Manual: Simulation Package based on In vitro Databases (SPID)

Jean-Baptiste Poullet
Diana Sima
Jan Luts
Maria Osorio Garcia
Anca Croitor
Sabine Van Huffel

December 18, 2008

Abstract

The goal of SPID is to provide the user with tools capable to **simulate**, **preprocess**, **process** (**quantification** and **feature extraction**) and **classify** in vivo and ex vivo MRS signals. These tools are embedded in a matlab graphical user interface (GUI). (Pre)processing and classification methods can be automatically run in a row using the matlab command line. The command line will be preferred when running automatic procedure on large data sets, while the GUI is more appropriate for checking the results of the different steps of the procedure.

Files saved in a .mat format from the JAVA version of AQSES-GUI can be loaded in SPID.

Pluging in new tools in SPID is much simpler than in the JAVA version of AQSES (AQSES-GUI) [DNLV+07] since it is based on a higher programming language level (i.e., Matlab instead of JAVA and FORTRAN). Moreover, SPID contains many more methods than AQSES-GUI and can be used as well for testing new methods as for comparing existing ones. Most of the parameters are fixed in AQSES-GUI while they can be tuned in SPID, making SPID much more flexible.

In this document, we describe what are the databases in SPID and how they can be used. how to use existing construct new databases Explanations about the databases,

how to create them and how to create simulated data are found in the first section. we describe separately the GUI (SPID-GUI) and the command line tools (TSPID).

Finally, this package can be used on any station supporting matlab (Linux or Windows, not tested on Mac OS or Sun yet). Matlab v7.0 or higher is required.

To cut a long story short, here is a software that should SPID'up your MRS signal analysis.

1 Introduction

This software package has been implemented to work with AQSES v1.0. The parameter names and their units are the same. Thus, signals saved with SPID are readable by AQSES-GUI and vice versa.

The goal of SPID is not to copy AQSES-GUI (even if most of the tools contained in AQSES-GUI can be retrieved in SPID) but to provide the user with a highly tunable GUI. For example, one can choose the maximum magnitude of the ripples when using the maximum phase finite impulse response (MPFIR) developed by Sundin [SVVH$^+$99]. More tools are also available in SPID than in AQSES-GUI. Therefore, we expect a much faster development of SPID than of AQSES-GUI. Note that most of the commands available from the GUI are also available from the command line (runTSPID, see Section 5).

In this document, we describe the different tools available in SPID. In Section 1.1, the directory tree is described. Details about the existing databases and how to create new ones can be found in Section 2. A overview of the graphical user interface SPID-GUI is given in Section 4 and how to use SPID via the command line (TSPID) is explained in Section 5. A simple example to familiarize the user with SPID-GUI is proposed in section 6. Section 7 describes how to plug in new methods in TSPID. Finally, future improvements of SPID are given in Section 8. Note that the list of available methods in TSPID is given in the appendix.

1.1 Directory tree

- DB: baselines, databases, raw_classes, Readme, tables, testdata, waterpeaks
  - baselines: B1_1024_SE63_Seeger2006.mat, ...
  - databases: DB1_invitro1_eqph_PRESS23_2048_11peaks.mat, DB1, ...
  - raw_classes: csf.mat, gbm.mat, ...
  - Readme: DBreadme.tex
  - tables: GBM1.txt
  - testdata: invitro_data, ...
  - waterpeaks: water1_2048_SE63.mat, ...
- Readme_doc: ManualSPID.pdf, ManualSPID.tex, images directory
- Software: matlab files classified in the directories: classification, preprocessing, processing, main, misc, plotting, simulating.

The simulation package contains three main folders:

DB This folder contains all the information corresponding to the databases: how they have been acquired, for which types of signals, for which tumor types, etc.

Readme_doc This folder contains general information about the software. It gives a manual of SPID (as you probably noticed since you are reading this manual).

Software The whole program of SPID is found in this directory.

We describe briefly below the directory `DB`. The software SPID and its tools are described in Section 4 and following.

DB

`baselines` .mat files containing baseline signals: `baseline`, `begin`, `step`, `ndp`, `frequency` are mandatory variables in each .mat file (`baseline` is a time domain signal).

`databases` List of the databases. The databases are stored in the directories DB1, DB2, etc (until now, there is only 1 DB). Each of these directories should contain a set of metabolite signals located in `raw_FIDs`, a full description of how these signals have been acquired is described in a doc or pdf file (ex: SPID/DB/databases/DB1/generate_dbase_manual.pdf). It might also contain the corrected signals (see the directory `corrected components`). However, all files or directories added to DB* directories (* = number) should be fully explained.

`raw_classes` All in vitro data used for constructing new tables for simulations (see directory `tables`) should be stored in this directory.

`Readme` A document gives you the details about the contents of `DB`. It explains, for example, how the tables, the water signals or the baseline signals have been created. In this manual (the one you are currently reading), we explain how to use the databases, the tables, etc, while the document in the `Readme` directory describes extensively what are these files (but will not explain how to use them).

`tables` txt files containing the means and standard deviations corresponding to a set of metabolites, they are constructed from the `raw_classes` directory. Details about how to create these tables can be found in Section 3. Details about the existing tables and how they have been created can be found in the `Readme` directory.

`testdata` This folder contains testing data. These data can be used as reference data to test the software (not done yet). Only *in vitro* data are currently available.

`waterpeaks` .mat files containing water signals: `water`, `begin`, `step`, `ndp`, `frequency` are mandatory variables in each .mat file (`water` is a time domain signal).

2 Databases in SPID

This section explains the goal of the databases, which types of data we can find in SPID, how to use them to generate new simulated data and how to create new data or databases.

The ultimate goal is to generate new simulated signals that will be used in studies where true values for the amplitudes, frequencies, etc, are needed to assess the accuracy of a method. It is therefore necessary to answer the question: what do we need to generate new simulated signals which mimic true signals? A model of the signal is necessary, but how to choose it? In this version, we opted for a relatively naive method which mainly consists of using the AQSES model [PSS$^+$07]. We will detail this below in Section 2.2.

Another goal is to create random data for testing classification methods. We briefly explain how classification data can be generated in Section 2.3.

2.1 What are the existing databases in SPID?

The existing databases are fully described in SPID/DB/Readme/DBreadme.pdf.

2.2 How to use existing databases to construct new simulated signals?

This section explains how to construct new simulated *in vivo* signals. We first introduce the model used to create new simulated signals. This tells us what we need (values for the metabolite profiles, amplitudes, frequencies, etc). How to access this information is given in Section 2.2.2. The use of these databases have to be accompanied by acknowledgments to the group who acquires the data. For example, the Biomedical Magnetic Resonance Research Group Radboud University Nijmegen Medical Center should be thanked in any publication where DB1 has been used. For databases created based on simulations, like DB2, no acknowledgements is needed.

2.2.1 Theoretical aspects

To construct the simulated signals, we use the following equation:

$$S = \sum_{i=1}^{K} \left(met_i \ \Delta a_i \exp\left[(-\Delta d_i + j2\pi\Delta f_i)n\Delta t + j\Delta\phi_i\right] \right) + \alpha_1 B + \alpha_2 W + N \qquad (1)$$

S = constructed signal, K = number of metabolites, the subscript i indicates the metabolite i, met_i=in vitro profile, Δa_i = amplitude correction , Δd_i = damping correction, Δf_i = frequency correction, Δt = time step (=1/sampling frequency), n = number of data points, $\Delta\phi_i$ = phase correction, B = baseline profile, W = water peak profile, N = noise profile, α_1 = constant$*$ $\max(|\text{fft}(\text{signal}_{\text{rpf}})|)/\max(|\text{fft}(\text{baseline})|)$ and α_2 = constant$*\max(|\text{fft}(\text{signal}_{\text{rpf}})|)/\max(|\text{fft}(\text{water})|)$, where $\text{signal}_{\text{rpf}}$ is the signal in the frequency region around the reference peak. By default, we choose an interval of 0.4 ppm around the reference peak, e.g [8.44-0.2,8.44+0.2].

The baseline profile B can be defined by a set of splines or gaussian curves. See [SVH06, SVH04] for more details.

Saturated or unsaturated water signals will typically be obtained by filtering real signals with HLSVD-PRO [LMV$^+$02]. The shape of saturated water signal is unpredictable and might be fairly far from lorentzian shapes. Consequently, it is recommended to use sufficiently high number for the model order in HLSVD-PRO. Typical model orders are 25 for *in vivo* MRS and 30-35 for *ex vivo* MRS signals.

The noise profile is obtained by the following equation

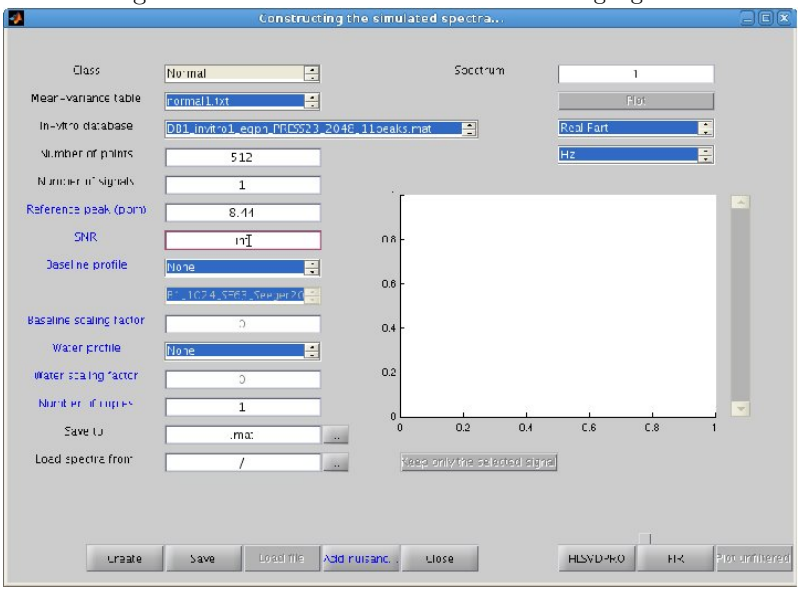
$$N = \sigma\ randn_1(n) + j\sigma\ randn_2(n) \tag{2}$$

where $\sigma = \max(\Re(\text{fft(noise-free signal)}))/SNR$ is the standard deviation of the noise, $randn_1$ and $randn_2$ are random vectors of length n following a standard normal distribution. Note that SNR is defined by the user. We assume the noise to be a circular[1] gaussian white noise.

2.2.2 Using SPID-GUI to construct simulated MRS data

This section describes step by step how to create simulated MRS signals from the GUI interface (SPID-GUI). An illustration of the menu is showed in Fig. 1

Figure 1: The GUI "menu_db" for simulating signals.



The different steps that should be followed are:

Step 1: Open matlab and run SPID from the directory SPID/Software
Step 2: Launch the menu `menu_db` from the menu tab "Simulating/Create simulated signals"
Step 3: Choose the parameters (see Section 2.2.3) of your signals and where you want to save them
Step 4: Push on the `Create` button (the signals are automatically saved in a mat file)
Step 5: Visualize the just created signals by pushing on the "plot" button and play with the slider if you have several signals to visualize

2.2.3 The parameters

Class: It refers to the class type ('Normal','GBM', 'MNG', etc). That means that you create simulated signals for a special class. If you want to create signals for different classes, you have

[1] equal variances for real and imaginary part

to repeat the whole procedure from step 3 to the end (see above). When you modify the current class (by clicking on it!), the set of available tables (see "the tables" below) are instantly changed.

Mean-variance table: It refers to the mean-variance tables. These tables are text files which contains the mean and the variance for each parameter of each metabolite. The means are the mean corrections to be applied to the in vitro signals as defined in Equ. 1. For example a ten-metabolite table will be of size 10x8. Each row corresponds to one metabolite. The first(resp. last) four columns corresponds to the mean (resp. standard deviation) values for amplitude, phase, damping and frequency corrections, respectively. Note that the first line contains the names of the metabolites. The third name will corresponds to the third line of the matrix (or the fourth line in the file).

In vitro database: It refers to the in vitro metabolite profile. You can find them in the directory `SPID/DB/databases` under a .mat format.

Number of points: It refers to the number of data points for your simulated signals. If this number is above the number of points of the in vitro data, the software will fill in the constructed signals by zeros in the time domain.

Number of signals: It refers to the number of signals that are created in one unique file (see "save to" below).

Reference peak: It refers to a reference peak which will be used for normalizing the noise, water and the baseline profiles. The water peak can be used if no other reference peak was defined during the acquisition. The reference peak is often characterized by a high peak in a frequency region far away from the usual region of interest, i.e [0.8,4.4] ppm.

SNR: It refers to the signal-to-noise ratio as described in Equ. 2. To construct a noise-free signal, the user will type 'inf' (infinity) in the edit box.

Baseline profile: It refers to the baseline profile which can be gaussians, splines or other. If the user choose "other", he has to choose between the available baseline signals which are stored in `SPID/DB/baselines`.

Baseline scaling factor: It refers to the constant α_1 in Equ. 1.

Water profile: It refers to the water profile. The water profiles or signals are contained in `SPID/DB/waterpeaks` as described above.

Water scaling factor: It refers to the constant α_2 in Equ. 1.

Number of copies: The goal of this field is to generate monte-carlo simulations (signals which differ only by their noise components) from a unique signal. If the field `Number of signals` indicates a number larger than 1, the error message pops up: "The signal contains more than 1 spectrum. This function is aimed to produce monte-carlo simulations from a unique signal." Number of copies will indicate the number of signals that are created from the unique signal. To make sense it is necessary to choose SNRs different from infinity, otherwise all the signals will be similar.

Save to: It refers to the filename where you want to save the created signals. The complete path must be written (e.g, on Windows machines, c:\mydata\NMRsignals\shortecho\PRESS\simulated_signals1.mat) This file will contain a list of variables: ndp, step, begin, B0, nucleus, frequency, refpeak, signal, amplsimul, phassimul, dampsimul, freqsimul, classt, table, invitrof, SNR, baselinet, baseline_simul, scaleB, watert, water_simul, scaleW. This variables corresponds to the number of points, the time step, the begin time, the external field, the nucleus (`e.g.`, -1 for Hydrogen), spectrometer frequency, the value in ppm where

the reference peak is located, the list of signals, the corrected amplitude, phase, damping and frequency, the protocol used, the class type, the mean-variance table, the in vitro database, the signal-to-noise ratio, the baseline type (Gaussian, Splines, etc.) the baseline profile , the scaling factor for the baseline, the water type (.mat file), the water profile and the scaling factor for the water, respectively. Note that signal will be of size (number of signals x number of points). These variables allow the user to retrieve where his simulations come from.

Load spectra from: The user can load already simulated spectra. In that respect, he will fill in this text field by the name of the set of spectra he wants to load, and will then push on the button "Load file".

HLSVDPRO filtering: The user can apply a HLSVDPRO filtering [LMV+02] on the the following region [-499,-280] and [-32,499] Hz.

FIR filtering: The user can apply a FIR filtering on the the following region [-499,-280] and [-32,499] Hz. This will be done with Tomas' method [SVVH+99].

