# famn_opt v0.3.2a user manual

Henri Colaux

7th June 2019

**Abstract**

FAM-N (Fast Amplitude Modulation - N pulses) is a composite pulse that exploits the phenomenon of inversion/saturation of satellite transtion into the central transition to enhance the efficiency of the conversion pulse in MQMAS experiments, improving the overall sensitivity of the experiment. This program determines the number of individual pulse and their respective durations required for an optimal conversion under the experimental conditions defined by the user.

## Contents

## 1 Installation

To be executed, make sure that you have all of the following programs installed:

**SIMPSON** `famn_opt` requires SIMPSON to be installed in order to function. SIMPSON can be obtained from `http://nmr.au.dk/software/simpson/`.

The installation is successful if typing ”simpson” in the command prompt returns the version number, i.e.,

```
SIMPSON version 4.1.1, Copyright (C)
1999-2000 Mads Bak and Jimmy T. Rasmussen
2001 Mads Bak and Thomas Vosegaard
2002-2007 Thomas Vosegaard
2008-2009 Zdenek Tosner, Thomas Vosegaard, and Niels Chr. Nielsen
2009 plus Rasmus Andersen
2010-2014 Zdenek Tosner, Niels Chr. Nielsen, Thomas Vosegaard
SIMPSON comes with ABSOLUTELY NO WARRANTY, for details
read the COPYING file included in this distribution.
This is free software, and you are welcome to redistribute
it under certain conditions according to the GNU General Public License.

Please specify an inputfile, optionally with other arguments.
```

**Python** `famn_opt` was developed in python 3.7.3. I recommend installing Anaconda, which provides by default all packages required for this program and for scientific production in general:

- `https://www.anaconda.com/distribution/#download-section`

You can also install the conventionnal python distribution for :

- `https://www.python.org/downloads/`

**famn_opt** The package can be obtained from different sources:

- From PyPI: Installation from online repository. If you use Anaconda, locate ”Anaconda prompt” or ”Anaconda powershell” and type
  `pip install famn_opt`

else, open a prompt window and type
`python pip install famn_opt` or `python3 pip install famn_opt`

---

- From executable (in progress):

---

- From source: Get the latest version from Github:
  `https://github.com/hcolaux/famn_opt`
  Uncompress the folder. If you use Anaconda, open "Anaconda prompt" or "Anaconda powershell", or a terminal/command prompt window if not. Browse to the root of the uncompressed directory that contains the file "setup.py" and type, if you use Anaconda,
  `setup.py sdist bdist_wheel`
  to assemble the code, then
  `pip install .`
  to install.
  If you use the a termimal/command prompt Window, type
  `python setup.py sdist bdist_wheel`
  then
  `python pip install .`
  The installation is successful if typing `import famn_opt` in the python interpreter does not return the error:
  `ModuleNotFoundError: No module named 'famn_opt'`

## 2 Usage in Spyder

Anaconda comes with Spyder, a python editor with built-in iPython interpreter and code execution. Here is a quick guide for FAM-N optimisation once `famn_opt` is successfully installed.

- Open Spyder
- Type `import famn_opt` in the iPython console.
- Type `famn_opt.inputfile()` in the iPython console to get and save an input script for FAMN optimisation. By default, the file name gives the class instance name, *e.g.* `filename` as used below.
- Open `filename.py` with spyder.
- Fill-in the input parameters.
- Execute the script with Spyder (press F5 or the play button) and wait for completion.
- If you have filled `filename.param['export_filepath']` with a non-valid or an empty path, execute
  `filename.export_all_select()`
  to select the output folder. For individual exports, see section 4.

## 3 Usage with the Python interpreter

This sections describes a more general usage than with Spyder that uses the default python interpreter.

**Import package** The package must be imported from the python interpreter with

`import famn_opt`

**Get input script** The execution in console mode is easier from the input script, from which the simulation parameters can be easily modified. This script defines a new class instance, highlights the parameters relevant for the end user then carry out the optimisation. Typing the command

`famn_opt.inputfile()`

opens a dialog file that allows you to save a copy of this input script. As before, the instance name is by default given by the file name, *e.g.* `filename` as used below. A detailed description of all of the parameters relevant for the end user are given in this file. For a detailed description of all parameters, see the file `famn_opt/opt.py`

**Code execution** Internally, the code execution is carried out by three methods:

- `filename.initialise()`: Organises the input and returns error codes.
- `filename.FAMN_loop()`: Runs the actual simulation.
- `filename.finalise()`: Organises the simulation results for the output.

All three methods are sequentially executed in the input script.

