Supporting Information: Deeper Insight into Photopolymerization: The Synergy of Time-Resolved Non-Uniform Sampling and Diffusion NMR

Kristina Kristinaityte ,[†] Adam Mames,[†] Mariusz Pietrzak,[†] Franz F. Westermair,[‡] Wagner Silva,[‡] Ruth M. Gschwind,[‡] Tomasz Ratajczyk,[†] and Mateusz Urbańczyk[*,†]

†*Institute of Physical Chemistry, Polish Academy of Sciences, Kasprzaka 44/52, 01-224 Warsaw, Poland*

‡*Faculty of Chemistry and Pharmacy, Univeristy of Regensburg, Universitätsstraße 31, 93053 Regensburg, Germany*

E-mail: murbanczyk@ichf.edu.pl

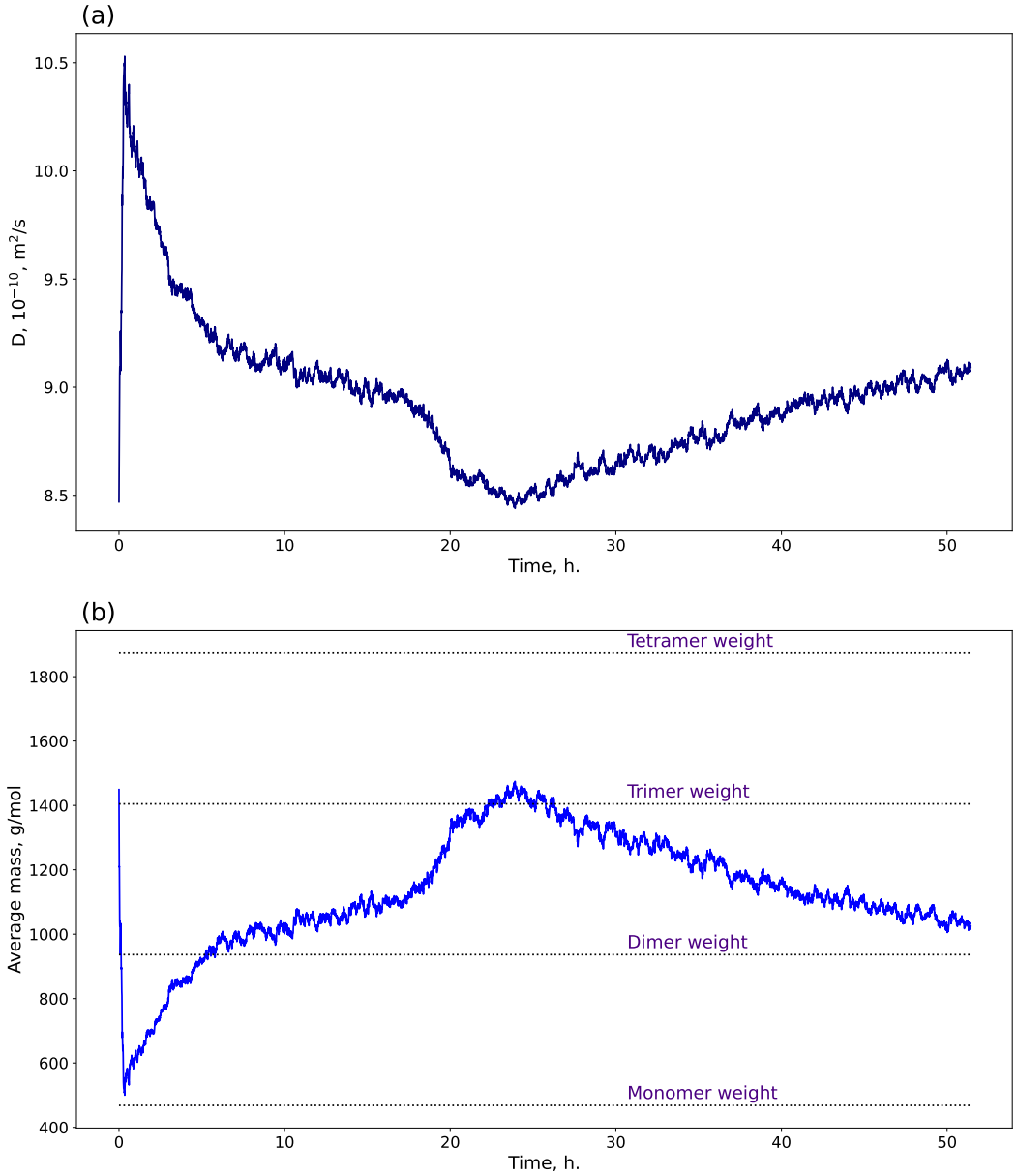


