diffusion-fundamentals

The Open-Access Journal for the Basic Principles of Diffusion Theory, Experiment and Application

DAMARIS – A flexible and open software platform for NMR spectrometer control

Achim Gädke[1], *Markus Rosenstihl*[1], *Christopher Schmitt*[2], *Holger Stork*[1], *Nikolaus Nestle*[1,3]

[1] Institute of Condensed Matter Physics, TU Darmstadt
[2] Informatics, University of Applied Sciences, Frankfurt
[3] BASF AG Ludwigshafen, GKP/P, G 201

Corresponding author: Achim Gädke, Institute of Condensed Matter Physics, TU Darmstadt, Hochschulstraße 6, D-64289 Darmstadt, E-Mail: Achim.Gaedke@physik.tu-darmstadt.de

(received 7 June 2007, accepted 10 July 2007)

Abstract

Home-built NMR spectrometers with self-written control software have a long tradition in porous media research. Advantages of such spectrometers are not just lower costs but also more flexibility in developing new experiments (while commercial NMR systems are typically optimized for standard applications such as spectroscopy, imaging or quality control applications). Increasing complexity of computer operating systems, higher expectations with respect to user-friendliness and graphical user interfaces as well as increasing complexity of the NMR experiments themselves have made spectrometer control software development a more complex task than it used to be some years ago. Like that, it becomes more and more complicated for an individual lab to maintain and develop an infrastructure of purely home-built NMR systems and software. Possible ways out are:

- commercial NMR hardware with full-blown proprietary software or
- semistandardized home-built equipment and common open-source software environment for spectrometer control.

Our present activities in Darmstadt aim at providing a nucleus for the second option: DArmstadt MAgnetic Resonance Instrument Software (DAMARIS) [1]. Based on an ordinary PC, pulse control cards and ADC cards, we have developed an NMR spectrometer control platform that comes at a price tag of about 8000 Euro.

The present functionalities of DAMARIS are mainly focused on TD-NMR: the software was successfully used in single-sided NMR [2], pulsed and static field gradient NMR diffusometry [3]. Further work with respect to multipulse/multitriggering experiments in the time domain [4] and solid state NMR spectroscopy multipulse experiments are under development.

Keywords

NMR instrumentation, spectrometer control software, open source

1. Introduction

Reviewing a long tradition of home-built spectrometers [5,6,7,8,9] at universities and research institutes, one can see that these developments were driven by

- the need for special features, mobility or flexibility,
- the detailed understanding of the measurement procedure,
- complex pulse sequences and precise definition of data processing,
- adaption to already existing devices,
- the lack of money.

Nevertheless, such equipment is often used with great success at the very forefront of research in NMR. However, often home-built software lacks proper documentation and has no long term support. Frequently, it is designed and built up just for a single spectrometer.

In the Darmstadt case, a lab with nine rather different spectrometers had evolved with time. They were run with three different software platforms, each of them bound to specific hardware.

In attempts to keep up with novel developments, each of these spectrometer concepts revealed design limits or was found not to work properly even within its original specifications. Based on these experiences, a more general solution was sought, enabling experimentalists to use the same experimental control environment at different spectrometers and sharing the effort to improve the common parts of software.

The outcome of these endeavors is the DAMARIS concept – consisting of hardware and software components – which are described and presented on exemplary measurements in this contribution.

2. Project Outline

DAMARIS spectrometers are based on usual personal computers which are equipped with a pulse pattern generator and an analog digital converter (ADC). The computer runs a common operating system (Windows XP or Linux) which provides access to the NMR specific hardware via the vendors' drivers. The drivers are controlled by a so called “back end” program which organizes the components' interactions to a running MR spectrometer.

