to its options property.

Superclass

FOkin.MultiElnetOptions

Properties

MAX_ITER – maximum number of iterations, if reached, an error message is invoked. Default value is 1E5.

ABSTOL, RELTOL,RHO0 and ALPHA_RELAX – control parameters for the ADMM algorithm, for details see the code and [2].

The MultiElnetPDCOOptions class

Description

Specifies options for a MultiElnetPDCO object as assigned to its options property.

Superclass

FOkin.MultiElnetOptions

Properties

MaxIter – maximum number of iterations, if reached, an error message is invoked. Default value is 1E4.

FeaTol, OptTol, Print, StepTol, StepSame, x0min, z0min, mu0, backtrack, Method, LSMRMaxIter, LSMRatol1, LSMRatol2, LSMRconlim and wait – control parameters for the PDCO algorithm, for details see the code and the documentation at [5].

The FOkin.Fitdata class

Description

Container for the data in a FOkin object and the results of FOkin.optimize() or FOkin.CV() executed last time for a single element of FOkin.group_param. An object of this class is internally created upon the execution of the above methods and assigned to the fitdata property of the FOkin.FOkinResults object returned by FOkin.results(). For a sort of property, the value depends on the level of information of the parent FOkin.FOkinResults object (see there).

Superclass

matlab.mixin.Copyable

Properties

group_param – (read only) the value of the corresponding element of FOkin.group_param.

b – (read only) the value of the corresponding element of FOkin.b.

t0 – (read only) the value of the corresponding element of FOkin.t0.

fwhm – (read only) the value of the corresponding element of FOkin.fwhm.

start – (read only) the value of the corresponding element of FOkin.start.

bfit – (read only) the value of the corresponding element of FOkin.bfit.

weightfit – (read only) the section of FOkin.weight participated in the fitting procedure, according to the corresponding values of the group_param and start properties.

fit – (read only, [] for level 0) the vector fitted to the property bfit by the optimization process.

residual – (read only, [] for level 0) the value of bfit – fit.

objval – (read only, [] for level 0 and 1) for MENP the value of the objective function at its minimum calculated by FOkin.optimize() for the corresponding element of FOkin.group_param. For GENP the value is [].

MSE – (read only, [] for level 0) the value of the weighted mean square error calculated from the values of the residual and the weightfit properties.

The FOkin.FOkinResults class

Description

Container for the data in a FOkin object and the results of FOkin.optimize() or FOkin.CV() executed last time for the whole dataset. An object of this class is internally created upon execution of FOkin.optimize() and returned by FOkin.results(). For a sort of property, the value depends on the level of information defined by the value of the operation property as follows:

0 – for ‘no operation’ and ‘select model’

1 – for ‘kCV’ or ‘RCVnv’

2 – for ‘optimize’

Superclass

matlab.mixin.Copyable

Subclasses

FOkin.FOkinDiscretized

Properties

name – char array describing the object. Default value is the copy of the name property of the generating FOkin object.

options – deep copy of FOkin.options decoupled from the parent FOkin object. Useful for resetting the options for the discretize() method.

operation – (read only) char array describing of the name of the operation executed to obtain this object. Possible values are:

‘optimize’ for FOkin.optimize()

‘kCV’ or ‘RCVnv’ for FOkin.CV(), depending on its method input argument

‘select_model’ for FOkin.select_model()

‘no operation’ for an object without any operation or with an uncompleted operation (e.g., due to error, stop in debug or pressing Ctrl C).

timestamp – (read only) char array describing the date and time of the creation of the object.

time, group_param – (read only) values of the corresponding properties of the FOkin object the results were obtained from.

tau – (read only) $(n-1)$-vector, containing the finite elements of the FOkin.tau property.

norm – (read only) in a newly created object the value is the factor by which FOkin.data is divided to obtain Fokin.b (i.e., the maximum absolute value of FOkin.data). On execution of the denorm() method the value becomes 1, while the execution of the renorm() methodreturns the original value. For details see the description of these methods.

fitdata – (read only) array of p, containing FOkin.Fitdata objects corresponding to the different elements of FOkin.group_param.

lambda, omega – (read only, [] for level 0) values of the corresponding properties of the FOkin.optimizer object.

x – (read only, [] for level 0) for level 2, an $(n-1)\times p$ matrix, containing the solution of the GENP/MENP corresponding to the finite elements of the FOkin.tau property. For level 1, the average of the solutions obtained on the training subsets of the data.