Add nuisance components: The user can add nuisance components to the current signals. The same nuisance components will be added to each signal except for the noise which is by nature random. The components that will be added depends on whether the baseline profile and water profile fields are activated or not and whether the noise text field is different from `inf`. Fields in the GUI related to this button are highlighted in blue (so is this button).

2.2.4 Example

If you want to create simulated signals with this package, you first have to answer to these questions: which protocols am I interested in?, for which classes do I want simulated signals?, which type of simulation do I want (*i.e*, which procedure do you choose to create the tables, or equivalently which mean-variance tables do I need)?

Different steps to follow (important to respect the order):
Step 1: Choosing the class. Suppose you want to generate signals from the classes GII and GIII to study the accuracy of your method on signals from each class. You have to first create a set of data for gII and then repeat all the steps for GIII afterwards. Note that the "mean-variance tables" list is updated.
Step 2: Choosing the mean-variance table. The mean-variance tables are described in `SPID/DB/Readme`. Suppose your goal is to mimic real signals from GII. You will likely choose a mean-variance table which is generated by a maximum number of signals. However, say that the available mean-variance tables seem to overestimate the variance for NAA, you might want to change the table manually and create a new table (see Section 3 to learn how to create a new table). On the contrary, you also might be glad with the current tables (you think that the procedure used to estimate the means and the variances is good enough) and in that case you will directly select the tables from the available tables. Step 4: Choosing the number of points, number of signals you want to create, the baseline, the water, the noise profiles that you want to add.
Step 5: Choosing where you want to save your signals and create them. Plot the just created signals by pushing on the plot button or by using the slider.

2.3 How to create classification data?

Two possibilities are available: either you want to create random classification data or you want to use existing datasets.

Creating random classification data
To create random classification data, go to `Simuating/Create data for classification`. Choose the type of data you want to create, it is only possible for the moment to create random data from normal distributions. Push on "Create". The window illustrated in Fig. 2 pops up and let you choose several parameters: the number of classes, the number of features, the number of data per class, the highest mean value, the level of ressemblance between classes. You can choose balanced data and then you can just type one number in the text field "number of data...". If you want different sample sizes between the classes, a vector is needed.

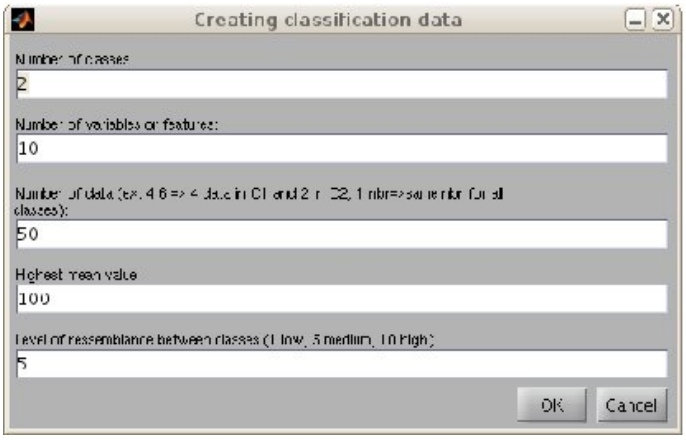


Figure 2: Pop up window when creating random classification data based (variables follow normal distributions).

The data are constructed as follows:
Step 1: Means are chosen randomly for class 1 (assuming a uniform distribution [1,highest mean value])
Step 2: Standard deviations are chosen randomly for class 1 (assuming a uniform distribution [1,just calculated Means/2])
Step 3: Means are chosen randomly for class 2 such that they differs from means of class 1 by `1+randn(1,nvar)/r`, where `nvar` is the number of features and `r` is the ressemblance level between classes.
Step 4: Standard deviations are chosen randomly for class 2 such that they differs from the standard deviations of class 1 by `1+randn(1,nvar)/r`, where `nvar` is the number of features and `r` is the ressemblance level.
Step 5: Steps 3 and 4 are repeated for all classes (if "number of classes" is larger than 2)
Step 6: Values are generated for the computed means and standard deviations for the different classes
Step 7: The data are stored in a matrix `procres.scores`, means and standard deviations are stored in `procres.misc.means` and `procres.misc.std` (matrices of size [number of data in total, number of features]). The class labels are stored in `classopt.classtype` (vector of size [number

of data in total]).

Convert existing data to a SPID format
It is also possible to convert existing data to a SPID format. For the moment, 2 datasets are available: "Wine Recognition Database" and "Wisconsin Breast Cancer Databases". Both datasets come from the the UCI Machine Learning Repository. More information can be found on the website `http://mlearn.ics.uci.edu/MLSummary.html`. These datasets are located in SPID in `SPID/Software/simulating/datasets`. In this directory, we find two file types, `.data` and `.names`, the first for the data themselves and the second for the description of the data. This conversion tool converts the data in the text file (`.data`) into a SPID type format: the dependent variable is stored in `classopt.classtype` and the independent variables or features are stored in `procres.scores`.

3 How to create new data in the databases

As already described above, there are several types of data which can be added to the databases. We will go through the different types of data that can be created and detail how to create them.

`baselines` The baseline signals are contained in mat files. To create a new baseline signal, the user must respect the following steps:
Step 1: Create a mat file which contains the variables `baseline` (baseline signal in the time domain), `ndp` (number of data points), `frequency` (spectrometer frequency in kHz), `begin` (begin time in ms), `step` (time step in ms). This mat file must be stored in `SPID/DB/baselines` and the name of the file must begin with "B*" (* is a number). The name should be choosen adequately.
Step 2: Update the `DBreadme.tex` file in `SPID/DB/Readme`. The description should allow to reconstruct the signal.

`databases` Here are described the metabolite profiles. Here are the steps to add new metabolite profiles:
Step 1: Create a new directory DB* (* = next number)
Step 2: Copy the acquired files (if they are *in vitro* signals) in `raw_FIDs`
Step 3: add a pdf or doc file which fully describes how the signals were acquired
Step 4: Create a basis set of metabolites and store it in a mat file, copy this file in `SPID/DB/databases`. The name of the file must start with DB* and will chosen adequately.

`raw_classes` To create new sets of signals from a class:
Step 1: Create a mat file which contains the classical `signal`, `begin`, `step`, `ndp` and `frequency` and store it in this directory (`raw_classes`)
Step 2: Update the `DBreadme.tex` file in `SPID/DB/Readme` accordingly.

`tables` The tables can be generated either automatically via the menu tab `Simulating/Create mean-variance tables` or manually. To create a mean-variance table manually:
Row 1 = component names
From row 2 to the end: matrix with mean and standard deviation values for the metabolites in the first row. The first(resp. last) four columns corresponds to the mean (resp. standard deviation) values for amplitude, phase, damping and frequency corrections, respectively.

`testdata` There is no strict rules for this directory provided that all files are fully described in `DBreadme.pdf` in the folder `DB/Readme`.

`water` The water signals are contained in mat files. To create a new water signal, the user must respect the following steps:
Step 1: Create a mat file which contains the variables `water` (water signal in the time domain), `ndp` (number of data points), `frequency` (spectrometer frequency in kHz), `begin` (begin time in ms), `step` (time step in ms). This mat file must be stored in `SPID/DB/water` and the name of the file must begin with "W*" (* is a number). The name should be choosen adequately.
Step 2: Update the `DBreadme.tex` file in `SPID/DB/Readme`. The description should allow to reconstruct the signal.

4 SPID-GUI: SPID Graphical User Interface

The GUI (SPID-GUI) is launched by typing `spid` in the Matlab command prompt. A window similar to the one in Fig. 3 shows up. Using SPID requires a matlab version higher than v7.0. Note that all methods available in SPID are not available in SPID-GUI. In order to be able to use all methods, one will have to use runTSPID (see Section 5 for more details).

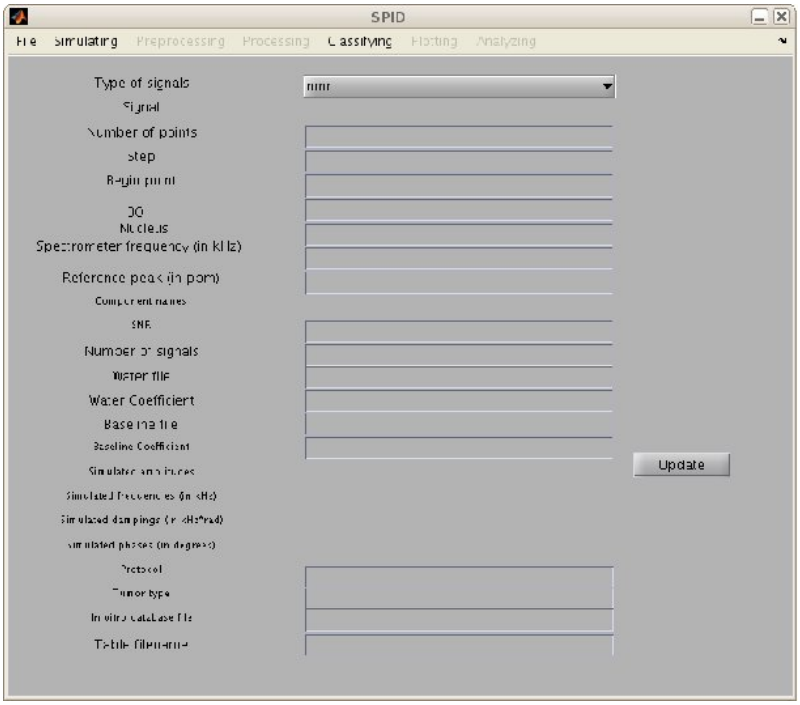


Figure 3: The GUI of SPID (SPID window).

Before entering the menu details, one should know that this program is based on a few global variables which are `signal`, `begin`, `step`, `frequency`, `ndp`, `classopt`, `procres` and `classres`:

`signal` $M \times N$ matrix signals (M signals, N points) in the time domain.

`begin` Begin time in ms (scalar), often set to 0.

`step` Step in ms between two time samples (scalar)

`frequency` Spectrometer frequency in kHz (scalar)

`ndp` Number of points in each signal (scalar), it must be identical to N

`classopt` Classification options (structure variable, see 'structure' in matlab for more details). It essentially contains the variable `classtype` (`classopt.classtype`) which is a vector of class labels (this vector must be a cell aray or a double array). Example: `classopt.classtype` = {'MEN','GII','MEN','GIII','GII','MEN'} or = [1 2 1 3 2 1].

`procres` Processing results. It contains the variable `scores` (matrix $M \times F$, where F is the number of selected features) and `misc`. `procres.misc` is a structure variable containing itself different features or variables. These features are not standardized and since they differ from one method to the other and they are not used for classification, they will not be detailed here. The variables `procres.scores` and `classopt.classtype` are the inputs of the classifier.

`classres` Classification results. It contains different statistics: misclassification (`classres.res.misclassification`), accuracy (`classres.res.percent_correct`), area under the curve (AUC) (`classres.res.auc`), etc (an extensive list is not given here yet since this part is under development).

Note that all these variables are not mandatory for loading signals; `classopt`, `procres`, `classres`, `frequency` can be missing. However, `classopt` and `procres` will be needed by all the classification methods.

4.1 Menus

Several menus are available, each devoted to a certain type of action.

File Classical tools for opening and saving files (the MRS signals are stored in these files).

Simulating Menu for creating new simulated signals.

Preprocessing Tools for preprocessing the signals.

Processing Tools for processing the signals or viewing the processing results.

Classifying Under development. This will include tools for classifying the data.

Plotting Menu for plotting the signals under investigation.

Analyzing Menu for analyzing the data. Under development.

4.1.1 Menu "File"

This menu contains the following items:

Save signal Save the signals under investigation. The parameters appearing in the main window (`spid.fig`) are saved.

Load signal Load a signal of a set of signals. Only text files from jMRUI3.0 [jMR] and mat files are readable. The mat files must contain the variables `ndp`, `begin`, `step` and `signal`. If `frequency` (spectrometer frequency) is not defined, it will be set at 63130 kHz.

Select spectra This allows to reduce the set of spectra under investigation. For example, to select the spectra 5 to 10 and 15, one should type "5:10 15" (matlab's vector format).

Add signals Add signals to the existing ones and put them at the end of the existing `signal` matrix.

Save figure Save the figure that is located in the `menu_plot` window.

Load/Create/Run template It allows to load, create or run a template. A template is a text file which contains the list of methods that should be applied to the signals or to some data matrix (more details can be found in Section 5).

4.1.2 Menu "Simulating"

See Section 2.2 for more details.

4.1.3 Menu "Preprocessing"

A set of spectra must be loaded (see above) before using the following tools. This menu contains the following items:

ECC (Klose's) Perform eddy current correction using Klose's method. The user is asked to load the water signal.

Filtering The main goal is to filter out the water resonances, but any part of spectra can also be removed.

Phase correction Manual or automatic (ACME method [CWGG02]) phase correction.

Normalization The signal is divided by the L2-norm of the spectrum in the frequency interval [0.25 4.2].

Eretic normalization Each spectrum is divided by the corresponding amplitude estimates of the reference peak when taking the reference peak of the first spectrum as basis signal in AQSES. More details can be found in the appendix.

Baseline correction The product of the signal and an Apodization function is subtracted from the signal (in the time domain).

Zero filling Add zeroes at the end of the signals.

Aligning Aligning spectra with respect to a reference singlet or doublet from which the user known the exact location.

Auto aligning Automatic procedure to align the signals. However, the user must anyway give a starting value for the singlet or doublet. This starting value does not need to be as accurate as the one used in `Aligning`.

Truncation Suppress either the first (begin) or last (end) points of the spectra.

Check redundant signals Check whether all the signals are different (the norm of the absolute values of the signals is taken as criterion to avoid considering a signal and its shifted version as different).

Resampling Resample the data (the parameters given by the user are the current and desired steps). This method is the matlab `resample` function.

4.1.4 Menu "Processing"

The menu contains the following items:

Quantifying (AQSES) Quantification with AQSES. Most of the parameters are tunable. The results can be saved in one file. Several signals can be processed. The basis set has to be chosen contrarily to the rest of the parameters which can remain by default. Evolution of the fitting procedure can be obtained by checking 'Plot at each iteration'.

Quantifying (HLSVD-PRO) Quantification with HLSVD-PRO. Passband and model order are the only parameters that the user has to choose.

View results The results of the quantification can be visualized in this window. The estimates of the parameters are displayed in the left panel and the several plots are shown in the right tables (from top to bottom: residuals, baseline, corrected metabolite profiles and filtered estimated vs filtered original signals).

Database normalization This tools allows to normalize several basis sets. One basis set (set of metabolite profiles) is chosen as reference basis set. The other will be normalized with respect to the amplitude estimates obtained when fitting each metabolite profile of the basis sets with the corresponding metabolite profile of the reference basis set. Necessarily the number of metabolite profiles must be the same in all the basis sets.

Peak Integration The peak integration menu allows the user to choose several frequency intervals in which the integrals of the spectra will be calculated. The results are saved in a mat file.

4.1.5 Menu "Classifying"

The menu file contains the following submenus:

Supervised Supervised Methods: only LS-SVM is available at the moment.