Figure SI.1: a) The diffusion coefficient time dependence calculated from sequential TR-DOSY monitoring of photo-polymerization of H2banthb; b) The average mass time dependence calculated from b). The mass calibration was done using diffusion coefficients and masses of anthracene, dimer of anthracene and H2banthb.

Figure SI.2: Picture of the NMR tube after reaction. On the bottom of the tube there is visible precipitate from the higher mass polymers.

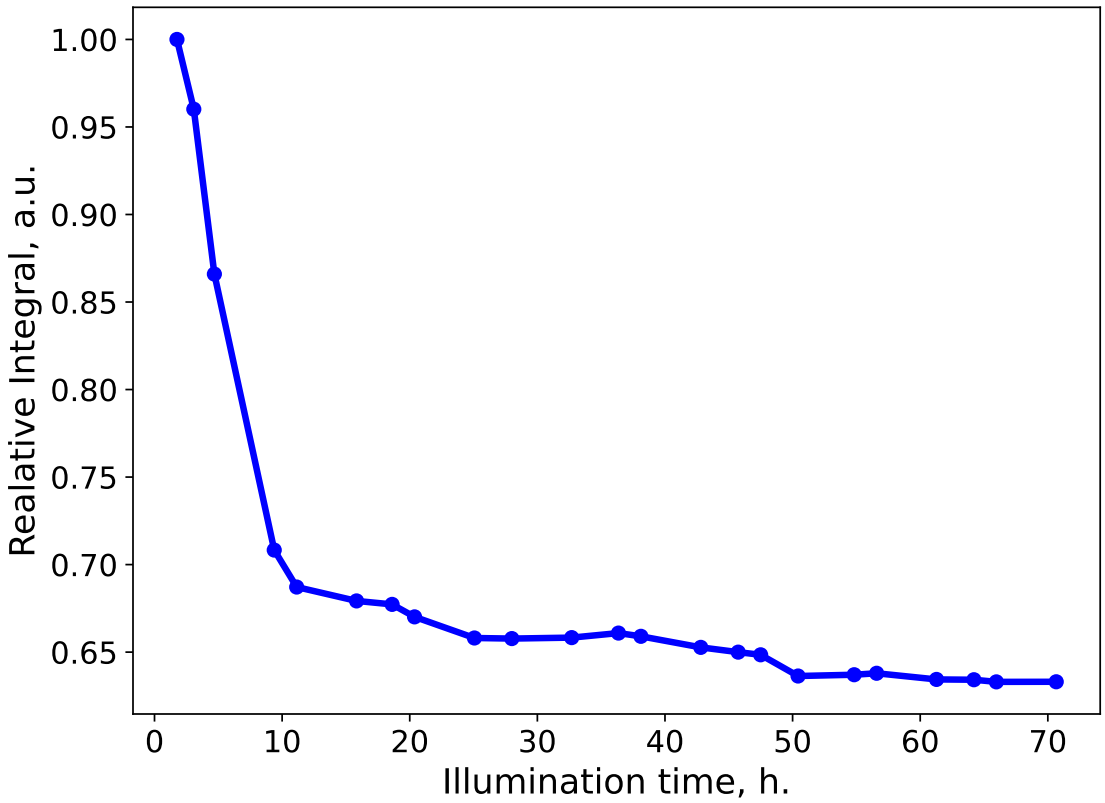


Figure SI.3: The relative integral of the ^{1}H NMR spectrum taken from diffusion data with the lowest gradient strength.