## 4 Output results

Once the execution is successful, output files can be exported with the following commands. All methods presented open a file dialog when executed.

**General outputs:**

- `filename.save()` Save a serialised copy of the class instance as a file (by default `.pkl`), so it can be subsequently reloaded if required.
- `filename.load()` Loads the class instance from a file saved with the `filename.save()` method.
- `filename.text_report()` Save a `.txt` file containing the characteristics of the FAM-N pulse.
- `filename.step_figure(nos = 1)` Plot and save a figure showing the build-up in detect coherence against time for the FAM-N pulse obtained with the optimisation for the step specified by `nos`.
- `filename.all_steps_figure()` Plot and save a figure containing all steps of the optimisation of a multiple-step FAM-N optimisation if applicable. For a single-step optimisation, produces the same figure as `filename.step_figure(nos=1)` for all steps.

**Pulse programs:**

- Pulse lists describing FAM-N and the durations of their components, to be copied and pasted in any pulse program: `filename.topspin_pulselist()` for output in the Bruker Topspin format and `filename .delta_pulselist()` for output in the Jeol Delta.
- Multiple-Quantum Filtered (MQF) pulse program: `filename.topspin_mqf()` for Bruker Topspin and `filename.delta_mqf()` for Jeol Delta.
- Shifted-echo split-$t_1$ Multiple-Quantum Magic-Angle-Spinning (MQMAS): `filename.topspin_mqmas()` for Bruker Topspin and `filename.delta_mqmas()` for Jeol Delta.
- Shifted-echo split-$t_1$ Multiple-Quantum Carr-Purcell-Meiboom-Gill Magic-Angle-Spinning (MQ-CPMG-MAS): `filename.topspin_mqcpmgmas()` for Bruker Topspin.

**All outputs:**

- `filename.export_all()` Export all of the above.

---

Alternative methods that do not open a dialogue box are also presented in brackets. These take two parameters: `outputfilename` (File name for the output, with or without extension) and `outputfilepath` (File path were the file is exported). Those are, in order,

- `filename.savestate(outputfilename, outputfilepath = '')`
- `filename.loadstate(outputfilename = '__previous_session__.pkl', outputfilepath = ''):`
- `filename.export_text_report(outputfilename, outputfilepath)`
- `filename.FAMNsteps[nos-1].figures['opt'].savefig(outputfullpath)` (NB: `nos` is the step number, 1 for a single-step FAM-N)
- `filename.plot_allsteps_fig()` then `filename.figures['allsteps_fig'].savefig(outputfullpath )`

**Pulse programs:**

- `filename.export_topspin_pulselist(outputfilepath, outputfilepath = '', first = 1))` (NB: `first` = index in Topspin of the first FAM-N pulse) and `filename.export_delta_pulselist(outputfilepath , outputfilepath)`
- `filename.export_topspin_MQF(outputfilepath, outputfilepath)` and `filename.export_delta_MQF (outputfilepath, outputfilepath)`
- `filename.export_topspin_MQMAS(outputfilepath, outputfilepath)` and `filename.export_delta_MQMAS (outputfilepath, outputfilepath)`
- `filename.export_topspin_MQCPMGMAS(outputfilepath, outputfilepath)`

**All outputs:**

- `filename.export_all_files(exportfilepath = None)` and `filename.param['export_filepath']`

---

**Screen outputs:**

- `filename.screen_report()`: Display a summary of the current FAM-N optimisation in the python interpreter.
- `filename.plot_allsteps_fig()`: Display a figure representing the magnetisation build-up of the final detected element as a function of time and the optimised FAM-N pulses including all steps of it.

## 5 Bibliography

**FAM-N:**

- Colaux, H., Dawson, D. M., Ashbrook, S. E., Efficient Amplitude-Modulated Pulses for Triple-to Single-Quantum Coherence Conversion in MQMAS NMR, The Journal of Physical Chemistry A, *2014*, 118, 31
- Colaux, H., Dawson, D. M., Ashbrook, S. E., Investigating FAM-N pulses for signal enhancement in MQMAS NMR of quadrupolar nuclei, Solid State Nuclear Magnetic Resonance, *2017*, 84, 89
- Kanwal N., Colaux, H., Dawson, D. M., Nishiyama Y., Ashbrook, S. E., Sensitivity improvement in 5QMAS NMR experiments using FAM-N pulses Solid State Nuclear Magnetic Resonance, *2019*, 100, 1

**SIMPSON:**

- Bak, M.; Rasmussen, J.; Nielsen, N. *et al.*, SIMPSON: a general simulation program for solid-state NMR spectroscopy Journal of Magnetic Resonance, *2000*, 147, 296