x0 – (read only, [] for level 0) for level 2, an $1\times p$ vector, containing the solution of the GENP/MENP corresponding to the infinite element of the FOkin.tau property. For level 1, the average of the solutions obtained on the training subsets of the data.

numiter – (read only, [] for level 0 and 1) scalar (for GENP) or p-vector (for MENP) with value(s) returned by FOkin.optimizer.numiter().

runtime – (read only, [] for level 0 and 1) scalar (for GENP) or p-vector (for MENP) with value(s) returned by FOkin.optimizer.runtime().

runtime_total – (read only, [] for level 0 and 1) the value of the runtime property if it is a scalar, otherwise the value of the sum of its elements.

group_objval – (read only, [] for level 0 and 1) for GENP the value of the objective function at its minimum calculated by FOkin.optimize(). For MENP its value is [].

MSE_total – (read only, [] for level 0) the value of the weighted mean square error calculated for the whole dataset.

info – (read only) struct of the following fields providing further information on the parameters of the corresponding FOkin object at the time of the optimization:

name, weight, t0, fwhm, start – copies of the corresponding properties of the FOkin object at the time of the last operation.

opt – struct with the following fields:

fokin – struct of fields with names and values identical to that of the properties of the corresponding FOkin.options object.

elnet – struct of fields with names and values identical to that of the properties of the corresponding FOkin.optimizer.options object.

previous – copy of the info input argument of FOkin.results().

Methods

The constructor of the class is not public.

denorm() – recalculates the data and the result by reverse normalizing with the value of the norm property, which then takes the value of 1. Subsequent calls of the method have no effects.

renorm() – If called after the execution of the denorm() method, recalculates the original data and results, including the value of the norm property. Subsequent calls of the method have no effects.

show(selection) – graphically visualizes the solution and the corresponding fits.

input argument:

selection – an array of values of the group_param property for which graphical display is required. For invalid values, no display takes place. A value of [] invokes display for all elements of group_param. Default value is [].

disc = discretize() – (can be executed only with operation property of 'optimize') discretizes the positions and values belonging to the different features in the solution contained in the x property. A feature – corresponding to a given element of group_param – is defined as a section in a column of the x property containing nonzero values with possible gaps of contiguous zero values if the lengths of these gaps are less than the value of options.min_feature_gap. A value is considered nonzero if its absolute value is higher than the value of options.nonzero_limit. The discretization of a feature results in a pair of time constant and amplitude. The time constant is determined by averaging the elements of tau falling into the region of the feature, applying the absolute value of the corresponding amplitudes as weighting factors. The amplitude is calculated by adding that of the contributing individual elements. The obtained time constants then sorted into threads spanning the whole range of the group_param. For GENP this sorting is trivial, due to the exact coincidence of the positions of the threads for all values of group_param. For MENP the threads are built by an algorithm, iteratively selecting new elements from the closest points. The value of options. thread_row_region controls the maximum number of rows to step upward or downward for finding the next point of a particular thread. The minimum allowed length of a thread is determined by the value of option. min_thread_length. The elements of the x0 can form a separate thread. If the vector of group_param represents the wavelength value of a spectroscopic experiment, the amplitudes corresponding to a thread can be interpreted as a decay associated spectrum (DAS) or decay associated difference spectrum (DADS) related to the average of time constants in the thread.

output argument:

disc – FOkin.FOkinDiscretized object.

The FOkin.Thread class

Description

Container for the data of threads generated by FOkin.FOkinResults.discretize() (see there for details).

Superclass

matlab.mixin.Copyable

Properties

param – (read only) $1\times p$ vector, copy of the group_param property of the generating FOkin.FOkinResults object.

tau – (read only) $1\times p$ vector, containing the time constants of the thread.

val – (read only) $1\times p$ vector, containing the amplitudes of the thread.

The FOkin.Neglected class

Description

Container for the neglected threads created by FOkin.FOkinDiscretized.neglect().

Superclass

matlab.mixin.Copyable

Properties

threads, average_tau, rel_amplitude – (read only) see the same properties of the FOkin.Discrete class.

limit – the value of the limit input argument of the creating FOkin.FOkinDiscretized.neglect() method.

The FOkin.Expfit class

Description

Container for the results of exponential fitting executed by FOkin.FOkinDiscretized.do_expfit().

Superclass

matlab.mixin.Copyable

Properties

tau – (read only) $q \times 1$vector, containing the time constants, where q is the length of the discrete.threads property of the invoking FOkin.FOkinDiscretized object.