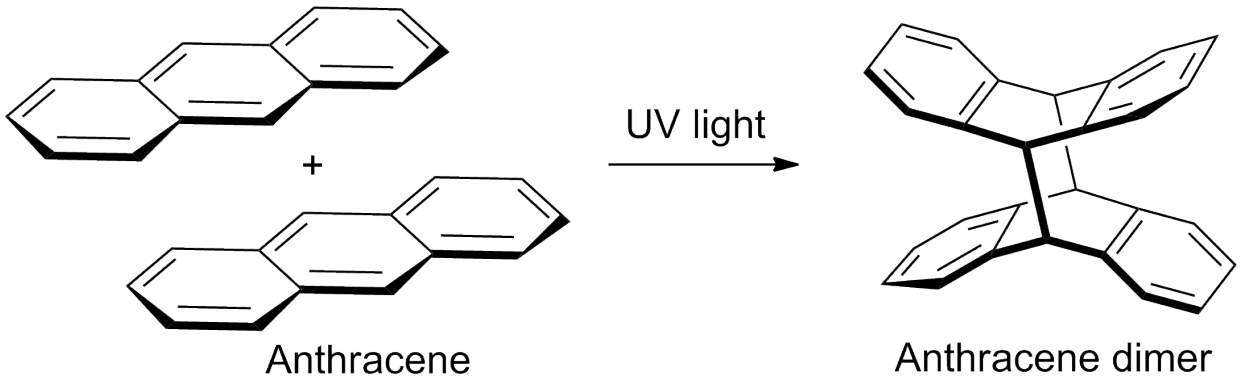


Figure SI.4: The scheme of the reaction of photo-dimerization of anthracene. Used for mass calibration.