Unsupervised Unsupervised (under construction)

5 Command line tools (TSPID)

This section describes how to run (pre)processing or classification methods from the command line. The main idea is to have a text file which lists the different methods we want to apply on some data. This file is called the template. TSPID tools are the tools available from the command line using `runTSPID.m`. TSPID stands for template-SPID.

5.1 Why such a tool?

Why using the command line (TSPID) when a nice GUI is available? There are several reasons that we detail below.

Advantage of TSPID with respect to SPID-GUI

- TSPID does not need the matlab java virtual machine or to use matlab in java mode. You save thereby memory (RAM or cache), resulting in a likely faster process.
- TSPID runs a stack of methods in a row, while only one method at a time can be run with SPID-GUI. With the latter, the user will have to wait until the process ends before starting a new process.
- More methods are available from TSPID (see appendix).
- Adding a new method in TSPID is much easier and faster than adding a new method in SPID-GUI.
- Some methods are only available from TSPID. A list of the available methods in TSPID is given in Section 8. This list of methods can be retrieved in SPID-GUI by selecting Load/Create/Run template in the File tab.

Disadvantage of TSPID with respect to SPID-GUI

- By definition, there is no plotting tools in TSPID, and so no simple way to check the (pre)processed signals. In SPID-GUI, you can plot the signals, the preprocessed signals, but you can also view the quantitation results.
- Some tools are only available from SPID-GUI: for example, viewing the filter properties (see preprocessing tools or Section 6).

In summary, both tools should be used alternatively, SPID-GUI for checking the original signals, the results of the different steps (preprocessing, processing, classification), while TSPID should be used as a black box. SPID-GUI can also be used to explain some results. For example: bad classification results are obtained with AQSES, while good ones are obtained with peak integration, the 'View Results (Quantification)' may show that the fit was not good in AQSES and another database of metabolite profiles is maybe needed.

5.2 How to use TSPID?

Using TSPID is very easy. You only need one command line which uses the function `runTSPID.m` (see the basic examples below). The most simple example is

```
%my program 1
template = 'mytemplate.txt';
loadsignals = '/users/sista/username/mydata.mat';
runTSPID(template,loadsignals)
```

where the selected preprocessing, processing and classification methods are by default applied and the corresponding results are saved in `preproc.mat`, `proc.mat` and `class.mat`, respectively. Another example is given below.

```
%my program 2
template = 'mytemplate.txt';
loadsignals = '/users/sista/username/mydata.mat';
savepar.preprocFlag = 0;
savepar.procFlag = 1;
savepar.classFlag = 1;
savepar.preproc = fullfile(pwd,'preproc.mat');
savepar.proc = fullfile(pwd,'proc.mat');
savepar.class = fullfile(pwd,'class.mat');
runTSPID(template,loadsignals,savepar)
```

where only the selected processing and classification methods are applied even if preprocessing techniques were selected in the template file.

`runTSPID.m` takes up to 3 parameters as input:

template: `mytemplate.txt` is the template file which can be written or edited manually, but will be more likely created automatically using the template menu `menu_template.m` from SPID-GUI.

data: `mydata.mat` contains the data that will be used in the preprocessing, processing or classification steps.

parameters: `savepar` variable is a structure variable

Before explaining how to create a template file it is important to distinguish hyperparameters from input parameters. We call hyperparameter a parameter specific to a method. For example, a method which keeps the first x time domain points of a signal (see truncateend in the appendix) will have x as hyperparameter and the signal as input parameter. The truncated signal will be the output parameter.

5.2.1 Creating a template file

To create a template file, there are two ways, either you edit a txt file with your favorite text editor, or you use SPID-GUI. Usually, the user will use SPID-GUI but he might also be interested in changing the value of one or more parameters in an automatic way (for example to test several hyperparameters in a preprocessing method) and will then edit directly the txt file. How to create a template file with SPID-GUI is illustrated in Section 6.

Editing a template file
An example of template file is given below.

```
1  ;  alignsignal  ;  "Signal alignment"  ; "..."
33  ;  centerfreq  ;  1.15  ;  "Current center frequency in ppm of the reference peak"  ;
33  ;  targetfreq  ;  1.47  ;  "Target frequency in ppm of the reference peak"  ;
33  ;  doublet  ;  1  ;  "Doublet if 1, singlet if 0"  ;
1  ;  ECCKlose  ;  "Eddy Current Correction with Klose's method"  ;
33  ;  watersigfile  ;  {'watersignal1.mat' watersignal2.mat'}  ;  ".mat files..."  ;
2  ;  peakintegtempl  ;  "Peak integration"  ; "..."
33  ;  ir  ;  1  ;  "Data type (1=SE in vivo,2=LE in vivo;3=ex vivo...)"  ;
```

```
33  ;  freqmatrix  ;  ''  ;  "Intervals (ex: [2.00 2.04 3.00 3.04 3.18 3.24])"  ;
33  ;  compNames  ;  {}  ;  "Names of intervals (metabolites)"  ;
3  ;  LDAtempl  ;  "LDA"  ; "..."
33  ;  validtype  ;  1  ;  "Validation method (LOO=1;L-fold CV=2;Random Split=3)"  ;
33  ;  L  ;  10  ;  "Number of folds in L-fold cross validation"  ;
33  ;  dispplot  ;  0  ;  "Display plot (1: predicted, 2:true)"  ;
```

The template must strictly follow the protocol explained below. There are two types of lines, the method lines and the parameter lines. The numbers 1, 2 and 3 at the begin of the method lines identify the type of method: 1=preprocessing method, 2=processing method, 3=classification method. The name of the method is given as the third element of the line (*e.g.*, "Signal alignment") while the corresponding `.m` file (the method interface file) is given in second position (*e.g.*, `alignsignal`). The 4th element (denoted here "..." for sake of space) is a more complete description of the method. The number 33 at the begin of the parameter lines indicates that this line defines a hyperparameter (parameter specific to the method itself, see Section 5.2). The second and third items (items are separated by ';') denote the parameter name (*e.g.*, centerfreq) and its value (*e.g.*, 1.15), respectively. The fourth items identifies the parameter. Let's consider the example given above. Two preprocessing methods (signal alignment and Eddy current correction with Klose's method), 1 processing method (Peak Integration) and 1 classification (LDA) method have been selected. To align the signals, alignsignal.m will be called with, as hyperparameters, the current center frequency (in ppm), the target frequency (in ppm), the algorithm option (based on finding a doublet = 1 or a singlet = 0). Note that alignsignal.m also needs input parameters like the signal, but this is given in the data file (`mydata.mat`) and not in the template file (for more details, see Section 5.2.2). `runTSPID.m` apply sequentially the methods in the same order as they appear in the template file (`ECCKlose.m` will be called only when `alignsignal.m` has ended).

5.2.2 Creating a data (.mat) file for TSPID

As one might expect, classification methods will not take the same inputs (*i.e.*, the same mat file) as processing methods. In this section, we describe the structure of the data for each type of methods. Note that if some input parameters are mandatory for some methods, they are never forbidden. In other words, it is never problematic to have more parameters than necessary in the mat file except that more memory will be used by matlab. The name of the input parameters should be strictly respected except for `signal` (time domain signal, input parameter) which will be recognized by `ndp` (the number of data points): `signal` will be the matrix with `ndp` columns. The format of the input parameters should also be kept. A description of the input parameters, with their corresponding formats and for which types of methods they are mandatory, is given in Table 1. Most of the variables have been already defined in Section 4.

Any other variable is optional. For example, `procres.misc` will not be used by any types of methods (as explained above), but will give some information to the user. A detailed description of the method inputs/outputs is given in the appendix (see Section 8).

6 A small walk in SPID-GUI

As already explained, SPID-GUI is meant to be used to check the results or the original signals for instance. In this section, we describe how to use SPID-GUI through a simple example.

Input parameters	Methods	format
`signal`	PP,P,C	matrix (1 signal per row)
`begin`	PP,P,C	scalar
`step`	PP,P,C	scalar
`frequency`	PP,P,C	scalar
`ndp`	PP,P,C	scalar
`classopt.classtype`	(P),C	vector
`procres.scores`	(P),C	matrix (1 data per row)
`classres`	-	structure (see Section 4)

Table 1: Description of the input parameters. The methods for which they are mandatory are reported in column 2 and their format in column 3. There are 3 types of methods (preprocessing, processing and classification methods denoted by PP,P and C, respectively. "-" means that this variable is never mandatory whatever the method. (P) means that the parameter is used by some processing methods but not all. More details about the methods that require these variables is given in the appendix.

Type `spid` in the matlab command prompt to launch SPID-GUI. As detailed in Section 4.1, several menus are at user's disposal. We are going through the most common tools with a simple example.

6.1 An example

Suppose one wants to classify *in vivo* MRS data.

Reading data
The data are Siemens data. Siemens data are not readable in SPID and preliminary conversion to a txt file is required (only text files with jMRUI3.0 format and mat files are readable from SPID). The first thing to do is so to open the Siemens data with jMRUI and export them to a txt file, which will be then readable in SPID-GUI using `Load signals` in the `File` menu tab. At this point, the user should see some information in the main frame (SPID frame) about the signals to be processed (an example is given in Fig. 4). In this frame, the second half of the parameters (from `SNR` to `Table filename`) are parameters for home-made simulated signals (see Section 2.2 for more details). The displayed values of the parameters should be checked before proceeding to the next step.

Checking signals
It is recommended to check the loaded signals to identify whether signals should be discarded (acquisition issues) or should be reloaded (problem when loading the signals in SPID or in jMRUI3.0). The common way to check the signals is by plotting them. In order to plot the signals, we need to open the plotting window `menu_plot(Plotting/plot2D)`, and push on the "Plot" button. Several functionalities are available in the plotting window. Let's have a quick overview of the main functionalities. Signals can be plotted in the time (in ms) and frequency domain (in Hz or in ppm); in real, imaginary or absolute values; within some bounds (in both X and Y axis); with an offset. It is also possible to plot several signal simultaneously, to plot the filtered version (one needs first

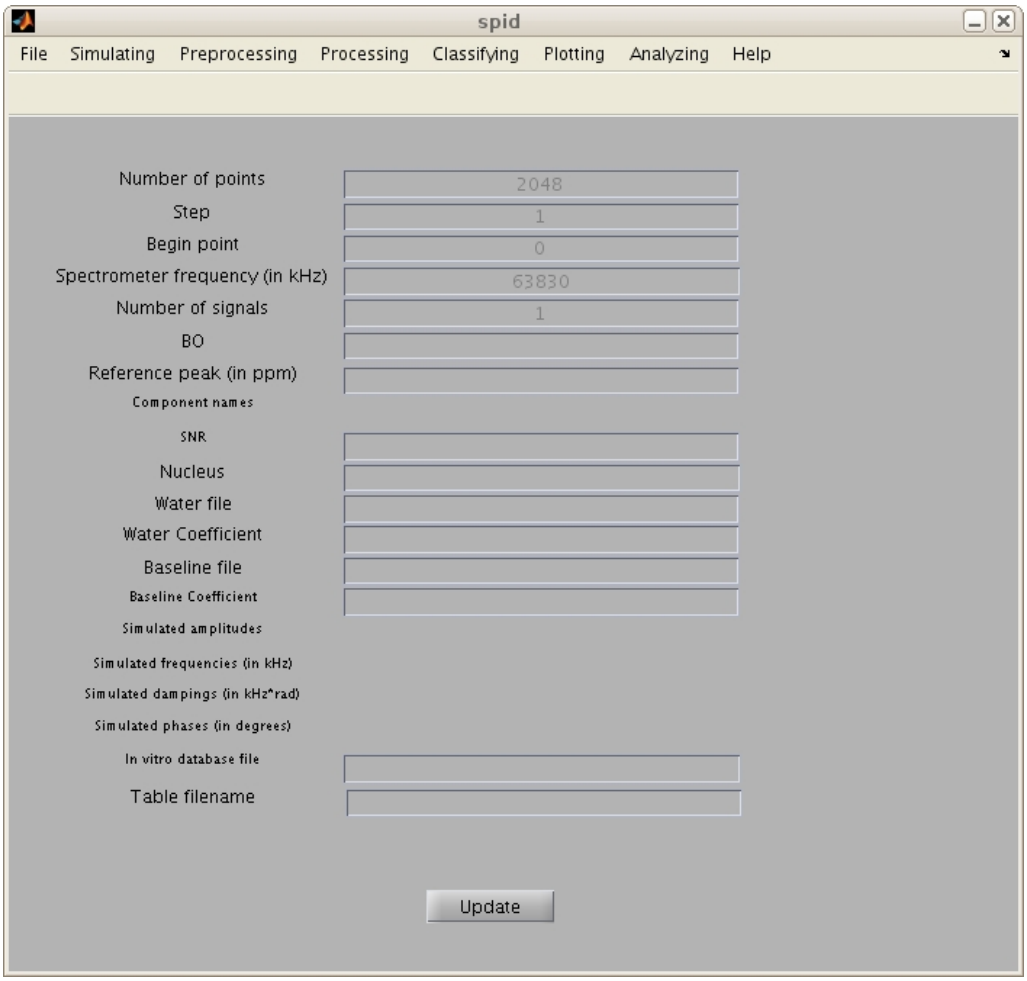


Figure 4: The GUI of SPID after loading data (SPID window).

to filter (see preprocessing tools) the signals), to plot the difference between the original and the filtered version of the signals and to plot the truncated version of the signals when removing a certain number of time domain begin points. It is also possible to save a figure using `Save Figure` in the `File` tab of the SPID window. An example of in vitro database is given in Fig. 5 using the multispectra option and an offset to make the spectra visible.

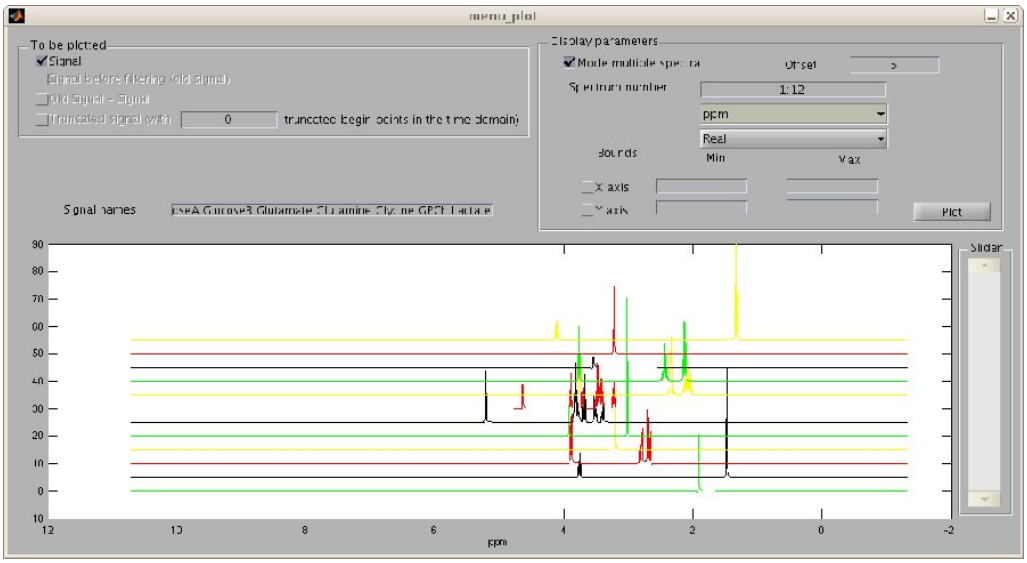


Figure 5: The plot window of SPID.