DADS – (read only) $q \times p$ matrix, the rows of which contain the amplitudes corresponding to the different elements of tau. (For spectroscopic data it represents decay associated spectra or decay associated difference spectra.)

rel_amplitude – (read only) $q \times 1$vector, containing the maximum of the absolute value of each row of DADS, divided by the value of abs_max_amplitude.

abs_max_amplitude– (read only) the maximum of the absolute value of DADS.

fit – (read only) $1 \times p$ cell array, containing the fitting vectors to the bfit properties of the corresponding elements of the fitdata property of the invoking FOkin.FOkinDiscretized object.

residual – (read only) $1 \times p$ cell array, containing the residuals (residual = bfit – fit with the corresponding elements as described above).

MSE – (read only) $1 \times p$ vector, containing the mean square errors, corresponding to the different elements of the group_param property of the invoking FOkin.FOkinDiscretized object.

MSE_total – (read only) the mean square errors corresponding to the whole dataset.

The FOkin.Discrete class

Description

Container for the discretized components created by FOkin.FOkinResults.discretize().

Superclass

matlab.mixin.Copyable

Properties

threads – (read only) array of FOkin.Thread objects. The length of the array is denoted by q.

average_tau – (read only) $q \times 1$vector, containing the average of the time constants of each thread over the whole range of the group_param property vector of the invoking FOkin.FOkinResults object.

rel_amplitude – (read only) $q \times 1$vector, containing the maximum of the absolute value of the val property of each element of threads, divided by the value of abs_max_amplitude.

abs_max_amplitude – (read only) the maximum of the absolute value of the whole set of the val properties collected from all elements of threads.

neglected – (read only) FOkin.Neglected object with empty property values for a new instance of FOkin.Discrete. The properties are assigned by calling the neglect() method of the parent FOkin.FOkinDiscretized object.

expfit – (read only) FOkin.Expfit object with empty property values for a new instance of FOkin.Discrete. The properties are assigned by calling the do_expfit() method of the parent FOkin.FOkinDiscretized object.

The FOkin.FOkinDiscretized class

Description

Objects of this class are created by invoking of FOkin.FOkinResults.discretize(). On creation all properties of the invoking object are copied to the constructed object and the results of the discretization are assigned to its discrete property.

Superclass

FOkin.FOkinResults

Properties

discrete – FOkin.Discrete object containing the results of the discretization.

Methods

The constructor of the class is not public.

denorm(), renorm () – these methods of the superclass are overridden to extend their effect on the properties of the discrete property.

neglect(limit) – moves the corresponding elements of discrete.threads, discrete.average_tau and discrete.rel_amplitude to the equivalent sub-properties of the discrete.neglected property, provided that values of discrete.rel_amplitude of that elements are not higher than the value of limit. If any element is moved the value of limit is copied to discrete.neglected.limit, and all sub-properties of discrete.expfit are set to []. Subsequent execution of the showd() and do_exfit() methods will ignore the neglected elements.

input arguments:

limit – positive number

reset() – moves all elements contained in the sub-properties of discrete.neglected to their original places and sets the values of these sub-properties, the discrete.neglected.limit property and the sub-properties of discrete.expfit to [].

showd() – graphically visualizes the results of discretization.

do_expfit() – corrects the results of discretization by exponential fitting. The number of the exponentials and the initial guess of the time constants are equal to the length and values of discrete.average_tau, respectively. The results of the fit are assigned to discrete.expfit.

showexp(selection) – graphically visualizes the results in discrete.expfit if they exist.

input argument:

selection – see the rules described for the input argument of
FOkin.FOkinResults.show(selection).

The FOkin.Sections class

Description

Container for the data created by FOkin.SelectedLambdaOmega.do_sections().

Superclass

matlab.mixin.Copyable

Properties

omega_slice_width – (read only) copy of the omega_slice_width input argument of FOkin.SelectedLambdaOmega.do_sections().

best_lambda – (read only) copy of the best_lambda property of the parent SelectedLambdaOmega object.

best_lambda_1STD – (read only) upper limit of the range of λ within the model mean of the Bayesian optimization problem is less than or equal to its minimum value plus 1 standard deviation of the noise error. For details see the description of FOkin.SelectedLambdaOmega.do_sections().

lambda_slice_width – (read only) copy of the lambda_slice_width input argument of FOkin.SelectedLambdaOmega.do_sections().

best_omega – (read only) copy of the best_omega property of the parent SelectedLambdaOmega object.

best_omega_1STD – (read only) lower limit of the range of ω within the model mean of the Bayesian optimization problem is less than or equal to its minimum value plus 1 standard deviation of the noise error.

info – (read only) struct with a single field of previous, the value of which is the copy of the info property of the parent SelectedLambdaOmega object.