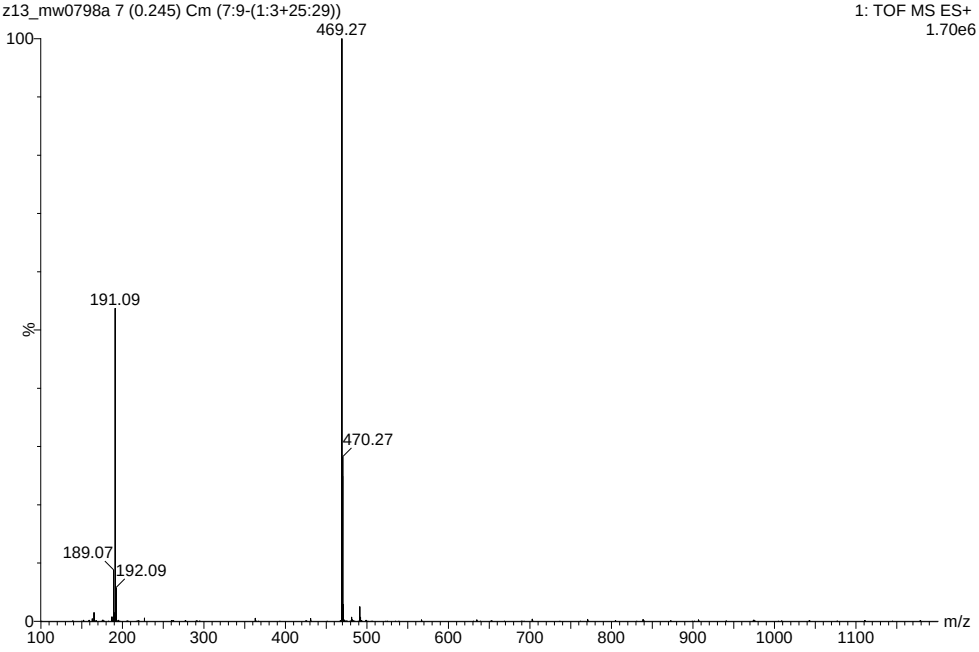
HRMAS of H2banthb

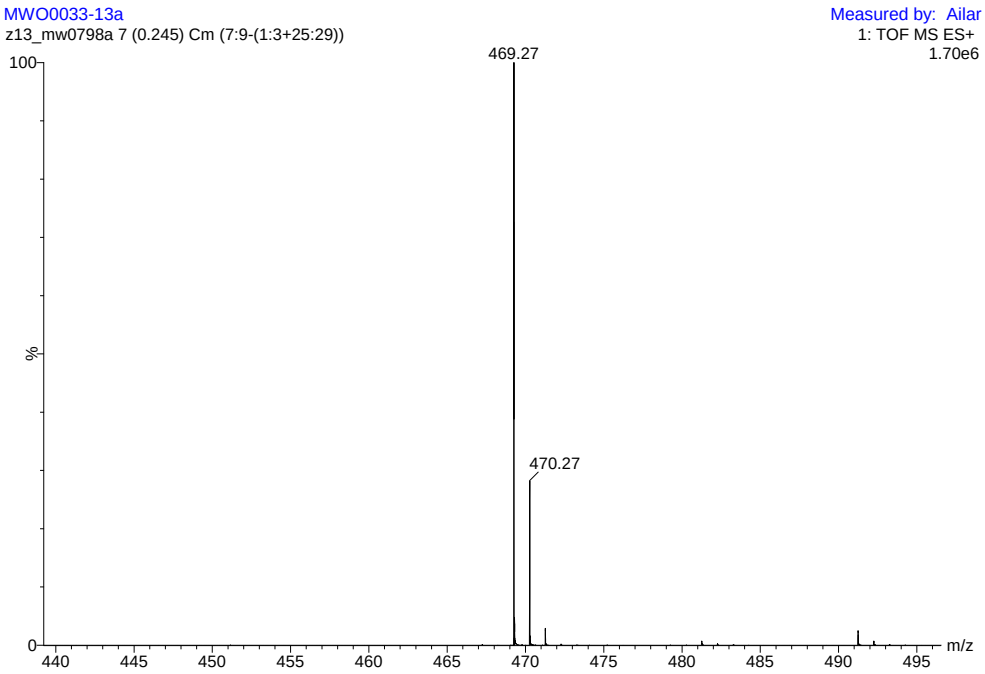
Single Mass Analysis
Tolerance = 3.0 mDa / DBE: min = -1.5, max = 300.0
Element prediction: Off
Number of isotope peaks used for i-FIT = 3
Monoisotopic Mass, Even Electron Ions
20 formula(e) evaluated with 1 results within limits (all results (up to 1000) for each mass)
Elements Used:
C: 0-100 H: 0-100 N: 0-2

Mass	Calc. Mass	mDa	PPM	DBE	Formula	i-FIT	i-FIT Norm	Fit Conf %	C	H	N
469.2643	469.2644	-0.1	-0.2	19.5	C34 H33 N2	960.3	n/a	n/a	34	33	2

MWO0033-13a
z13_mw0798a 7 (0.245) Cm (7:9-(1:3+25:29))

Measured by: Ailar
1: TOF MS ES+
1.70e6




MWO0033-13a
z13_mw0798a 7 (0.245) Cm (7:9-(1:3+25:29))
Measured by: Ailar
1: TOF MS ES+
1.70e6
469.27
470.27
100
%
0
440
445
450
455
460
465
470
475
480
485
490
495
m/z

The choice of gradient sampling

The gradient range was chosen manually based on a series of simulated diffusion decays 7e-10,8e-10,9e-10,8.5e-10,1e-9 m^2/s. The sampling was chosen to be exponential and to guarantee the best separation of decays for aforementioned diffusion coefficients (See Figure SI.5).