Preprocessing signals

Once the signals have been checked, they can be preprocessed. Suppose we want to align the signals and filter them using the maximum-phase FIR filter proposed by Sundin *et al.* [SVVH+99]. To align the signals, one should define a reference doublet (Ala or Lac are usually taken as reference peaks for HR-MAS data) and enter the current and desired frequencies for the doublet frequency center. After pushing OK, the user can monitor the progress of the method: "Aligning..." is displayed at the prompt until the method ends, normally by displaying "...done" at the prompt. However, the method may not find any doublet and will then give a warning message at the prompt. It is highly recommended to check after each (pre)processing steps that the signals have been correctly processed. `alignsignal.m` can yield wrongly aligned signals, especially if there is no doublet or if the current frequency entered by the user do not match the actual current frequency. Aligning signals from different machines of acquisition protocols might be difficult to do at once. It is often preferable to align the signals of each machine or acquisition protocol separately and eventually merge them. Recall that it is only possible to merge signals (`File/Add signals`) from different mat files if the number of points per signal (`ndp`), the begin points (`begin`) and the time step (`step`) are identical. Once the aligned signals have been checked it is time to filter them. To filter the signals we open the `menu_filter window` (Preprocessing/Filtering/Filters). The filter coefficients can be saved and the magnitude response can be plotted; one can also use the matlab `fvtool` to analize the filter. Once the signals have been filtered, they can be visualized in the plotting window (using the "Plot" button). One can also plot the signal before filtering as explained above.

Processing

We decide to use AQSES [PSS+07] to quantify the preprocessed signals and get some estimated concentrations (in arbitrary values) of some metabolites. To quantify the signals with AQSES we can use the processing tool in SPID-GUI (`Processing/Quantifying (AQSES)`). An illustration of the AQSES quantitation window is shown in Fig. 6. Most of the hyperparameters in AQSES are tunable but the only mandatory parameter is the filename of the database. More information about the tunable parameters can be found in [SVH04, SVH05a]. The results are saved by default in .mat if the "Save results to" text field is left empty. The results can then be visualized in a separate window (`Processing/View results (Quantification)`) similar to the one in the java version of AQSES (see Fig. 7). On the left hand side, the amplitude, frequency, phase and damping estimates of the metabolites. On the right hand side from top to bottom, the residual plot, the baseline plot, the individual corrected metabolite profile plot and the plot of the modeled filtered signal vs the original filtered signal plot. The plots in this figure can also be saved via the last tab (`File/Save Figure`) of the **same** figure. Note that `File/Save Figure` in the SPID window will only save the figure currently displayed in the plotting window.

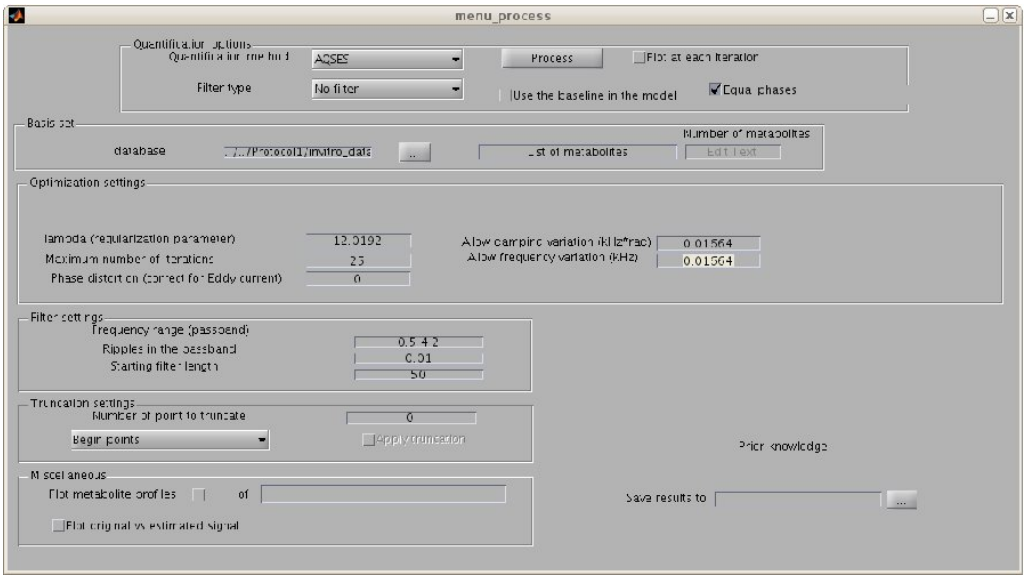


Figure 6: The AQSES quantitation window.

Classifying

Suppose we want to classify the data (in our case, the data are the estimates of the metabolite amplitudes) using LS-SVM with a Leave-One-Out (LOO) validation scheme. In order to classify the results, we decide to use the template tool (`File/Load Create Run template`). Note that using the `Classifying` menu is also possible. Preprocessing and processing methods could also have been used via the template tool. In the template window the methods are ordered by method types (preprocessing, processing or classification methods). We select LS-SVM and LOO as validation tool in the hyperparameters. The hyperparameter window pops up when pushing on the corresponding "Parameters" button. An illustration of what the user should have is shown in Fig. 6.1. The next steps are to save the template under a txt file and run the saved template. The

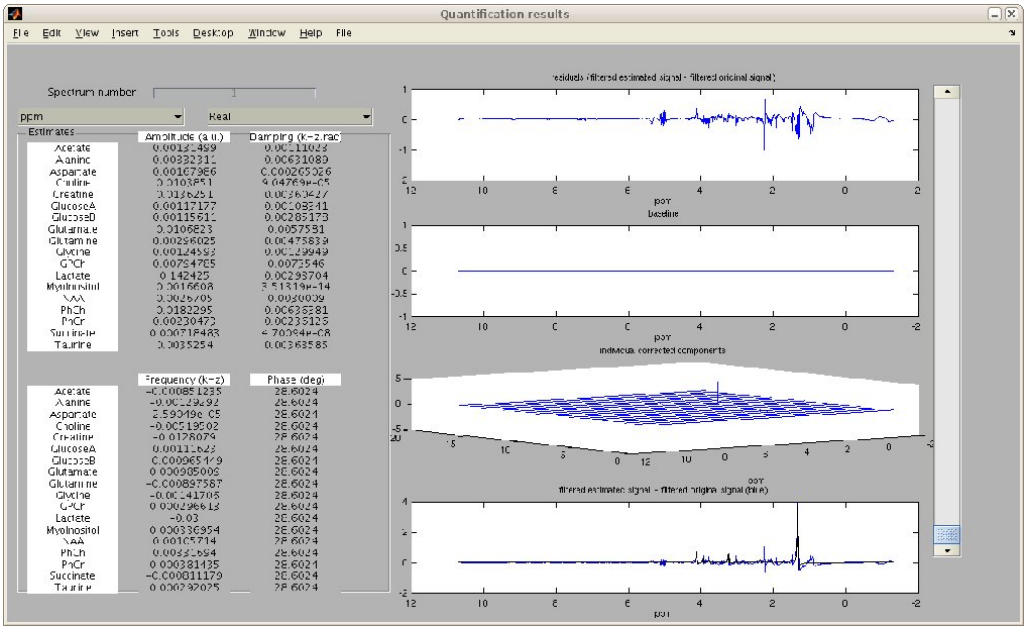


Figure 7: The quantitation result window.

template tool will use the global variables as input (in this case `classopt` and `procres`) and need therefore that these variables exist. The user will take care not to modify these variables before classifying the data. Adding new signals to the current ones will automatically change some of the global variables and should be avoided. If necessary, it is always possible to save the global variables via the SPID window (`File/Save signals`). This not only saves the signals but also the global variables detailed in Section 4.

6.2 A few more tools

Visualizing FIR filter features
To analyze FIR filter characteristics, one can open the `menu_viewfilter` window from the menu tab Preprocessing/Filtering/View FIR Filter. The window is illustrated in Fig. 6.2. One can plot the signal under investigation, the filter frequency magnitude response, open the matlab tool `fvtool` (zero-pole plots, magnitude response, phase response plots, etc), or the magnitude response when filtering lorentzians instead of pure cosinusoids as it is the case when plotting the common frequency magnitude response.

Database normalization
It is also possible to normalize metabolite basis sets. Suppose you want to use compare metabolite concentration estimates from basis sets coming from different spectrometers. A comparison is only possible if the metabolite profiles are normalized. If the databases (or basis sets) are sufficiently similar, it is possible to normalize one with respect to the other by using AQSES. The idea is to fit with AQSES each metabolite profile from one basis set to its correspondent in the other basis set, and then to divide each metabolite profile from the first basis set (basis set to normalize) by

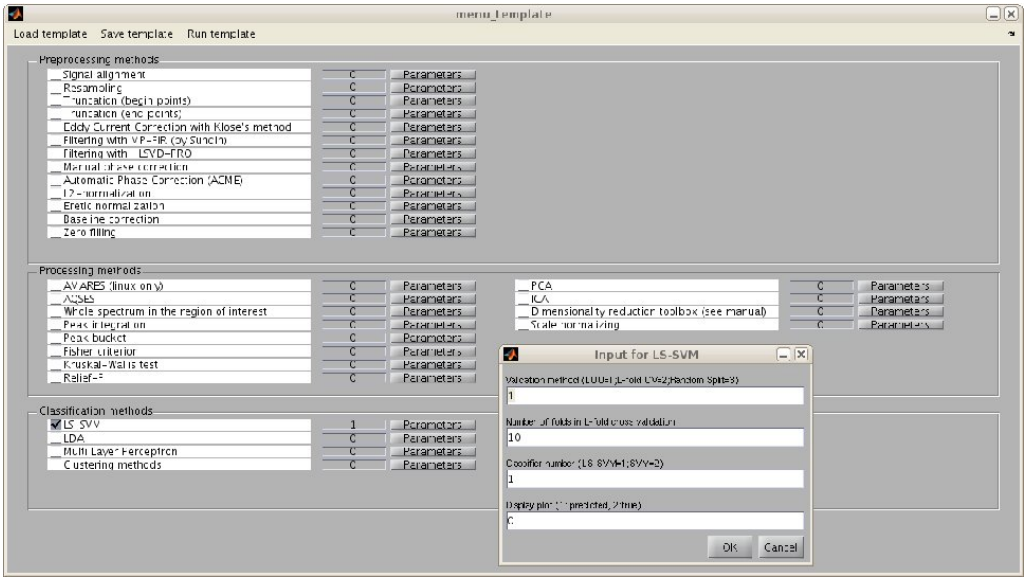


Figure 8: The template window in SPID-GUI.

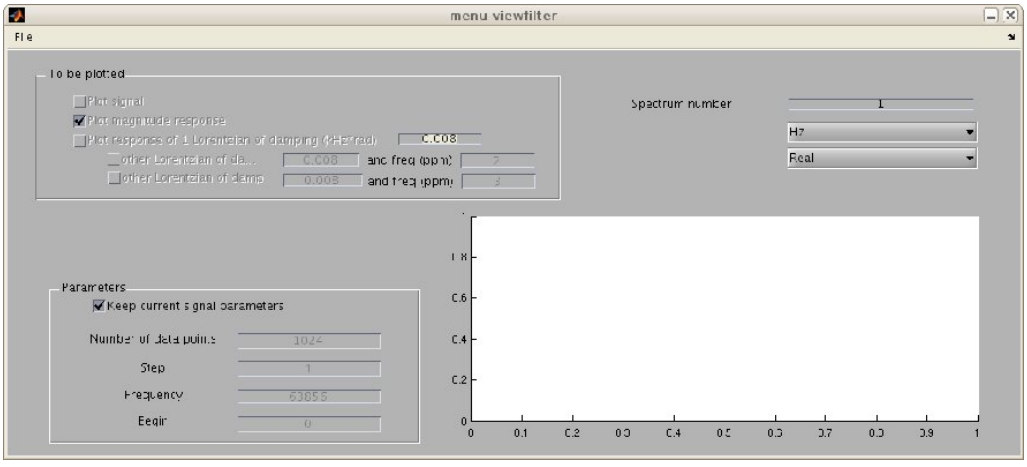


Figure 9: The window to visualize FIR filters.

the computed amplitude estimates. The window for normalizing basis sets is given in Fig. 6.2. We notice that the user can choose for each metabolite whether it is needed or not to filter the water components. In case of artifically made components (where the water components are absent), it is recommended not to use filtering methods since the noise level in the water region may be considered as zero, resulting in numerical problems. Interesting to note is that the results of database normalization can be visualized in the "View results (Quantification)" window (via Processing menu tab).

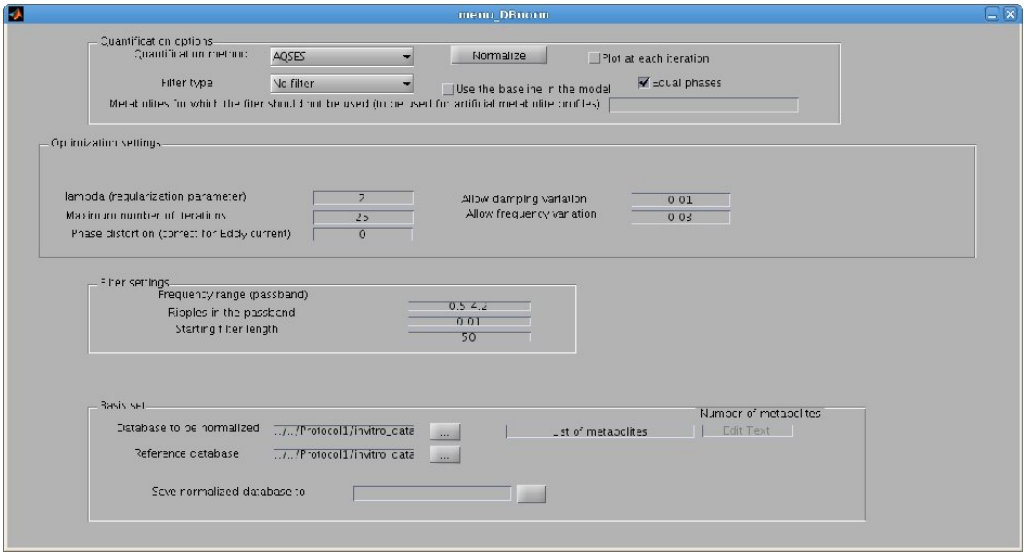


Figure 10: Database normalization window.