Methods

The constructor of the class is not public.

show() – graphically visualizes the data contained in the object. For details see the description of FOkin.SelectedLambdaOmega.do_sections().

The FOkin.SelectedModel class

Description

Defines the properties and methods common for FOkin.SelectedLambda, FOkin.SelectedOmega and FOkin.SelectedLambdaOmega. The objects of these subclasses are created by the FOkin.select_model() method.

Superclass

matlab.mixin.Copyable

Subclasses

FOkin.SelectedLambda, FOkin.SelectedOmega, FOkin.SelectedLambdaOmega

Properties

name – char array describing the object. Default value is a copy of the name property of the creating FOkin object.

timestamp – (read only) char array describing the date and time of the creation of the object.

lambda_range – (read only) copy of the lambda input argument of FOkin.select_model().

omega_range – (read only) copy of the omega input argument of FOkin.select_model().

CV_spec – (read only) char array describing the applied cross-validation method.

maxiter – (read only) copy of the maxiter input argument of FOkin.select_model().

BO – (read only) BayesianOptimization object returned by the bayesopt() function in the Statistics and Machine Learning Toolbox of MATLAB.

runtime – (read only) the runtime of the Bayesian optimization procedure needed for the model selection in seconds.

info – (read only) struct with the following fields:

weight, t0, fwhm, start – copies of the corresponding properties of the FOkin object invoked its select_model() method.

warm – copy of the warm input argument of FOkin.select_model().

opt – struct with the following fields:

fokin – struct of fields with names and values identical to that of the properties of the corresponding FOkin.options object.

elnet – struct of fields with names and values identical to that of the properties of the corresponding FOkin.optimizer.options object.

previous – copy of the info input argument of FOkin.select_model().

Methods

The constructor of this class is not public.

show () – graphically visualizes the results of model selection.

The FOkin.SelectedLambdaOmega class

Description

Container for the results of FOkin.select_model() if both its lambda and omega input parameters are 2-vectors, indicating a Bayesian optimization for both the λ and the ω hyperparameters.

Superclass

FOkin.SelectedModel

Properties

best_lambda – (read only) the value of the λ hyperparameter at which the Bayesian optimization procedure found the minimum.

best_omega – (read only) the value of the ω hyperparameter at which the Bayesian optimization procedure found the minimum.

best_value – (read only) the value of the objective function (i.e., the value of the meanMSPE output argument of the underlying FOkin.CV() method) of the Bayesian optimization procedure at the minimum found.

sections – (read only) – FOkin.Sections object with empty property values for a newly created object. The properties are assigned by calling the do_sections() method.

Methods

The class has no own constructor.

do_sections(lambda_slice_width, omega_slice_width, show) – makes sections of the objective function of the Bayesian optimization procedure across the point of the minimum and in parallel with the axes of λ and ω. Along these sections the model means, model error bars and noise error bars of the Bayesian optimization problem are calculated in the same way as in the execution of the bayesopt() function. The limit of the range within the model mean is less than or equal to its minimum value plus 1 standard deviation of the noise error is also calculated towards the simpler models [6]. For λ, this direction is towards the higher values, while for ω, towards the lover ones. In addition, a selection of the points where the objective function was evaluated within a defined slice is also executed. These points together the model means, model error bars and noise error bars can be graphically visualized by the show() method of the FOkin.Sections object created by this method and assigned to the sections property.

input arguments:

lambda_slice_width – the minimum width of the range of λ in logarithmic scale within which the evaluated points are selected for a section along ω, unless the value is larger than the total interval of λ. The maximum width is the double of this value, which is achieved if the distances between the point of the minimum and the lower and upper limits of the interval of λ are not less than this argument.

omega_slice_width – the same as lambda_slice_width but applied for ω instead of λ.

show – logical, if true the show() method of the FOkin.Sections object created by this method and assigned to the sections property is invoked on execution of this method.

The FOkin.SelectedLambda class

Description

Container for the results of FOkin.select_model() if its lambda input parameter is a 2-vector and omega is a scalar, indicating a Bayesian optimization for the λ hyperparameter, while keeping ω at a constant value.