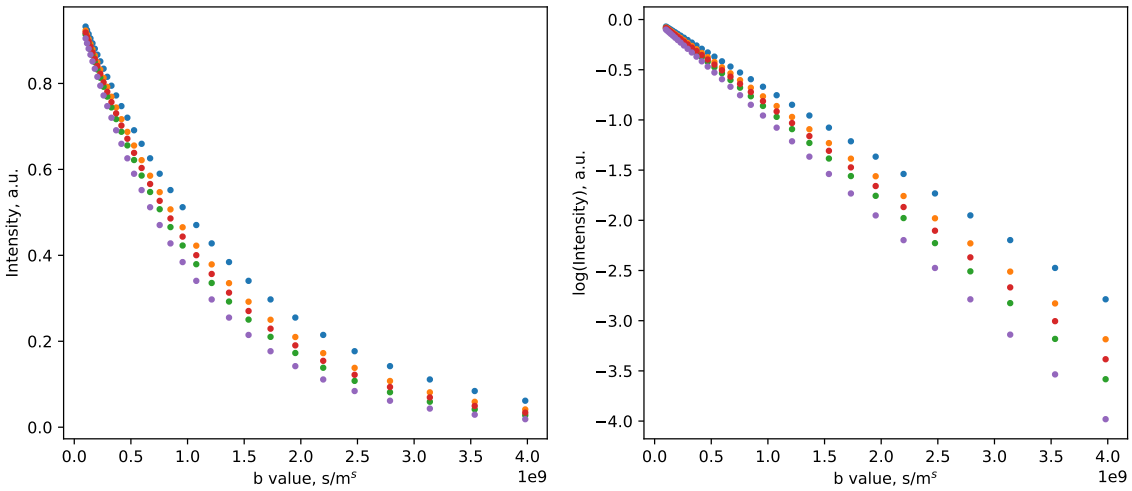


Figure SI.5: The simulated decays of the diffusion coefficients: 7e-10,8e-10,9e-10,8.5e-10,1e-9 m^2/s