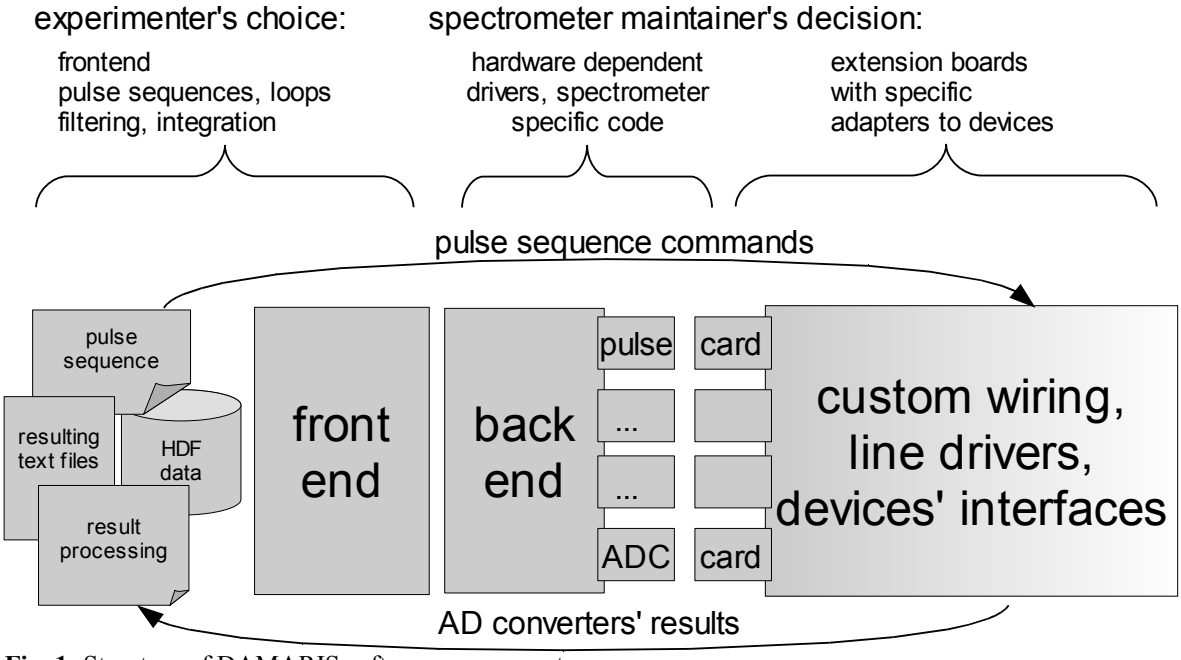

Fig. 1: Structure of DAMARIS software components

To address the spectrometer's power components (e.g. RF amplifiers, gradient shape controls, pulse formers) a customized digital controller unit (often called the “line driver”, see figure 2) is used. It connects the pulse pattern generator with all other devices, and achieves a timing control with a precision of several nanoseconds during a single MR signal scan.

This “back end” is fed with the actual MR experiment sequences by the “front end”, which is independent of the specific hardware components, therefore the same “front end” can be used on all DAMARIS machines. Information is exchanged via file system, files written subsequently by the “front end” represent “jobs” (e.g. single shots) for the “back end”. Correspondingly data measured by the “back end” is stored in “result” files. So it is easily possible to replay experiments or analyze results on single shot level. Furthermore, this interface only requires basic programming skills to write one's own “frontend”. Modern operating systems handle file access via cache, so the performance is satisfying.

This modularized approach enables the lab staff to fit in old and new devices into the DAMARIS environment. The instrumental modularization rules the software design of the “back end”. Dedicated driver components (classes in terms of object oriented software written in C++) are provided, which are lumped together in the “back end” program. Furthermore the “back end” translates the pulse sequence into specific control signals for the individual hardware components.

Dividing the spectrometer control software into a “back end” and a “front end” part one can realize both dedicated NMR programs for well-defined use cases (“single button applications”) and more advanced options based on general scripting facilities for ongoing methods development. Presently two types of “front ends” are available: a LabView [10] based “front end” and a “front end” solely dependent on free software, which will be described in this article.

This “front end” based on Python [11] and GTK [12] is used for method development in our lab. It provides two scripts which control the experimental procedure and the data processing separately. To monitor the spectrometer while the measurement is conducted, plots of the recorded datasets can be displayed and saved online. These scripts are based on the Python scripting language, so they offer all features of a full programming language. Especially the data processing script benefits from numerical extensions like scipy and numpy [13,14,15].

This “front end” provides an unified access to research lab's MR spectrometer capabilities:

- Pulse sequences can be modified in all parameters,
- Instrumental constants (e.g. preamplifiers' dead time and gating) are accessible,
- Raw data treatment can be controlled in detail by the “front end”,
- Storage of all relevant parameters and processing results is possible in the portable data format HDF5 [16].

Furthermore the expertise on the control software and specific hardware components is collected online [1]. The source code of DAMARIS is freely available, so other labs can join this project. By founding a widespread community sharing code and documentation, mutual benefits will be achieved.