7 How can I create my own methods and plug them in TSPID?

One of the most important characteristic of SPID, and in particular TSPID, is that it is really easy to add new methods. Adding new methods for SPID-GUI has not been optimized yet and will not be discussed here. To add a new method in TSPID (this method will automatically be added in the template tool of SPID-GUI), only three things are needed: writing (a) m file(s) with the methods, edit `listmethods.txt` (in `simulate_signals/main`), update `interprate_template.m` (same directory) if necessary.

How to edit these files ?

Whatever the method there must be one m file which should have certain inputs and outputs. This .m file can call other .m files but will play as interface between these .m files and TSPID. Depending on the method type, the inputs and output will be different, as described in Table 2.

Note that this table can be easily modified by editing `interprate_template.m`.

Editing `listmethods.txt` is also mandatory for introducing the new method in TSPID. A part of `listmethods.txt` is given as illustration below. One can notice that the syntax is

Method type	Inputs	Outputs
Preprocessing	`signal`,`step`,`frequency`,`ndp`,`begin`,(`opt`)	`signal`,(`ndp`),(`step`)
Processing	`signal*`,`step`,`frequency`,`ndp`,`begin`, (`procres`),(`classopt`),(`opt`)	`scores`,`misc`
Classifying	`procres`,`classopt`,(`opt`)	`classres`

Table 2: Inputs and outputs required by `interprate_template.m`. The parameters in parentheses are only mandatory for some methods (see appendix for details). `opt` can be one or several **hyperparameters** (the concept of hyperparameters is defined in Section 5.2). The '*' symbol means that some methods requires the fourier transform of the signals instead of the time domain signal (see appendix for details).

similar to the syntax used to edit template files. For example, `opt` in Table 2 will be `centerfreq`, `targetfreq`, `doublet` for the "Spectrum alignment" method defined in `alignsignal.m`. Note that the comments are given on top of the file and are preceded by '#'.

```
# List of methods usable in templates with default parameters
# A template can be run from the command prompt of matlab: runTSPID...
# The template is a text file which can be automatically generated via menu_template.fig
# This file must be read line per line
# The first character of each line indicates the type of variables that follows in ...
# ex: '1 ; align ; Alignment'  => 1: preprocessing method, align: callback function;...
# ex: 33 ; centerfreq ; 1.15 ; "Current center frequency in ppm of the reference peak"
#   33 => it is a variable (not a method)
#   centerfreq = name of the variable
#   "Current center frequency in ppm of the reference peak" = description of the variable
1 ; alignsignal ; "Signal alignment" ;  # Spectrum alignment
33 ; centerfreq ; 1.15 ; "Current center frequency in ppm of the reference peak"
33 ; targetfreq ; 1.47 ; "Target frequency in ppm of the reference peak"
33 ; doublet ; 1 ; "Doublet if 1, singlet if 0"
1 ; resamptempl ; "Resampling" ;
33 ; P ; 0.2 ; "Original step (1/fs)"
33 ; Q ; 0.1664 ; "Final step(1/fs)"
1 ; truncbegin ; "Truncation (begin points)" ; # Truncation of the begin points
33 ; truncbegpts ; 1024 ; "Number of points to keep (? to end)"
1 ; truncend ; "Truncation (end points)" ; # Truncation of the end points
33 ; truncendpts ; 1024 ; "Number of points to keep (1 to ?)"
1 ; ECCKlose ; "Eddy Current Correction with Klose's method";
33 ; watersigfile; "{'watersignal1.mat' watersignal2.mat'}" ; ".mat files with the ..."
1 ; MPFIR ; "Filtering with MP-FIR (by Sundin)";
33 ; boundL ; 0.25 ; "Low bound of the passband (in ppm)"
33 ; boundH ; 4.2 ; "High bound of the passband (in ppm)"
33 ; rippass ; 0.01 ; "Passband ripples"
33 ; M ; 50 ; "Filter order"
```

8 Further improvements

Numerous improvements can be added to SPID (*i.e.*, TSPID and SPID-GUI). Here are some of them.

1. New methods
    - New tools for constructing simulated data: Heteregenoity between voxels is not taken into account yet (specially important for MRSI data), new tools for creating simulated HRMAS data and MRSI data, add tools in the GUI for making easier the creation of new databases.
    - Automatic tool for discarding data based on the papers by Kreis *et al.* [Kre04] or Van der Graaf *et al.* [vdGJSH+08].
    - New preprocessing tools: Aligning method based on a singlet or several singlets (for in vivo MR, Cho, Cr and NAA can be used as reference singlets), Eddy current correction methods (QUALITY [dG90], QUECC [BDMW00]), baseline correction methods for short echo time in vivo data ([YSM98],[ESL+05],[SYM01]).
    - Processing tools: AMARES (windows and linux version), QUEST [RCC+04], QUEST-AQSES fusion?
    - New classification tools: LS-SVM with Bayesian framework, add multiclass classification methods, C4.5, random forest
    - Implementing bootstrapping: this might be useful when only a few samples are available (difficult to learn a reliable model) or too many samples are available (high computation time). The combination of models is supposed to be more reliable than the corresponding separated models.
    - Implement Bagging (bootstrap aggregating). Especially useful if the learning algorithms are unstable as it is the case for SVM-based methods (like LS-SVM), decision tree methods or neural network methods (note that clustering methods like the K-nearest neighbors are known to be stable).
    - Implement boosting methods like Adaboost to boost weak learning algorithms (hardly better than a random prediction).
    - Tools for MRSI data ((pre)processing and classification + plotting tools)
    - Unsupervised methods: Kernel clustering
    - Data comparison: Kernel CCA (to compare in vivo and ex vivo HR-MAS for instance)
2. Programming
    - Re-program in a more object-oriented way: for example, `fids` could be a class with the attributes `signal`, `ndp`, `begin`, `step` and `frequency` and with features or methods the preprocessing methods and some processing methods. Another class would be the processing results with the features `scores` and `misc`.
    - Add some GUI tools to visualize the classification results
    - Generate the preprocessing, processing and classification menus automatically (except for the plotting tools like `View results (Quantitation)`).
    - Improve the code documentation (set up a code syntax)
3. Miscellaneous
    - General debugging, especially for the classification methods

A Preprocessing methods

A.1 Aligning spectra

```
%************************************************************************
%                                ALIGNSIGNAL.M
%************************************************************************
% PURPOSE:  Align signal with respect to a given reference peak (singlet or
% doublet)
%************************************************************************
% CALL:     [signal] = alignsignal(signal, step,frequency,ndp,begin,fref1,fref2,doublet)
%************************************************************************
% INPUT:    signal      -- signal                        row vector
%           step        -- time step between points (ms) scalar
%           frequency   -- spectrometer frequency (kHz)  scalar
%           ndp         -- number of data points         scalar
%           begin       -- begin time (ms)               scalar
%           fref1       -- current frequency             scalar
%                       of the reference peak
%           fref2       -- frequency where the reference scalar
%                       peak should be
%           doublet     -- 1 if ref. peak is a doublet   binary
% OUTPUT:   signal      -- aligned signal                row vector
%************************************************************************
```

Description
The algorithm detects a doublet (the singlet version is not yet implemented) within a small ppm range around `fref1`. The spectrum is shifted such that the center of the doublet is at `fref2` ppm. A warning message is displayed if no doublet has been found.

A.2 Resampling

```
%************************************************************************
%                                RESAMPLTEMPL.M
%************************************************************************
% PURPOSE:  Resample signals (based on the 'resample' matlab function)
%************************************************************************
% CALL:     [sig,ndp,step] = resamptempl(signal, step, frequency,ndp,begin,P,Q)
%************************************************************************
% INPUT:    signal      -- signal                        row vector
%           step        -- time step between points (ms) scalar
%           frequency   -- spectrometer frequency (kHz)  scalar
%           ndp         -- number of data points         scalar
%           begin       -- begin time (ms)               scalar
%           P           -- original step (ms)            scalar
%           Q           -- final step (ms)               scalar
% OUTPUT:   sig         -- resampled signal              row vector
%           ndp         -- number of points              scalar
```

```
%           step        -- final step                    scalar
%***********************************************************************
```

Description
Resample time domain signals of step `P`. Remind that if the sample frequency (or the step) changes, the spectrometer frequency (`frequency`) remains unmodified.

A.3 Truncation of the begin or end points

```
%***********************************************************************
%                                TRUNCBEGIN.M
%***********************************************************************
% PURPOSE:  Truncate the begin points of the signals
%***********************************************************************
% CALL:     [signal,ndp] = truncbegin(signal, step, frequency,ndp,begin,tr)
%***********************************************************************
% INPUT:    signal      -- signal                        matrix
%           step        -- time step between points (ms) scalar
%           frequency   -- spectrometer frequency (kHz)  scalar
%           ndp         -- number of data points         scalar
%           begin       -- begin time (ms)               scalar
%           tr          -- number of points to keep      scalar
% OUTPUT:   signal      -- truncated signal              matrix
%           ndp         -- number of data points         scalar
%***********************************************************************
```

Description
`truncbegin` (respectively `truncend`) truncates the begin (respectively end) points of the time domain signals to keep only `tr` points.

A.4 Eddy current correction with ECC by Klose

```
%***********************************************************************
%                                ECCKLOSE.M
%***********************************************************************
% PURPOSE:  Correct for eddy currents using Klose's method
%***********************************************************************
% CALL:     [signal,ndp] = ECCKlose(signal,step,frequency,ndp,begin,watersigfile)
%***********************************************************************
% INPUT:    signal      -- signal                        matrix
%           step        -- time step between points (ms) scalar
%           frequency   -- spectrometer frequency (kHz)  scalar
%           ndp         -- number of data points         scalar
%           begin       -- begin time (ms)               scalar
%           watersigfile-- mat files of water signals    cell of strings
% OUTPUT:   signal      -- ECC corrected signal          matrix
%           ndp         -- number of data points         scalar
%***********************************************************************
```

Description
Correct for eddy currents using Klose's method [Klo90]. The water signals can be in one or several mat file(s) but only one variable per mat file can represent the signals.
Example: Suppose `signal` is a matrix 20x1024 (20 signals of 1024 points each), `watersigfile` can be one mat file with a variable which contains all the water signals (matrix 20x1024), or can be 20 files and each file contains 1 water signal (vector 1x1024). In the latter case, the name of the variable which represents the water signal can differ from one file to the other. This variable is recognized thanks to the variable ndp which is present in all water mat files. In brief, any water mat file must contain the variables `watersignal` (or any other name), `ndp` and `begin` (the number of columns in `watersignal` must be `ndp`).

A.5 Lineshape correction with QUALITY method

```
%************************************************************************
%                                QUALITY.M
%************************************************************************
% PURPOSE:  QUALITY method based on de Graaf et al.'s paper
%************************************************************************
% CALL:     [signal,ndp] = QUALITY(signal,step,frequency,ndp,begin,watersigfile,dispplot)
%************************************************************************
% INPUT:    signal      -- signal                        matrix
%           step        -- time step between points (ms) scalar
%           frequency   -- spectrometer frequency (kHz)  scalar
%           ndp         -- number of data points         scalar
%           begin       -- begin time (ms)               scalar
%           watersigfile-- mat files of water signals    cell of strings
% OUTPUT:   signal      -- ECC corrected signal          matrix
%           ndp         -- number of data points         scalar
%************************************************************************
```

Description
Lineshape correction with the QUALITY method. This method has been implemented from Bartha et al.'s paper (Magnetic Resonance in Medicine 44:641-645,2000) and could be different from the original QUALITY method invented by de Graaf et al. in (Magnetic Resonance in Medicine 13:343-357,1990)

A.6 Water filtering with MPFIR

```
%************************************************************************
%                                MPFIR.M
%************************************************************************
% PURPOSE:  Filter out a specific region of the spectrum using the
% maximum-phase FIR filter by T. Sundin
%************************************************************************
% CALL:     [signal] = MPFIR(signal,step,frequency,ndp,begin,boundL,boundH,rippass,M)
%************************************************************************
% INPUT:    signal      -- signal                        row vector
%           step        -- time step between points (ms) scalar
```

```
%           frequency   -- spectrometer frequency (kHz)  scalar
%           ndp         -- number of data points         scalar
%           begin       -- begin time (ms)               scalar
%           boundL      -- low bound                     scalar
%           boundH      -- high bound                    scalar
%           rippass     -- passband ripples              scalar
%           M           -- filter order (pair)           scalar
% OUTPUT:   signal      -- filtered signal                row vector
%*************************************************************************
```

Description
Pass band filter in [boundL,boundH] based on the maximum-phase FIR filter by Sundin *et al.* [SVVH+99]. The current implementation only allows to filter one region (not a multi-passband). Experience shows that if the constraints on the ripples (rippass) are too strict, the algorithm might fail to find good filter coefficients. The number of coefficients (filter order+1) should be kept under 91 to avoid numerical problems.

A.7 Water filtering with MPFIR0

```
%*************************************************************************
%                                 MPFIR0.M
%*************************************************************************
% PURPOSE:  Filter out a specific region of the spectrum using the
% maximum-phase FIR filter by JB Poullet
%*************************************************************************
% CALL:     [signal] = MPFIR(signal,step,frequency,ndp,begin,boundL,boundH,M,TBW,Dfact,dis))
%*************************************************************************
% INPUT:    signal      -- signal                        row vector
%           step        -- time step between points (ms) scalar
%           frequency   -- spectrometer frequency (kHz)  scalar
%           ndp         -- number of data points         scalar
%           begin       -- begin time (ms)               scalar
%           boundL      -- low bound                     scalar
%           boundH      -- high bound                    scalar
%           M           -- filter order (pair)           scalar
%           TBW         -- transition band width         scalar
%           Dfact       -- delay factor                  scalar
%           dis         -- displaying figures            binary
% OUTPUT:   signal      -- filtered signal               row vector
%*************************************************************************
```

Description
Filter out a specific region of the spectrum using the maximum-phase FIR filter by Poullet *et al.* [PPH08]. The default values for TBW, Dfact should be kept unchanged if the user has no knowledge. We refer to [PPH08] for more information.

A.8 Water filtering with HLSVD-PRO

```
%************************************************************************
%                                HLSVDPROtempl.M
%************************************************************************
% PURPOSE:  Filter out a specific region of the spectrum using the
% HLSVD-PRO by T. Laudadio
%************************************************************************
% CALL:     signal = HLSVDPROtempl(signal,step,frequency,ndp,begin,boundL,boundH,M)
%************************************************************************
% INPUT:    signal      -- signal                          row vector
%           step        -- time step between points (ms) scalar
%           frequency   -- spectrometer frequency (kHz)  scalar
%           ndp         -- number of data points         scalar
%           begin       -- begin time (ms)               scalar
%           boundL      -- low bound                     scalar
%           boundH      -- high bound                    scalar
%           M           -- model order                   scalar
% OUTPUT:   signal      -- filtered signal               row vector
%************************************************************************
```

Description
State-space based method proposed by Laudadio *et al.* [LMV+02] to filter out components in a specific frequency region. The signal is modeled as a sum of lorentzians. The lorentzians located outside [`boundL`,`Boundh`] suppressed are. Common values for `M` for *in vivo* MRS signals are between 25 and 30. Highest values can be used for high resolution magic angle spinning data (HR-MAS), although it seems that values around 30 provide already good suppression of the water peak (at least for 1D PRESAT).