Superclass

FOkin.SelectedModel

Properties

best_lambda – (read only) the value of the λ hyperparameter at which the Bayesian optimization procedure found the minimum.

best_value – (read only) the value of the objective function of the Bayesian optimization procedure at the minimum found.

best_lambda_1STD – (read only) The upper limit of the range of λ within the model mean of the Bayesian optimization problem is less than or equal to its minimum value plus 1 standard deviation of the noise error. For details see the description of FOkin.SelectedLambdaOmega.do_sections().

lambda_grid – (read only) if the count_features() method was executed, an array of logarithmically equidistant points spanning the interval specified in the lambda_range property, appended by best_lambda, otherwise is [].

num_features – (read only) if the count_features() method was executed, an array containing the number of features in the solution matrix **X** of GENP/MENP, corresponding to the values of λ in lambda_grid and the single value of ω in omega_range, otherwise is [].

step_left_limit – (read only) if the count_features() method was executed the lower limit of the region of lambda_grid in which the corresponding values in num_features are equivalent to that at best_lambda, otherwise is [].

Methods

The class has no own constructor.

count_features(ngrid, verbose) – calculates the values for lambda_grid and num_features.

input arguments:

ngrid – the size of lambda_grid and num_features.

verbose – logical, if true the number of the actual step of the calculation is displayed on the consol.

show() – graphically visualizes the results of the Bayesian optimization as well as that of the count_features() method if executed.

The FOkin.SelectedOmega class

Description

Container for the results of FOkin.select_model() if its omega input parameter is a 2-vector and lambda is a scalar, indicating a Bayesian optimization for the ω hyperparameter, while keeping λ at a constant value.

Superclass

FOkin.SelectedModel

Properties

best_omega – (read only) the value of the ω hyperparameter at which the Bayesian optimization procedure found the minimum.

best_value – (read only) the value of the objective function of the Bayesian optimization procedure at the minimum found.

best_omega_1STD – (read only) The lower limit of the range of ω within the model mean of the Bayesian optimization problem is less than or equal to its minimum value plus 1 standard deviation of the noise error. For details see the description of FOkin.SelectedLambdaOmega.do_sections().

omega_grid – (read only) if the count_support() method was executed an array of logarithmically equidistant points spanning the interval specified in the omega_range property, appended by best_omega and optionally by zero, otherwise is [].

average_support_size – (read only) if the count_support() method was executed an array containing the average support size in the solution matrix $\mathbf{X}$ of GENP/MENP, corresponding to the values of ω in omega_grid and the single value of λ in lambda_range, otherwise is []. The support of an array is defined here as the subset of its elements the absolute values of which are higher than the value specified in the options.nonzero_limit property of the instantiating FOkin object. The average is taken over the sizes of the supports belonging to the individual columns of $\mathbf{X}$.

step_right_limit – (read only) if the count_support() method was executed the higher limit of the region of omega_grid in which the corresponding values in average_support_size are equivalent to that at best_omega, otherwise is [].

Methods

The class has no own constructor.

count_support(ngrid, include_zero, verbose) – calculates the values for omega_grid and num_features.

input arguments:

ngrid – the size of omega_grid and average_support_size.

include_zero – logical, if true omega_grid is appended by zero.

verbose – logical, if true the number of the actual step of the calculation is displayed on the consol.

show() – graphically visualizes the results of the Bayesian optimization as well as the count_support() method if executed.

The FOkin.RisefitOptions class

Description

Options for the FOkin.risefit() function to determine the t0 and fwhm properties of a FOkin object exclusively from its data.

Superclass

matlab.mixin.Copyable

Properties

individual – logical, if true the elements of both the t0 and fwhm output arguments of FOkin.risefit() will be determined independently, without supposing any correlation among them. If false, the dependence of the elements of both t0 and fwhm on the group_param input argument is modeled by a spline function. For details see the description of FOkin.calc_t0_fwhm(). Default value is 0.

separate norm – logical, if true the columns of the input argument data (corresponding to the different elements of the group_param input argument) are normalized individually by the maxima of the absolute value of their elements before further processing. Default value is 0.

repeat – logical, if true the optimization is executed in two phases. The first phase is executed as usual. Then the elements of the fwhm output argument are averaged and the options.fwhm_fixed property of the underlying FOkin object is assigned to the average value. The second phase is executed with the modified options. Default value is 0.