3. Representative Spectrometer Setup

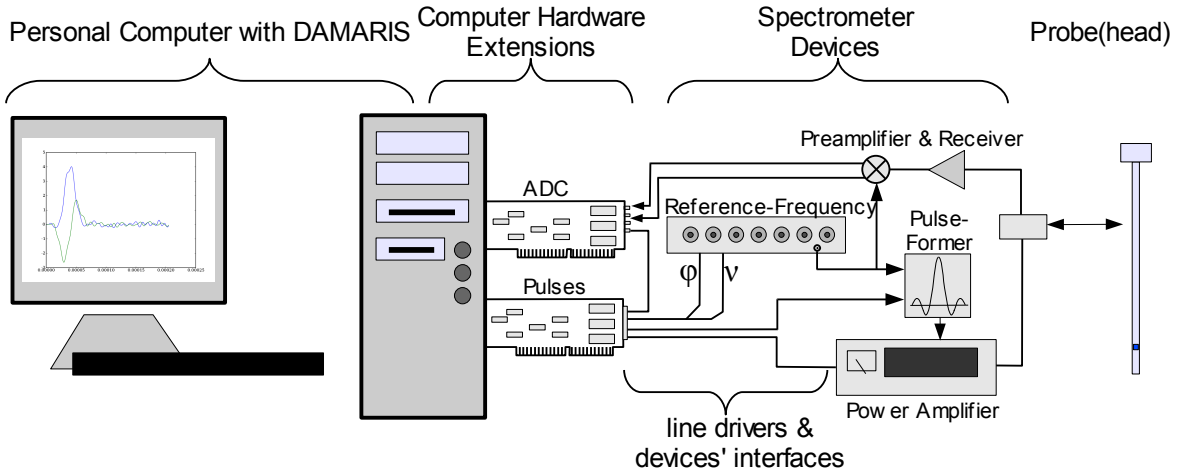

Fig. 2: Exemplary DAMARIS hardware structure

The present standard setup for DAMARIS works with a PulseBlaster 24Bit (SpinCore, Gainsville, FL) pulse programming board and a MI4021 (Spectrum GmbH, Grosshansdorf, Germany) ADC card. To address all remote controlled frequency and phase registers of PTS (Littleton, MA) frequency synthesizers, we have developed a demultiplexing device with 12 TTL lines – in contrast to 48 PTS input lines. All other devices are driven by direct connections to the pulse card. For example such a spectrometer is in operation at a static field gradient spectrometer and provides a home-built positioning stage to control the sample position relative to the magnetic field gradient.

Besides those standard components several other boards and extensions are supported:

- ADC cards from Datel (now C&D Technolgies, Blue Bell, PA), TiePie (Sneek, Netherlands) and Spectrum (Grosshansdorf, Germany)
- Pulse pattern generators and DDS frequency generator boards from SpinCore, (Gainsville, FL)
- Reference frequency generator from PTS (Littleton, MA) with phase and frequency control
- TecMag (Houston, TX) 20Bit DAC for pulsed field gradient control
- Eurotherm (Leesburg, VA) 2000 Series temperature control

All required software components are free and open source (except from some low level drivers bundled together with hardware). For Debian Linux a ready-made package is provided, which makes installation very easy, for most other recent Linux systems the base packages are contained in the standard distribution. In this case, the actual DAMARIS software has to be installed from source files. All packages are also available for Microsoft Windows, but unfortunately each must be installed manually. Detailed installation instructions are given on the homepage.

4. Example use case: Recording of primary and stimulated echo in a stimulated echo sequence

Recording both a Hahn echo and a stimulated echo attenuation curve in static field gradient NMR allows separation of T_2 and diffusion effects [3].

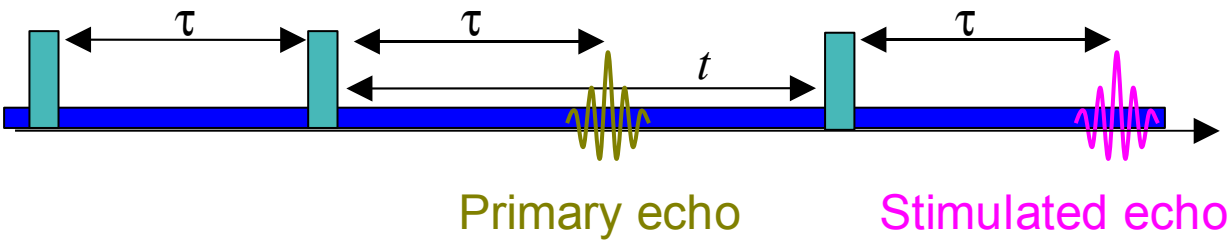

Fig. 3: Static field gradient stimulated echo sequence with primary and stimulated echo. If multiple triggering is available both echoes can be read out in one run of the sequence.