A.9 Manual phase correction

```
%************************************************************************
%                                PHASEMANUAL.M
%************************************************************************
% PURPOSE:  Phase manually (zero and first order correction are given by
% the user)
%************************************************************************
% CALL:     signal = phasemanual(signal,step,frequency,ndp,begin,phc0,phc1)
%************************************************************************
% INPUT:    signal      -- signal                          row vector
%           step        -- time step between points (ms) scalar
%           frequency   -- spectrometer frequency (kHz)  scalar
%           ndp         -- number of data points         scalar
%           begin       -- begin time (ms)               scalar
%           phc0        -- zero order phase correction   scalar
%           phc1        -- first order phase correction  scalar
% OUTPUT:   signal      -- phased signal                 row vector
%************************************************************************
```

Description
Manual phasing in the sense that the zero and first order phase correction values are given by the user.

A.10 Automatic phase correction using ACME

```
%************************************************************************
%                                autophasACME.M
%************************************************************************
% PURPOSE:  Automatic phase correction using ACME method
%************************************************************************
% CALL:     signal = autophasACME(signal,step,frequency,ndp,begin,phc0,phc1)
%************************************************************************
% INPUT:    signal      -- signal                         row vector
%           step        -- time step between points (ms) scalar
%           frequency   -- spectrometer frequency (kHz)  scalar
%           ndp         -- number of data points         scalar
%           begin       -- begin time (ms)               scalar
%           phc0        -- zero order initialization(deg)scalar
%           phc1        -- first order initialization(deg)scalar
% OUTPUT:   signal      -- autophased signal             row vector
%************************************************************************
```

Description
Automatic phase correction using ACME method [CWGG02] (based on entropy minimization). This method has not been fully studied and could yield bad phasing.

A.11 L2-normalization

```
%************************************************************************
%                                L2norm.M
%************************************************************************
% PURPOSE:  L2-Normalization
%************************************************************************
% CALL:     signal = L2norm(signal,step,frequency,ndp,begin,boundL,boundH)
%************************************************************************
% INPUT:    signal      -- signal                         row vector
%           step        -- time step between points (ms) scalar
%           frequency   -- spectrometer frequency (kHz)  scalar
%           ndp         -- number of data points         scalar
%           begin       -- begin time (ms)               scalar
%           boundL      -- low bound                     scalar
%           boundH      -- high bound                    scalar
% OUTPUT:   signal      -- normalized signal             row vector
%************************************************************************
```

Description
Divide the signal by its L2-norm in the frequency region [`boundL`, `boundH`].

A.12 Eretic normalization

```
%************************************************************************
%                               normEretic.M
%************************************************************************
% PURPOSE:  Eretic-Normalization
%************************************************************************
% CALL:     signal = normEretic(signal,step,frequency,ndp,begin,boundL,boundH,nbr)
%************************************************************************
% INPUT:    signal      -- signal                        row vector
%           step        -- time step between points (ms) scalar
%           frequency   -- spectrometer frequency (kHz)  scalar
%           ndp         -- number of data points         scalar
%           begin       -- begin time (ms)               scalar
%           boundL      -- low bound                     scalar
%           boundH      -- high bound                    scalar
%           nbr         -- reference signal number       scalar
% OUTPUT:   signal      -- normalized signal             row vector
%************************************************************************
```

Description
Normalize all the time domain signals (rows of `signal`) with respect to the the amplitude of a reference peak lying in the frequency interval [`boundL`, `boundH`]. One signal is chosen as reference (`nbr` is the signal number in the signal matrix) and the reference peak of this signal is reconstructed by HLSVD-PRO [LMV+02]. This reference peak is then used for normalizing the other signals.
STEPS:
1. HLSVD-PRO applied to the reference signal (model order = 30) → the reference peak is reconstructed
2. the reference peak is used as database for quantifying the other
signals in the frequency region [`boundL`, `boundH`]
3. the signals are normalized with respect to the results of quantitation with AQSES [PSS+07]

A.13 Baseline correction by apodization

```
%************************************************************************
%                               bascorrApod.M
%************************************************************************
% PURPOSE:  Baseline correction with an apodization function
%************************************************************************
% CALL:     signal =  bascorrApod(signal,step,frequency,ndp,begin,factor)
%************************************************************************
% INPUT:    signal      -- signal                        row vector
%           step        -- time step between points (ms) scalar
%           frequency   -- spectrometer frequency (kHz)  scalar
%           ndp         -- number of data points         scalar
%           begin       -- begin time (ms)               scalar
%           factor      -- apodization coefficient       scalar
% OUTPUT:   signal      -- baseline corrected signal     row vector
%************************************************************************
```

Description
The baseline is modeled by the signal multiplied by an apodization function. This baseline is then substracted from time domain signal:

```
signaltemp = signal.*exp(-factor*t);
signal = signal-signaltemp;
```

A.14 Baseline offset correction

```
%************************************************************************
%                                basOffCor.M
%************************************************************************
% PURPOSE:  Suppress the baseline offset (but do not suppress the baseline
% due to macromolecules)
%************************************************************************
% CALL:     signal = basOffCor(signal,step,frequency,ndp,begin,boundL,boundH,M)
%************************************************************************
% INPUT:    signal      -- signal                         row vector
%           step        -- time step between points (ms) scalar
%           frequency   -- spectrometer frequency (kHz)  scalar
%           ndp         -- number of data points         scalar
%           begin       -- begin time (ms)               scalar
%           boundL      -- low bound                     scalar
%           boundH      -- high bound                    scalar
% OUTPUT:   signal      -- corrected signal               row vector
%************************************************************************
```

Description
Suppress the baseline offset but do not suppress the baseline due to macromolecules. The mean of the spectrum in the frequency region of NO interest is subtracted from the spectrum. This region should only contain noise! No peaks!

A.15 Correction for baseline distortions

```
%************************************************************************
%                                basCorrGol.M
%************************************************************************
% PURPOSE:  Baseline correction with a method based on Golotvin et al.'s paper
%************************************************************************
% CALL:     signal =  basCorrGol(signal,step,frequency,ndp,begin,M,N,n,nn,dispplot)
%************************************************************************
% INPUT:    signal      -- signal                         row vector
%           step        -- time step between points (ms) scalar
%           frequency   -- spectrometer frequency (kHz)  scalar
%           ndp         -- number of data points         scalar
%           begin       -- begin time (ms)               scalar
%           M           -- filter order (smoothing)      scalar
%           N           -- number of spectral points in the
```

```
%                       rectangle (baseline detection)   scalar
%           n           -- noise factor                  scalar
%           nn          -- noise definition
%                         (number of points)             scalar
%           dispplot    -- display of denoised and       binary
%                       original signals
% OUTPUT:   signal      -- denoised signal             row vector
%*************************************************************************
```

Description
Correction for baseline distortions with a method based on Golotvin *et al.*'s paper (Journal of Magnetic Resonance 146, 122125 (2000)). This method should only be used on baseline distortions (rolling baseline), not on macromolecular baseline. For a distinction between macromolecular baseline and baseline distortions, we refer to the review paper by Poullet et *et al.* [PSH08].

A.16 SNR improvement

```
%*************************************************************************
%                                SNRapodization.M
%*************************************************************************
% PURPOSE:  SNR improvement using an apodization function
%*************************************************************************
% CALL:     signal =  SNRapodization(signal,step,frequency,ndp,begin,factor)
%*************************************************************************
% INPUT:    signal      -- signal                          row vector
%           step        -- time step between points (ms) scalar
%           frequency   -- spectrometer frequency (kHz)  scalar
%           ndp         -- number of data points         scalar
%           begin       -- begin time (ms)               scalar
%           factor      -- apodization coefficient       scalar
% OUTPUT:   signal      -- SNR improved signal           row vector
%*************************************************************************
```

Description
SNR improvement using an apodization function. The signal is simply multiplied by an apodization function, which is here a simple exponential function.

A.17 Denoising using wavelets

```
%*************************************************************************
%                               denoisWav.M
%*************************************************************************
% PURPOSE:  Denoising using wavelets
%*************************************************************************
% CALL:     signal =  denoisWav(signal,step,frequency,ndp,begin,wname,level,adl,thr,sorh,keepapp,d
%*************************************************************************
% INPUT:    signal      -- signal                          row vector
%           step        -- time step between points (ms) scalar
```

```
%           frequency   -- spectrometer frequency (kHz)  scalar
%           ndp         -- number of data points         scalar
%           begin       -- begin time (ms)               scalar
%           wname       -- wavelet name                  cell of 1 string
%           level       -- Level at which the wavelet    scalar
%                       decomposition is performed
%           adl         -- automatic def. of defaults    binary
%           thr         -- Global positive threshold     scalar
%           sorh        -- 1:hard, 0:soft thresholding   binary
%           keepapp     -- 1:approximation coefficients  binary
%                       cannot be thresholded, or 0:not
%           dispplot    -- display of denoised and       binary
%                       original signals
% OUTPUT:   signal      -- denoised signal             row vector
%***************************************************************************
```

Description
Denoising using wavelets.

A.18 Denoising using moving average filter

```
%***************************************************************************
%                                denoisMAF.M
%***************************************************************************
% PURPOSE:  Denoising using moving average filter
%***************************************************************************
% CALL:     signal =  denoisMAF(signal,step,frequency,ndp,begin,,dispplot)
%***************************************************************************
% INPUT:    signal      -- signal                          row vector
%           step        -- time step between points (ms) scalar
%           frequency   -- spectrometer frequency (kHz)  scalar
%           ndp         -- number of data points         scalar
%           begin       -- begin time (ms)               scalar
%           M           -- filter order                  scalar
%           dispplot    -- display of denoised and       binary
%                       original signals
% OUTPUT:   signal      -- denoised signal             row vector
%***************************************************************************
```

Description
Denoising using moving average filter. This method is not usually used for MRS and should be used for tests only.

A.19 zerofill

```
%***************************************************************************
%                                zerofill.M
%***************************************************************************
```

```
% PURPOSE:  Zero filling
%*********************************************************************
% CALL:     [signal,ndp] = zerofill(signal,step,frequency,ndp,begin,nbr)
%*********************************************************************
% INPUT:    signal      -- signal                          row vector
%           step        -- time step between points (ms) scalar
%           frequency   -- spectrometer frequency (kHz)  scalar
%           ndp         -- number of data points         scalar
%           begin       -- begin time (ms)               scalar
%           nbr         -- number of zeros to add        scalar
% OUTPUT:   signal      -- baseline corrected signal     row vector
%*********************************************************************
```

Description
Add `nbr` zeros at the end of the signal.

B Processing methods

B.1 Quantification with HLSVD-PRO

```
%*********************************************************************
%                                HLSVDPROquant.M
%*********************************************************************
% PURPOSE:  Filter out a specific region of the spectrum using the
% HLSVD-PRO by T. Laudadio et al.
%*********************************************************************
% CALL:     signal = HLSVDPROquant(signal,step,frequency,ndp,begin,boundL,boundH,M)
%*********************************************************************
% INPUT:    signal      -- signal                          row vector
%           step        -- time step between points (ms) scalar
%           frequency   -- spectrometer frequency (kHz)  scalar
%           ndp         -- number of data points         scalar
%           begin       -- begin time (ms)               scalar
%           boundL      -- low bound                     scalar
%           boundH      -- high bound                    scalar
%           M           -- model order                   scalar
% OUTPUT:   scores      -- filtered signal               row vector
%           misc        -- see bottom of the file        structure
%*********************************************************************
```

Description
Quantification with HLSVD-PRO. More information about the method can be found in [PSVHVH07, LMV+02].

B.2 Quantification with AMARES

```
%*********************************************************************
```

```
%                                AMARESTempl.M
%***************************************************************************
% PURPOSE:  Quantitation using AMARES
%***************************************************************************
% CALL:     [scores,misc] = AMARESTempl(signal,step,frequency,ndp,begin,MPFIR)
%***************************************************************************
% INPUT:    signal      -- signal                         row vector
%           step        -- time step between points (ms) scalar
%           frequency   -- spectrometer frequency (kHz)  scalar
%           ndp         -- number of data points         scalar
%           begin       -- begin time (ms)               scalar
%           MPFIR       -- MPFIR =1, nothing =0          binary
% OUTPUT:   newscores   -- new scores (best variables)   matrix
%           misc        -- nothing                       structure
%***************************************************************************
```

Description
THIS FUNCTION NEEDS TO BE UPDATED. Calcul of the metabolite amplitude estimates based on the quantitation method AMARES [VvdBVH97]. AMARES can be used with the maximum-phase FIR filter proposed by Sundin *et al.* [SVVH$^+$99] (`MPFIR`=1) or not (`MPFIR`=0). The prior knowledge is given in a separate .par file called `input.par`.

B.3 Quantification using AQSES and based on the subtract method from QUEST for the baseline modeling

```
%***************************************************************************
%                                SUBTRACTTEMPL.M
%***************************************************************************
% PURPOSE:  subtract method using AQSES
%***************************************************************************
% CALL:     [scores,misc] = subtract(signal,step,frequency,ndp,begin,trunc,bascorr,dbase,abs_scl_f
%    lineshape, phasedistort,filtertype,boundL,boundH,ripple,filterlength,...
%    prior_equal,equal_to,allow_damp,allow_freq,equalph,baseline_boolean,lambda,...
%    maxiter,plotting,wname,level,adl,thr,sorh,keepapp)
%***************************************************************************
% INPUT:    signal      -- signal                         row vector
%           step        -- time step between points (ms) scalar
%           frequency   -- spectrometer frequency (kHz)  scalar
%           ndp         -- number of data points         scalar
%           begin       -- begin time (ms)               scalar
%           trunc       -- number of truncated points to scalar
%                          estimate the baseline
%           bascorr     -- baseline fitting method       scalar
%                          1:HLSVD-PRO, 2:wavelet (denoising)
%           iter        -- 1:iterations,0: no iter.      binary
%           dbase       -- database name (met. profiles) string cell
%           abs_scl_factor-- scaling factor              scalar
%           lineshape   -- 1=Lorentzian,2=Gaussian,      scalar
%                           3 = Voigt
```

```
%           phasedistort-- phase distortion              binary
%           filtertype  -- 'fir' for MP-FIR              string
%                             (Sundin et al.)
%                          'hlsvd' for HLSVD-PRO
%                             (Laudadio et al.)
%           boundL      -- low bound                     scalar
%           boundH      -- high bound                    scalar
%           ripple      -- ripples (FIR filter)          scalar
%           filterlength-- nbr of filter coefficients  scalar
%           prior_equal --                               vector
%           equal_to    -- vectors of indeces specifying vector
%                       simple prior knowledge, i.e., dampings
%                       and frequences of metabolites in prior_equal
%                       are equal to equal_to (default [], [])
%           allow_damp  -- allowed damping variations    scalar
%           allow_freq  -- allowed frequency variations  scalar
%           equalph     -- 1 = equal phase               binary
%                          0 = non equal phase
%           baseline_boolean-- 1 = baseline in the model binary
%                              0 = no basline in the model
%           lambda      -- regularization parameter      scalar
%           maxiter     -- maximum nbr of iterations     scalar
%           plotting    -- display (=1) or not (=0)      binary
%           wname       -- wavelet name                  cell of 1 string
%           level       -- Level at which the wavelet    scalar
%                       decomposition is performed
%           adl         -- automatic def. of defaults    binary
%           thr         -- Global positive threshold     scalar
%           sorh        -- 1:hard, 0:soft thresholding   binary
%           keepapp     -- 1:approximation coefficients  binary
%                       cannot be thresholded, or 0:not
% OUTPUT:   scores      -- new scores (best variables)   vector
%           misc        -- parameters (ampl., damp., etc)structure
%                       modeled baseline, residual, etc.
%************************************************************************
```

Description
Subtract method using AQSES to quantify MRS data. Information about the subtract method can be found in [RSC+05]. This method is employed for baseline modeling. Information about AQSES can be found in [PSS+07]. More details about AQSES and its parameters can be found in [SVH04], [SVH05a] and [SVH05b].