elnet_lambda – the value to be assigned to the optimizer.lambda property of the underlying FOkin object. Default value is 1.E-4.

elnet_omega – the value to be assigned to the optimizer.omega property of the underlying FOkin object. Default value is 1.E-10.

maxiter – positive integer specifying the number of objective function evaluations as stopping criteria for the underlying Bayesian optimization algorithm.

show_details – value of 0, 1, 2 or 3, determining the level of display for the intermediate results of the calculation as follows: 0 – no display, 1 – console display only, 2 – console display and low level of graphical visualization, 3 – full level display. Default level is 0.

title_text – char array assigned to the name property of the underlying FOkin object and displayed at the beginning of the titles of the plots in graphical visualization.

Publicly available functions

A = FOkin.design_matrix(time, t0, tau, fwhm) – calculates the design matrix for exponential decay convolved by an IRF modeled by a Gaussian. Invoked internally during building and rebuilding of FOkin objects.

input arguments:

time – m-vector, identical to the time input argument of FOkin constructor, preferably with quasi logarithmically equidistant arrangement.

t0 – location of the peak of the IRF in units of the time input argument.

tau – n-vector, containing the time constants in increasing order and preferably with quasi logarithmically equidistant arrangement.

fwhm – FWHM of the IRF in units of the time input argument.

output argument:

A – the $m \times n$ design matrix.

[t0, fwhm, MSE_total] = FOkin.risefit(data, time, group_param, opt, fokin_opt, elnetADMM_opt, elnetPDCO_opt) – handy wrapper around the FOkin.calc_t0_fwhm() method for estimating the value of the t0 and fwhm properties of a FOkin objects from its data themselves.

input arguments:

data, time, group_param – identical to the corresponding input arguments of FOkin constructor.

opt – FOkin.RisefitOptions object for tuning the operation of the function. Default value is a new instance of FOkin.RisefitOptions.

fokin_opt – FOkinOptions object for the underlying FOkin object. Default value is a new instance of FOkinOptions.

elnetADMM_opt – MultiElnetADMMOptions object for the underlying MultiElnetADMM object. Default value is a new instance of MultiElnetADMM.

elnetPDCO_opt – MultiElnetPDCOOptions object for the underlying MultiElnetPDCO object. Default value is a new instance of MultiElnetPDCO.

output arguments:

t0 – $1 \times p$ vector, containing the values to be assigned to FOkin.t0.

fwhm – $1 \times p$ vector, containing the values to be assigned to FOkin.fwhm.

MSE_total – the value of the weighted mean square error calculated for the whole dataset.

[best_lambda, best_omega, best_value, best_lambda_1STD, best_omega_1STD] = FOkin.process_selected(selected, lambda_slice_width, omega_slice_width, show) – multipurposed utility function invoked internally by objects of different classes, but generally not needed by users. See the code in +FOkin\process_selected.m for details.

stop = FOkin.FOkinPlotObjectiveModel(results, state) – internally used by FOkin.SelectModel() for plotting of the results during the iterations of the bayesopt() function in the Statistics and Machine Learning Toolbox of MATLAB. This function visualizes better the position of a 2D model minimum than the original plotObjectiveModel() function of MATLAB, hence in the FOkinOptions class it is set the default plotting function.

References

1. Zimányi L, Sipos Á, Sarlós F, Nagypál R, Groma GI. Machine-learning model selection and parameter estimation from kinetic data of complex first-order reaction systems. PLoS One. 2021.
2. Boyd S, Parikh N, Chu E, Peleato B, Eckstein J. Distributed optimization and statistical learning via the alternating direction method of multipliers. Foundations and Trends® in Machine Learning. 2011;3(1):1-122. doi: 10.1561/2200000016.
3. Boyd S, Parikh N, Chu E, Peleato B, Eckstein J. Matlab scripts for alternating direction method of multipliers. Available from: https://stanford.edu/~boyd/papers/admm/.
4. Chen SSB, Donoho DL, Saunders MA. Atomic decomposition by basis pursuit. SIAM J Sci Comput. 1998;20(1):33-61. PubMed PMID: ISI:000075434800003.
5. Saunders M. Pdco: Primal-dual interior method for convex objectives. Available from: http://stanford.edu/group/SOL/software/pdco/.
6. Hastie T, Tibshirani R, Friedman J. The elements of statistical learning: Data mining, inference, and prediction. 2nd ed: Springer; 2009.