Traditionally two separate runs were made: The Hahn echo experiment (H) was conducted separately and afterwards the stimulated echo (S) has been recorded in another experiment run.

$$H(\tau)=\exp(-\frac{2}{3}\gamma^2G^2\tau^3D)\cdot\exp(-\frac{2\tau}{T_2}) \quad (1)$$

$$S(\tau,t)=\exp(-\gamma^2G^2D\tau^2(t+\frac{2}{3}\tau))\cdot\exp(-\frac{2\tau}{T_2})\cdot\exp(-\frac{t}{T_1}) \quad (2)$$

The echoes evolve differently in τ [17], so it is possible to derive both parameters by fitting the two datasets simultaneously.

An alternative approach is to use the primary echo (P), which arises inside the stimulated echo sequence. It has half the amplitude of the Hahn echo and follows the same diffusion attenuation [18]. By dividing both echo amplitudes, one can remove the influence of T_2:

$$\frac{S(\tau)}{P(\tau)}\propto\exp(-\gamma^2G^2D\tau^2t) \quad (3)$$

DAMARIS supports multiple signal acquisition intervals, which can be placed arbitrarily in the sequence. The following pulse program definition will serve as example: Pulse sequences are housed in Python functions, so essential parameters become arguments of that function:

```
def stim_echo(repetitiontime, tau1, tau2, pi, det_phase, cycle):
```

A simple experiment from scratch will be defined, so one starts definition like that:

```
exp=Experiment()
exp.set_description("tau1", tau1)
exp.set_description("tau2", tau2)
```

The previous two commands attribute the values τ_1 and τ_2 to this single shot. The times assigned here are the actual time intervals as they can be found in literature. The technical details are tackled in the code below and by the “back end”. The “back end” copies these attributes to the corresponding single shot data, so the result processing script will handle them properly.

The next lines declare the phase cycles for pulses and accumulation; for more pulses calculation rules may be used replacing long lists:

```
# phases of rf pulses used later
cycle_phase1=(  0,  0,  0,  0,180,180,180,180)[cycle%8]
cycle_phase2=(  0,  0,180,180,  0,  0,180,180)[cycle%8]
cycle_phase3=(  0,180,  0,180,  0,180,  0,180)[cycle%8]
# transmit corresponding signal phase to accumulation
exp.set_description("det_phase1", (0,0,0,0,
                                   180,180,180,180)[cycle%8])
exp.set_description("det_phase2", (0,180,180,0,
```

```
                                        180,0,0,180)[cycle%8])
```

Here the very pulse sequence is written:

```
    # wait necessary mulitple of T1 before starting
    exp.set_frequency(frequency, cycle_phase1)
    exp.wait(repetitiontime)
    # first pulse
    exp.ttl_pulse(2e-6,   GATE)
    exp.ttl_pulse(pi/2.0, PULSE)
    # wait dephasing time tau1
    exp.set_phase(cycle_phase2)
    exp.wait(tau1-(2.5e-6)-pi/2.0)
    # second pulse
    exp.ttl_pulse(2e-6,   GATE)
    exp.ttl_pulse(pi/2.0, PULSE)
    # set detection phase and wait for echo
    exp.set_phase(det_phase)
    exp.wait(tau1-(9.5e-6)-pi/4.0)
    #first recording (interval lasts 210 µs)
    exp.record(1024*4, 20e6, 210e-6)
    exp.set_phase(cycle_phase3)
    # wait rest of mixing time tau2
    exp.wait(tau2-(202.5e-6)-tau1-pi/4.0)
    # third pulse
    exp.ttl_pulse(2e-6,   GATE)
    exp.ttl_pulse(pi/2.0, PULSE)
    # set detection phase and wait echo time
    exp.set_phase(det_phase)
    exp.wait(tau1-(9.5e-6)-pi/4.0)
    exp.record(1024*4, 20e6)
    return exp
```

The experiment is subsequently defined by Python command lines, which allow arbitrary mathematical operations to place pulses and calculate their parameters. In this example the acquisition intervals are chosen in a way that the echoes' maxima occur at 10 µs after start of the acquisition interval, so that a suitable data processing script can easily find and evaluate the maxima. Because this is a subroutine, the defined and parametrized pulse sequence must be returned to the calling program.