B.4 Quantification with AQSES

```
%************************************************************************
%                                AQSES.M
%************************************************************************
% PURPOSE:  Quantitation using AQSES
%************************************************************************
```

```
% CALL:     [scores,misc] = AQSES(signal,step,frequency,ndp,begin,dbase,abs_scl_factor,...
%    lineshape, phasedistort,filtertype,boundL,boundH,ripple,filterlength,...
%    prior_equal,equal_to,allow_damp,allow_freq,equalph,baseline_boolean,lambda,...
%    maxiter,plotting)
%************************************************************************
% INPUT:    signal      -- signal                         row vector
%           step        -- time step between points (ms) scalar
%           frequency   -- spectrometer frequency (kHz)  scalar
%           ndp         -- number of data points         scalar
%           begin       -- begin time (ms)               scalar
%           dbase       -- database name (met. profiles) string cell
%           abs_scl_factor-- scaling factor              scalar
%           lineshape   -- 1=Lorentzian,2=Gaussian,      scalar
%                           3 = Voigt
%           phasedistort-- phase distortion              binary
%           filtertype  -- 'fir' for MP-FIR              string
%                             (Sundin et al.)
%                          'hlsvd' for HLSVD-PRO
%                             (Laudadio et al.)
%           boundL      -- low bound                     scalar
%           boundH      -- high bound                    scalar
%           ripple      -- ripples (FIR filter)          scalar
%           filterlength-- nbr of filter coefficients    scalar
%           prior_equal --                               vector
%           equal_to    -- vectors of indeces specifying vector
%                       simple prior knowledge, i.e., dampings
%                       and frequences of metabolites in prior_equal
%                       are equal to equal_to (default [], [])
%           allow_damp  -- allowed damping variations    scalar
%           allow_freq  -- allowed frequency variations  scalar
%           equalph     -- 1 = equal phase               binary
%                          0 = non equal phase
%           baseline_boolean-- 1 = baseline in the model binary
%                              0 = no basline in the model
%           lambda      -- regularization parameter      scalar
%           maxiter     -- maximum nbr of iterations     scalar
%           plotting    -- display (=1) or not (=0)      binary
% OUTPUT:   scores      -- new scores (best variables)   vector
%           misc        -- parameters (ampl., damp., etc)structure
%                       modeled baseline, residual, etc.
%************************************************************************
```

Description
Calcul of the metabolite amplitude estimates based on the quantitation method AQSES [PSS+07]. More details on the method can be found in [SVH04], [SVH05a] and [SVH05b].

B.5 Feature extraction using the whole spectrum

```
%************************************************************************
```

```
%                                WHOLESPECT.M
%************************************************************************
% PURPOSE:  Feature extraction by taking the whole spectrum in the region
% of interest
%************************************************************************
% CALL:     [scores,misc] = wholespect(signal,step,frequency,ndp,begin,boundL,boundH)
%************************************************************************
% INPUT:    signal      -- signal                          row vector
%           step        -- time step between points (ms) scalar
%           frequency   -- spectrometer frequency (kHz)  scalar
%           ndp         -- number of data points         scalar
%           begin       -- begin time (ms)               scalar
%           boundL      -- low bound                     scalar
%           boundH      -- high bound                    scalar
% OUTPUT:   scores      -- amplitude values of the points matrix
%           misc        -- see bottom of the file        structure
%************************************************************************
```

Description
Feature extraction using the whole spectrum in the region of interest.

B.6 Quantification with peak integration

```
%************************************************************************
%                               peakintegtempl.M
%************************************************************************
% PURPOSE:  Feature extraction with peak integration
%************************************************************************
% CALL:     [scores,misc] = peakintegtempl(signal,step,frequency,ndp,begin,...
% ir,freqmatrix,compNames)
%************************************************************************
% INPUT:    signal      -- signal                          row vector
%           step        -- time step between points (ms) scalar
%           frequency   -- spectrometer frequency (kHz)  scalar
%           ndp         -- number of data points         scalar
%           begin       -- begin time (ms)               scalar
%           ir          -- interval types                scalar (integer)
%           freqmatrix  -- interval matrix               matrix
%           compNames   -- Names of the intervals        cell
% OUTPUT:   scores      -- peak integrated values        matrix
%           misc        -- see bottom of the file        structure
%************************************************************************
```

Description
The integrals of the absolute spectrum in the given frequency regions are calculated. `scores` is a $N \times M$ matrix where N is the number of spectra and M is the number of intervals. Five different types of intervals are at user's disposal.

ir=0 User's intervals. The user must give the intervals in a $M \times 2$ matrix simply concatening

the intervals one after the other (one interval per row). Example: intervals [2.09 2.17], [2.21 2.23], [2.39 2.50] will be encoded by [2.09 2.17; 2.21 2.23; 2.39 2.50]. The component names associated to the corresponding intervals must also be given as a cell of strings. The number of rows of `freqmatrix` must match the length of `compNames`.

ir=1 Intervals corresponding to short echo time MR spectra at 1.5 T: Not corrected yet.

ir=2 Intervals corresponding to long echo time MR spectra at 1.5 T: Not corrected yet.

ir=3 Intervals corresponding to HRMAS spectra at 11 T: [1.30 1.34], [1.45 1.49], [1.84 1.94], [1.99 2.025], [2.09 2.17], [2.21 2.23], [2.39 2.50], [3.01 3.03], [3.19 3.205], [3.205 3.23], [3.235 3.245], [3.255 3.275], [3.33 3.36],[3.39 3.435], [3.50 3.58], [3.58 3.70], [3.71 3.82], [3.91 3.945], [4.03 4.075], [4.08 4.14].

ir=4 Intervals corresponding to HRMAS spectra at 14.1 T: idem as at 11 T.

The intervals should be disjoint to avoid information redundancy. The intervals are defined in `processing/readintervals.m`. An error message is displayed if `ir`> 4.

B.7 Quantification with peak bucketting

```
%************************************************************************
%                                peakbuckettempl.M
%************************************************************************
% PURPOSE:  Feature extraction with peak bucketing
%************************************************************************
% CALL:     [scores,misc] = peakbuckettempl(signal,step,frequency,ndp,begin,...
% ir,freqmatrix,compNames)
%************************************************************************
% INPUT:    signal      -- signal                          row vector
%           step        -- time step between points (ms) scalar
%           frequency   -- spectrometer frequency (kHz)  scalar
%           ndp         -- number of data points         scalar
%           begin       -- begin time (ms)               scalar
%           ir          -- interval types                scalar (integer)
%           freqmatrix  -- interval matrix               matrix
%           compNames   -- Names of the intervals        cell
% OUTPUT:   scores      -- peak integrated values        matrix
%           misc        -- see bottom of the file        structure
%************************************************************************
```

Description
All the points of the spectrum in given frequency intervals are extracted as features (`scores`). The intervals are defined in the same way as in `peakintegtempl` (see `processing/readintervals.m`).

B.8 Feature extraction with fish criterion

```
%************************************************************************
%                                fishcrit.M
```

```
%************************************************************************
% PURPOSE:  Feature selection using the Fisher criterion
%************************************************************************
% CALL:     [newscores,misc] = fishcrit(signal,procres,classopt,nbrvar)
%************************************************************************
% INPUT:    signal      -- signal                          row vector
%           step        -- time step between points (ms) scalar
%           frequency   -- spectrometer frequency (kHz)  scalar
%           ndp         -- number of data points         scalar
%           begin       -- begin time (ms)               scalar
%           procres     -- structure of processing results
%           classopt    -- structure of classification options
%           nbrvar      -- number of variables to keep   integer
%           boundL      -- low bound                     scalar
%           boundH      -- high bound                    scalar
% OUTPUT:   newscores   -- new scores (best variables)   matrix
%           misc(i).ranked_vector -- ranked vector       vector
%************************************************************************
```

Description
The main features are selected using the Fisher criterion. The number of features to keep is given by `nbrvar`. This method can only select features of already extracted features. In other words, `procres.scores` cannot be empty. The selected features are saved in a 3D array: data in the first dimension, features in the second dimension and classification problem number in the third dimension. This method is only usable with binary classification methods and `classopt.classtype` must be defined.

```
Example:
procres.scores = [1 2 3 4;1.1 2.1 3.1 4.1; 1.2 2.2 3.2 4.2];
classopt.classtype = {'GBM','MET','MEN','GBM'};
nbrvar = 2;
... selection of the best features for each pair of classes...
newscores(:,:,1) = [1 4;1.1 4.1;1.2 4.2]; %classes GBM vs MET
newscores(:,:,2) = [2 4;2.1 4.1;2.2 4.2]; %classes GBM vs MEN
newscores(:,:,3) = [2 4;2.1 4.1;2.2 4.2]; %classes MET vs MEN
```

B.9 Feature extraction with kruskal-wallis method

```
%************************************************************************
%                                kruswal.M
%************************************************************************
% PURPOSE:  Feature selection using the Kruskal-Wallis test
%************************************************************************
% CALL:     [newscores,misc] = kruswal(signal,procres,classopt,nbrvar)
%************************************************************************
% INPUT:    signal      -- signal                          row vector
%           step        -- time step between points (ms) scalar
%           frequency   -- spectrometer frequency (kHz)  scalar
%           ndp         -- number of data points         scalar
```

```
%           begin       -- begin time (ms)              scalar
%           procres     -- structure of processing results
%           classopt    -- structure of classification options
%           nbrvar      -- number of variables to keep   integer
%           boundL      -- low bound                     scalar
%           boundH      -- high bound                    scalar
% OUTPUT:   newscores   -- new scores (best variables)   matrix
%           misc(i).ranked_vector -- ranked vector       vector
%*************************************************************************
```

Description
Feature selection using the Kruskal-Wallis test. For information about the parameters, see the description of `fishcrit.m` above.

B.10 Feature extraction with Relief-F

```
%*************************************************************************
%                                relieff.M
%*************************************************************************
% PURPOSE:  Feature selection using  Relief-f
%*************************************************************************
% CALL:     [newscores,misc] = relieff(signal,procres,classopt,nbrvar,iterations,neighbors)
%*************************************************************************
% INPUT:    signal      -- signal                        row vector
%           step        -- time step between points (ms) scalar
%           frequency   -- spectrometer frequency (kHz)  scalar
%           ndp         -- number of data points         scalar
%           begin       -- begin time (ms)               scalar
%           procres     -- structure of processing results
%           classopt    -- structure of classification options
%           nbrvar      -- number of variables to keep   integer
%           iterations  -- number of iterations          scalar
%           neighbors   -- number of neighbors           scalar
%           boundL      -- low bound                     scalar
%           boundH      -- high bound                    scalar
% OUTPUT:   newscores   -- new scores (best variables)   matrix
%           misc(i).ranked_vector -- ranked vector       vector
%*************************************************************************
```

Description
Feature selection using Relief-F. For information about the parameters, see the description of `fishcrit.m` above.

B.11 Feature extraction with PCA

```
%*************************************************************************
%                               PCAtemp.M
%*************************************************************************
```

```
% PURPOSE:  Feature selection using PCA
%*************************************************************************
% CALL:     [newscores,misc] = PCAtempl(signal,procres,classopt,nbrvar)
%*************************************************************************
% INPUT:    signal      -- signal                          row vector
%           step        -- time step between points (ms) scalar
%           frequency   -- spectrometer frequency (kHz)  scalar
%           ndp         -- number of data points         scalar
%           begin       -- begin time (ms)               scalar
%           procres     -- structure of processing results
%           classopt    -- structure of classification options
%           nbrvar      -- number of variables to keep   integer
%           boundL      -- low bound                     scalar
%           boundH      -- high bound                    scalar
% OUTPUT:   newscores   -- new scores (best variables)   matrix
%           misc        -- see bottom of file            structure
%*************************************************************************
```

Description
Feature selection using principal component analysis (PCA). For information about the parameters, see the description of `fishcrit.m` above.

B.12 Feature extraction with ICA

```
%*************************************************************************
%                                ICAtemp.M
%*************************************************************************
% PURPOSE:  Feature selection using ICA
%*************************************************************************
% CALL:     [newscores,misc] = ICAtempl(signal,procres,classopt,nbrvar)
%*************************************************************************
% INPUT:    signal      -- signal                          row vector
%           step        -- time step between points (ms) scalar
%           frequency   -- spectrometer frequency (kHz)  scalar
%           ndp         -- number of data points         scalar
%           begin       -- begin time (ms)               scalar
%           procres     -- structure of processing results
%           classopt    -- structure of classification options
%           nbrvar      -- number of variables to keep   integer
%           boundL      -- low bound                     scalar
%           boundH      -- high bound                    scalar
% OUTPUT:   newscores   -- new scores (best variables)   matrix
%           misc        -- see bottom of file            structure
%*************************************************************************
```

Description
Feature selection using fast-ICA (http://www.cis.hut.fi/projects/ica/fastica/), fast-independent component analysis. For information about the parameters, see the description of `fishcrit.m` above.

B.13 Dimensionality reduction using the toolbox by Vandermaaten

```
%************************************************************************
%                              dimreductempl.M
%************************************************************************
% PURPOSE:  Feature selection using dimensionality reduction techniques
%************************************************************************
% CALL:     [newscores,misc] = dimreductempl(signal,step,frequency,ndp,begin,...
procres, classopt,methname,nbrvar,boundL,boundH)
%************************************************************************
% INPUT:    signal      -- signal                        row vector
%           step        -- time step between points (ms) scalar
%           frequency   -- spectrometer frequency (kHz)  scalar
%           ndp         -- number of data points         scalar
%           begin       -- begin time (ms)               scalar
%           procres     -- structure of processing results
%           classopt    -- structure of classification options
%           methname    -- method name                   string
%           nbrvar      -- number of variables to keep   integer
%           boundL      -- low bound                     scalar
%           boundH      -- high bound                    scalar
% OUTPUT:   newscores   -- new scores (best variables)   matrix
%           misc        -- see bottom of file            structure
%************************************************************************
```

Description
Feature selection using dimensionality reduction techniques (Matlab Toolbox for Dimensionality Reduction v0.1b, http://www.cs.unimaas.nl/l.vandermaaten). For information about the parameters, see the description of `fishcrit.m` above. In addition to classical methods like 'PCA', 'LDA' or 'ICA', this toolbox proposes the following methods 'MDS', 'Isomap', 'LandmarkIsomap', 'LLE', 'Laplacian', 'HessianLLE', 'LTSA', 'DiffusionMaps', 'KernelPCA', 'GDA', 'SNE', 'SPE', 'AutoEncoder', and 'AutoEncoderEA'. For more information on the techniques, we refer to the paper "Dimensionality Reduction: A Comparative Review" by L.J.P. van der Maaten, E.O. Postma, and H.J. van den Herik. The paper is available from http://www.cs.unimaas.nl/l.vandermaaten. Please refer this paper or the website if you use the dimension toolbox in SPID.