The main program named “experiment” uses this pulse sequence definition inside nested loops for the desired parameter ranges and repetitions:

```
def experiment():
    # declaration of relevant values
    t1 = 0.117289 # s
    pi = 3.0e-6 # at 20 db attenuation
    phase0 = -20 # deg
    # define tau2 dependent ranges for tau1
    tau1_dict={ 1e-3 : log_range(80e-6, 400e-6, 20),
                5e-3 : log_range(80e-6, 250e-6, 20),
                10e-3: log_range(80e-6, 150e-6, 20)}
    # nested loops vary parameters
    for tau2 in [1e-3, 5e-3, 10e-3]:
        for tau1 in tau1_dict[tau2]:
            for accu in xrange(1600):
                e=stim_echo(t1*3, tau1, tau2, pi, phase0, accu)
                yield e
```

DAMARIS is now ready to run – it is possible to save each single shot's raw data for extensive inspection. Normally one prefers to process these data immediately by a "result script":

In the following only the separation of the two data sets with subsequent baseline correction, phase rotation and accumulation for each echo will be explained. Everything happens again in a dedicated Python function called "result". The "for" loop is fed with the single shots one after the other. The variable "data" holds all data that go to the monitor: The experiment can be observed by browsing through the contents of "data" by name in the graphical "front end". In addition its contents is saved periodically to HDF files in order to prevent data loss in the rare case a crash occurs during a long run.

```
def result():
    for timesignal in results:
        # provide raw single scan to monitor
        data["timesignal"]=timesignal
        # read pulse sequence's time constants
        tau1=float(timesignal.get_description("tau1"))
        tau2=float(timesignal.get_description("tau2"))
        both_param="tau1=%g,tau2=%g"%(tau1,tau2)
        # baseline correction and phase rotation for first echo
        timesignal1=timesignal.get_result_by_index(0)
        baseline_correction(timesignal1, 30e-6, 400e-6)
        p1=float(timesignal.get_description("det_phase1"))
        rotate_signal(timesignal1, -p1)
        # insert digital filtering, clipping, fft here
        # accumulate (and create a new data set if necessary)
        if both_param+",echo1" not in data:
            data[both_param+",echo1"]=Accumulation(error=True)
        data[both_param+",echo1"]+=timesignal1
        # similar code for second echo
        # ...
```

Here DAMARIS tries to be as explicit as possible. It should be obvious how data are treated and this procedure should be adaptable in all details. A "swiss knife" results' processing script can be found on the DAMARIS website. Robust echo height estimation tools – suitable for noisy data – are available on the homepage, too. The creation of plots with error-bars and Fourier transform features are explained in the documentation, which is on-hand online [1].

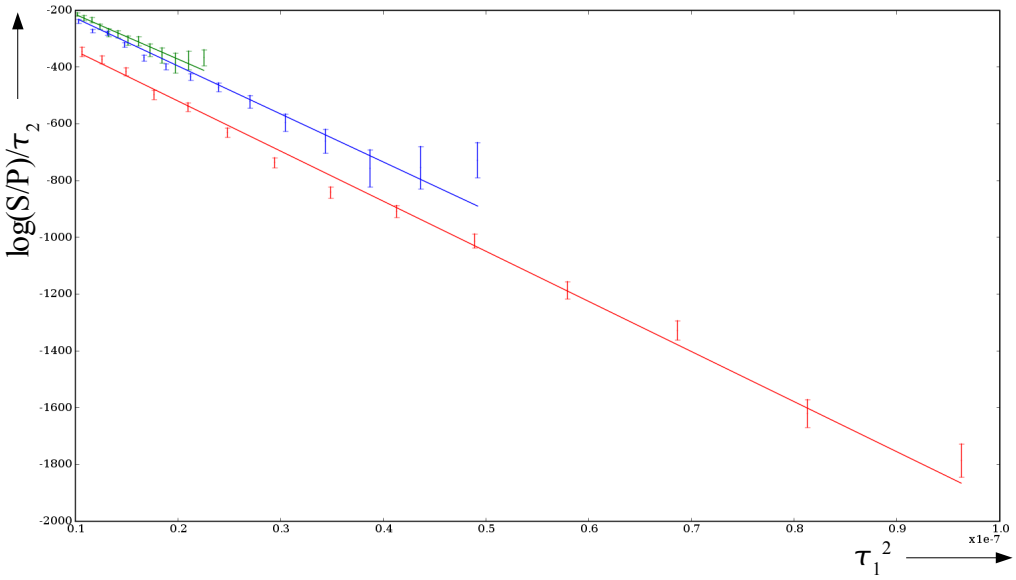

Fig. 4: Results of *ex post* evaluation script based on scipy [14] for diffusion measurement for glycerol at 25°C.