B.14 Scale normalization

```
%************************************************************************
%                               scalenorm.M
%************************************************************************
% PURPOSE:  Normalize variables
%************************************************************************
% CALL:     [scores,misc] = scalenorm(signal,step,frequency,ndp,begin,ir,freqmatrix,compNames)
%************************************************************************
% INPUT:    signal      -- signal                        row vector
%           step        -- time step between points (ms) scalar
%           frequency   -- spectrometer frequency (kHz)  scalar
%           ndp         -- number of data points         scalar
```

```
%           begin       -- begin time (ms)              scalar
%           procres     -- structure of processing results
%           classopt    -- structure of classification options
%           scale       -- scaling option               integer
%           sigmoid     -- =1 : sigmoid shape           binary
% OUTPUT:   scores      -- spectrum point values        matrix
%           misc        -- see bottom of file           structure
%************************************************************************
```

Description
This method normalize the columns or rows of a matrix, or both:

```
%  scale             =  0: do nothing,
%                       1: columns have mean 0, std 1
%                       2: rows have mean, std 1
%                       3: try to do do both 1 & 2
```

If sigmoid = 1, `x` is replaced by `tanh(x*sigmoid)`.

C Classification methods

C.1 Least-Squares Support Vector Machines (LSSVM)

```
%************************************************************************
%                                LSSVMtempl.M
%************************************************************************
% PURPOSE:  Multiclass Binary Classification with LSSVM
%************************************************************************
% CALL:    classres = LSSVMtempl(procres,classopt,validtype,L,classifiernbr,dispplot)
%************************************************************************
% INPUT:    procres     -- processing results        structure
%               scores     -- data (MxN with M=nbr of
%                             data and N=nbr of variables) matrix
%           classopt    -- classification options        structure
%               classtype  -- class type                 vector/cell
%           validtype   -- validation type               scalar
%           L           -- number of folds in CV         scalar
%           classifiernbr-- LSSVM=1, SVM=2               scalar
%           dispplot    -- plot=1, no plot=0             binary
% OUTPUT:   classres    -- classification results        structure
%                misclassifications-- nbr of             vector
%                    misclassifications for the
%                    different pair combinations
%                gamma      -- hyperparam in LSSVM        scalar
%                sigma      -- hyperparam in LSSVM        scalar
%                predictionmatrix -- pred. matrix         matrix
%                auc        -- area under ROC curve       vector
%************************************************************************
```

Description
Multiclass Binary Classification with LS-SVM (can only handle binary classification problems). This method uses the LS-SVM toolbox available at http://www.esat.kuleuven.ac.be/sista/lssvmlab/. Please refer to the LS-SVM toolbox website if you use LS-SVM in SPID. Three validation methods are available: leave-one-out, L-fold cross validation and stratified random sampling. Gamma and sigma are tuned by `tunelssvm.m`. The bayesian version is based on the paper by Lu *et al.* [LDS+07]. It is also possible to weight the features using the automatic relevance determination (ARD) method.

C.2 Kernel Logistic Regression (KLR)

```
%************************************************************************
%                                KLRtempl.M
%************************************************************************
% PURPOSE:  Multiclass Binary Classification with Kernel Logistic
% Regression (KLR)
%************************************************************************
% CALL:    classres = KLRtempl(procres,classopt,validtype,L,classifiernbr,dispplot)
%************************************************************************
% INPUT:    procres     -- processing results  structure
%               scores     -- data (MxN with M=nbr of
%                             data and N=nbr of variables) matrix
%           classopt    -- classification options        structure
%               classtype  -- class type                 vector/cell
%           validtype   -- validation type               scalar
%           L           -- number of folds in CV         scalar
%           classifiernbr-- LSSVM=1, SVM=2               scalar
%           dispplot  -- plot=1, no plot=0             binary
% OUTPUT:   classres    -- classification results        structure
%                misclassifications-- nbr of            vector
%                    misclassifications for the
%                    different pair combinations
%                gamma     -- hyperparam in KLR        scalar
%                sigma     -- hyperparam in KLR        scalar
%                predictionmatrix -- pred. matrix        matrix
%                auc       -- area under ROC curve       vector
%************************************************************************
```

Description
Multiclass Binary Classification with Kernel Logistic Regression (can only handle binary classification problems).

C.3 Linear Discriminant Analysis (LDA)

```
%************************************************************************
%                                LDAtempl.M
%************************************************************************
% PURPOSE:  Classification using LDA
```

```
%************************************************************************
% CALL:    [classres] = LDAtempl(procres,classopt,validtype,L,dispplot)
%************************************************************************
% INPUT:    procres     -- structure of processing results
%           classopt    -- classification options        structure
%               classtype  -- class type                 vector/cell
%           procres     -- processing options            structure
%               scores     -- data (MxN with M=nbr of
%                             data and N=nbr of variables) matrix
%           validtype   -- validation type               scalar
%           L           -- number of folds in CV         scalar
%           dispplot    -- display options               binary
% OUTPUT:   classres    -- classification results        structure
%                misoptimal-- minimum nbr of             vector
%                    misclassifications for the
%                    different pair combinations
%                predictionmatrix -- pred. matrix        matrix
%************************************************************************
```

Description
Multiclass Binary Classification with LDA (can only handle binary classification problems).

C.4 Multi-Layer Perceptron (MLP)

```
%************************************************************************
%                                MLPtempl.M
%************************************************************************
% PURPOSE: Multiclass Binary Classification with Multi-Layer Perceptron (MLP)
%************************************************************************
% CALL:    classres =  MLPtempl(procres,classopt,validtype,L,classifiernbr,...
nhidden,alpha,ncycles,actfct,optim)
%************************************************************************
% INPUT:    procres     -- processing results        structure
%               scores     -- data (MxN with M=nbr of
%                             data and N=nbr of variables) matrix
%           classopt    -- classification options        structure
%               classtype  -- class type                 vector/cell
%           validtype   -- validation type               scalar
%           L           -- number of folds in CV         scalar
%           classifiernbr-- MLP=1                        scalar
%           nhidden     -- number of hidden neurons      integer
%           alpha       -- weigth decay                  scalar
%           ncycles     -- number of training cycles     integer
%           actfct      -- activation function           string
%           optim       -- optimization algorithm        string
% OUTPUT:   classres    -- classification results        structure
%                misclassification-- minimum nbr of      vector
%                    misclassifications for the
%                    different pair combinations
```

```
%                predictionmatrix -- pred. matrix        matrix
%************************************************************************
```

Description
Multiclass Binary Classification with Multi-Layer Perceptron (MLP; it can only handle binary classification problems). MLP is used via de neural network software "Netlab" available at http://www.ncrg.aston.ac.uk/netlab/down.php. Possible activation functions are `'linear'`,`'logistic'` or `'softmax'` and possible optimization algorithms are `'quasinew'`, `'conjgrad'`,`'scg'`, `'graddesc'` (quasi-Newton, conjugate gradients, scaled conjugate gradients, and gradient descent). More information can be found at http://www.ncrg.aston.ac.uk/netlab/.

C.5 clustertempl

```
%************************************************************************
%                               clustertempl.M
%************************************************************************
% PURPOSE:  Multiclass Binary Classification with clustering methods
%************************************************************************
% CALL:    classres = clustertempl(procres,classopt,validtype,L,classifiernbr,...
% neighbors,genclust,dispplot)
%************************************************************************
% INPUT:    procres     -- processing results            structure
%               scores     -- data (MxN with M=nbr of
%                             data and N=nbr of variables) matrix
%           classopt    -- classification options        structure
%               classtype  -- class type                 vector/cell
%           validtype   -- validation type               scalar
%           L           -- number of folds in CV         scalar
%           classifiernbr-- KNN=1, Kmeans=2, Fuzzy c-means scalar
%           neighbors   -- number of neighbors           integer
%           genclust    -- <2 (2 centers), >=2 (nbr classes) scalar
%           dispplot    -- plot = 1, no plot = 0         binary
% OUTPUT:   classres    -- classification results        structure
%                misclassification -- nbr of             vector
%                    misclassifications for the
%                    different pair combinations
%                predictionmatrix -- pred. matrix        matrix
%************************************************************************
```

Description
Multiclass Binary Classification with clustering methods (unsupervised methods). Available clustering methods are `'KNN'`, `'Kmeans'` and `'Fuzzy c-means'`.

References

[BDMW00] R. Bartha, D. J. Drost, R. S. Menon, and P. C. Williamson. Spectroscopic lineshape correction by QUECC: combined QUALITY deconvolution and eddy current correction. *Magn. Reson. Med.*, 44(4):641–5, 2000.

[CWGG02] Li Chen, Zhiqiang Weng, LaiYoong Goh, and Marc Garland. An efficient algorithm for automatic phase correction of NMR spectra based on entropy minimization. *J. Magn. Reson.*, 158:164–68, 2002.

[dG90] A. A. de Graaf. QUALITY: quantification improvement by converting lineshapes to the Lorentzian type. *Magn. Reson. Med.*, 13:343–57, 1990.

[DNLV+07] B. De Neuter, J. Luts, L. Vanhamme, P. Lemmerling, and S. Van Huffel. Java-based framework for processing and displaying short-echo-time magnetic resonance spectroscopy signals. *Computer Methods and Programs in Biomedicine*, 85:129–137, 2007.

[ESL+05] C. Elster, F. Schubert, A. Link, M. Walzel, F. Seifert, and H. Rinneberg. Quantitative magnetic resonance spectroscopy: Semi-parametric modeling and determination of uncertainties. *Magn. Reson. Med.*, 53:1288–96, 2005.

[jMR] Magnetic resonance user interface in java (jmrui).

[Klo90] U. Klose. In vivo proton spectroscopy in presence of eddy currents. *Magn. Reson. Med.*, 14:26–30, 1990.

[Kre04] R. Kreis. Issues of spectral quality in clinical ^{1}H-magnetic resonance spectroscopy and a gallery of artifacts. *NMR Biomed.*, 17(6):361–381, Oct 2004.

[LDS+07] Chuan Lu, Andy Devos, Johan A K Suykens, Carles Arús, and Sabine Van Huffel. Bagging linear sparse bayesian learning models for variable selection in cancer diagnosis. *IEEE Trans. Inf. Technol. Biomed.*, 11(3):338–347, May 2007.

[LMV+02] T. Laudadio, N. Mastronardi, L. Vanhamme, P. Van Hecke, and S. Van Huffel. Improved Lanczos algorithms for blackbox MRS data quantitation. *J. Magn. Reson.*, 157(2):292–7, 2002.

[PPH08] Jean-Baptiste Poullet, Rik Pintelon, and Sabine Van Huffel. A new fir filter technique for solvent suppression in mrs signals. *J Magn Reson*, Oct 2008.

[PSH08] Jean-Baptiste Poullet, Diana M Sima, and Sabine Van Huffel. Mrs signal quantitation: A review of time- and frequency-domain methods. *J Magn Reson*, Sep 2008.

[PSS+07] J. Poullet, D. M. Sima, A. W. Simonetti, B. De Neuter, L. Vanhamme, P. Lemmerling, and S. Van Huffel. An automated quantitation of short echo time MRS spectra in an open source software environment: AQSES. *NMR Biomed.*, 20(5):493–504, 2007.

[PSVHVH07] J. Poullet, Diana M Sima, S. Van Huffel, and P. Van Hecke. Frequency-selective quantitation of short-echo time ^{1}H magnetic resonance spectra. *J. Magn. Reson.*, 186(2):293–304, Jun 2007.

[RCC+04] H. Ratiney, Y. Coenradie, S. Cavassila, D. van Ormondt, and D. Graveron-Demilly. Time-domain quantitation of ^{1}H short echo-time signals: background accommodation. *MAGMA*, 16(6):284–96, 2004.

[RSC+05] H. Ratiney, M. Sdika, Y. Coenradie, S. Cavassila, D. van Ormondt, and D. Graveron-Demilly. Time-domain semi-parametric estimation based on a metabolite basis set. *NMR Biomed.*, 18(1):1–13, 2005.

[SVH04] D.M. Sima and S. Van Huffel. A note on template splines. Technical Report 04-228, K.U.Leuven (Leuven, Belgium), 2004.

[SVH05a] D. M. Sima and S. Van Huffel. AQSES$_{VP}$ - description of a variable projection implementation for nonlinear least squares with linear bounds constraints, applied to accurate quantification of short-echo time magnetic resonance spectroscopic signals. Technical report, Department of electrical engineering (ESAT), Katholiek Universiteit Leuven, Leuven, Belgium, 2005.

[SVH05b] D. M. Sima and S. Van Huffel. Description of a variable projection implementation for nonlinear least squares with linear bounds constraints, applied to accurate quantification of short-echo time magnetic resonance spectroscopic signals. Technical report, 2005.

[SVH06] D. M. Sima and S. Van Huffel. Separable nonlinear least squares fitting with linear bound constraints and its application in magnetic resonance spectroscopy data quantification. *J. Comput. Appl. Math.*, 203(1):264–278, 2006.

[SVVH$^+$99] Tomas Sundin, Leentje Vanhamme, Paul Van Hecke, Ioannis Dologlou, and Sabine Van Huffel. Accurate quantification of ^{1}H spectra: From finite impulse response filter design for solvent suppression to parameter estimation. *J. Magn. Reson.*, 139(2):189–204, 1999.

[SYM01] B. J. Soher, K. Young, and A. A. Maudsley. Representation of strong baseline contributions in ^{1}H MR spectra. *Magn. Reson. Med.*, 45(6):966–72, 2001.

[vdGJSH$^+$08] Marinette. van der Graaf, Margarida Julià-Sapé, Franklyn A. Howe, Anne Ziegler, Carles Majós, Angel Moreno-Torres, Mark Rijpkema, Dionisio Acosta, Kirstie S. Opstad, Yvonne M van der Meulen, Carles Arús, and Arend Heerschap. MRS quality assessment in a multicentre study on MRS-based classification of brain tumours. *NMR Biomed.*, 21(2):148–158, Feb 2008.

[VvdBVH97] L. Vanhamme, A. van den Boogaart, and S. Van Huffel. Improved method for accurate and efficient quantification of MRS data with use of prior knowledge. *J. Magn. Reson.*, 129:35–43, 1997.

[YSM98] K. Young, B. J. Soher, and A. A. Maudsley. Automated spectral analysis II: application of wavelet shrinkage for characterization of non-parameterized signals. *Magn. Reson. Med.*, 40(6):816–21, 1998.