The axes are chosen to indicate the proportionality of (3) explicitly.

When the experiment is done, the postprocessing of these data will take place. Either the result script creates a plain text file with all necessary echo heights or one uses the HDF dump file. The *ex post* analysis will be done outside the python "front end" and thereby it is not limited to python. The HDF file can be processed by Matlab, Mathematica, hdfview [15] and Python. In this case a dedicated diffusion analysis with Python [11] and the scipy [13] packages provides automatically the diffusion coefficient for a glycerol sample at 25°C and G=156 T/m as graphics output (see figure 4) and text:

```
#> python EvalStimEcho.py DAMARIS_data_pool.h5
D(tau2=0.001)=1.0163e-11 +/- 1.83801e-13 (1.8 %) [red]
D(tau2=0.01)=9.07037e-12 +/- 7.23758e-13 (8.0 %) [green]
D(tau2=0.005)=9.73602e-12 +/- 3.27998e-13 (3.4 %) [blue]
```

Compared to literature, this value is too high at a factor of 4, due to traces of water in the glycerol [19].

5. Conclusions

By now the software is in routine use on five spectrometers with different hardware configurations in Darmstadt. The software is portable for Windows and Linux – only limited by the hardware vendors' drivers. Also spectrometers in Dortmund and Berlin were set up.

The software design aims to research labs, which extend standard pulse sequences, integrate arbitrary devices to standard NMR setups to pursue their investigations. It is designed to serve scientific needs of explicit data handling and novel approaches to signal processing

The open-source approach allows to shares improvements and bug fixes among the user community for mutual progress. Also the exchange of pulse programs, data processing features and implementation of new hardware is easily possible.

In order to develop its potential fully, the DAMARIS community needs to grow, involvement of many labs of different branches of NMR is highly welcome and desired.

References

[1] DArmstadt MAgnetic Resonance Instrument Software (DAMARIS) user page and documentation http://www.fkp.physik.tu-darmstadt.de/damaris/

[2] Single-sided and semisingle-sided NMR sensors for highly diffusive samples: Application to bottled beverages, H. Stork, A. Gädke, N. Nestle, J. Agric. Food Chem. 54(15), 2006

[3] Water self-diffusion studies in complex materials with fast-relaxing components: Static and pulsed field methods revisited, Achim Gädke, Karen Friedemann, Petrik Galvosas, Frank Stallmach, Jörg Kärger, Nikolaus Nestle, accepted to Diffusion Fundamentals, 2007

[4] Apparent Longitudinal Relaxation of Mobile Spins in Thin, Periodically Excited Slices, A. Gädke, N. Nestle, Diffusion Fundamentals 2(71), 2005

[5] An earth's field nuclear magnetic resonance apparatus [...] under Antarctic conditions, P.T. Callaghan e.a., Rev. Sci. Instrum. 68(11), 1997

[6] universal MS-DOS based NMR spectrometer software by Gerald Hinze, used in Mainz, Dortmund and Darmstadt (unpublished)

[7] LabView based NMR spectrometer software by Markus Nolte, used in Darmstadt (unpublished)

[8] ODIN - Object-oriented development interface for NMR, T.H. Jochimsen, M. von

Mengershausen, Journal of Magnetic Resonance 170 (1), 2004
[9] Strategies for solid-state NMR in high-field Bitter and hybrid magnets, P. J. M. van Bentum, J. C. Maan, J. W. M. van Os, A. P. M. Kentgens, Chem. Phys. Let. 376(3-4), 2003
[10] LabView by National Instuments http://www.ni.com/labview
[11] Python scripting language homepage http://www.python.org/
[12] GTK “The GIMP Toolkit” http://www.gtk.org/
[13] Hans Petter Langtangen: Python Scripting for Computational Science, Springer, 2nd edition 2005
[14] Python scientific extension http://www.scipy.org/
[15] Numerical Python extension http://www.numpy.org/
[16] Hierarchical Data Format: HDF-group homepage http://hdf.ncsa.uiuc.edu/
[17] Measurement of Translational Molecular Diffusion Using Ultrahigh Magnetic Field Gradient NMR, Burkhard Geil, Concepts in Magnetic Resonance, 10(5), 1998
[18] Spin Echoes, E.L. Hahn, Physical Review, 80(4), 1950
[19] Diffusion Coefficients for the Binary System Glycerol+Water at 25°C, J. Chem Eng. Data, 49(2004)