Pulse sequences used

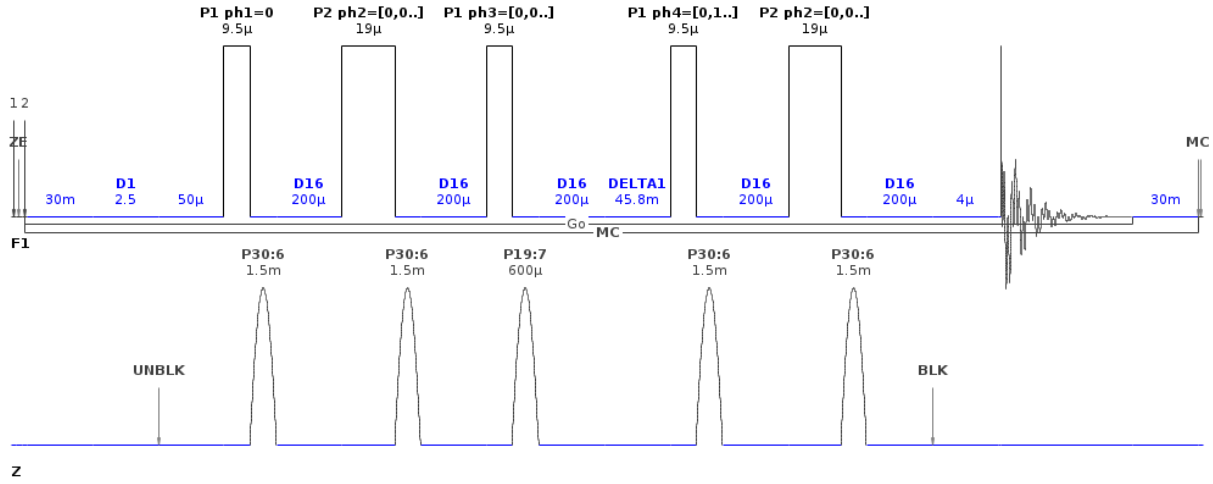


Figure SI.6: Pulse sequence: stebpgp1s1d

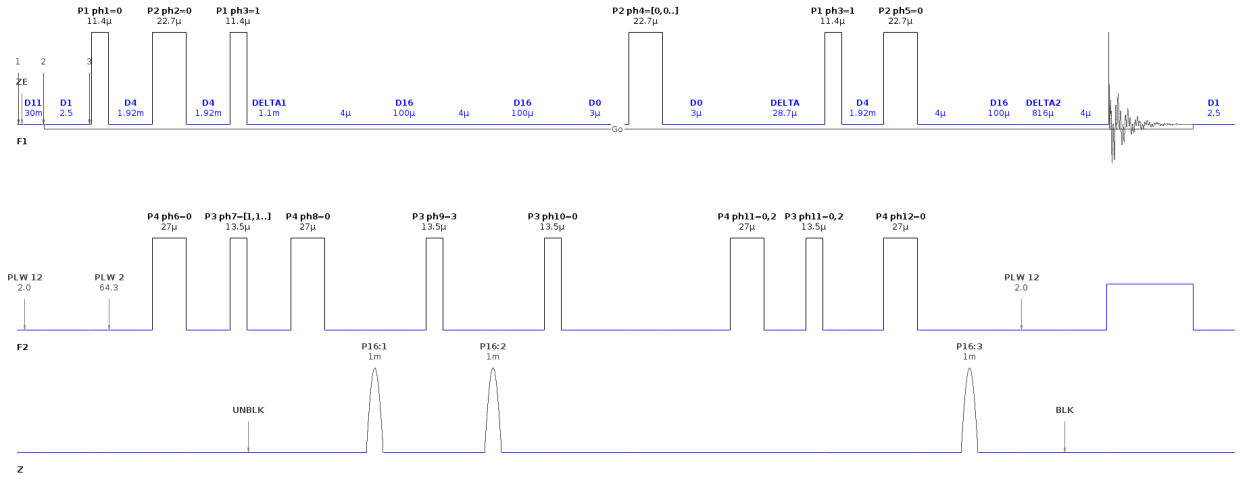


Figure SI.7: Pulse sequence: hsqcgpph

Acquisiton code based on TReNDS:

```
#!/usr/local/bin/jython
# -*- coding: utf-8 -*-
import os
import shutil
import threading
import time
import datetime
from java.awt import *
from javax.swing import *
from javax.swing.filechooser import FileNameExtensionFilter


class TopTReND(JFrame):
```

```
14      def __init__(self):
15          super(TopTReND, self).__init__()
16          self.initUI()
17
18      def initUI(self):
19          stepsTXT = JLabel('Steps for IL acquisition:   ')
20          Spacer1 = JLabel(' \n')
21          Exp1 = JLabel('\n\n\n*******************2D exp No
            ↪  1*******************\n\n\n')
22          Exp2 = JLabel('\n*******************2D exp No
            ↪  2*******************\n')
23          Exp3 = JLabel('\n*******************2D exp No
            ↪  3*******************\n')
24          ExpTR = JLabel('\n*******************TR DOSY exp
            ↪  *******************\n')
25
26          Spacer = JLabel(' \n\n ****************************************\n\n
            ↪  ')
27          Spacer2 = JLabel(' \n\n ****************************************\n\n
            ↪  ')
28          numof = JLabel('Number of 2D exp:  ')
29          self.samp = JButton('Choose sampling schedule',
            ↪  actionPerformed=self.FileChooser)
30          self.samp2 = JButton('Choose sampling schedule',
            ↪  actionPerformed=self.FileChooser2)
31          self.samp3 = JButton('Choose sampling schedule',
            ↪  actionPerformed=self.FileChooser3)
```

```
32          self.Patern2D = JButton('Choose experiment to be used as 2D
            ↪  template', actionPerformed=self.TwodimPatern)
33          self.Patern2D_1 = JButton('Choose experiment to be used as 2D
            ↪  template', actionPerformed=self.TwodimPatern2)
34          self.Patern2D_2 = JButton('Choose experiment to be used as 2D
            ↪  template', actionPerformed=self.TwodimPatern3)
35          self.Patern2D_tr = JButton('Choose experiment to be used as TR DOSY
            ↪  template', actionPerformed=self.TwodimPaternTR)
36          self.sampTR = JButton('Choose sampling schedule',
            ↪  actionPerformed=self.FileChooserTR)
37
38          self.Patern1D = JButton('Choose experiment to be used as 1D
            ↪  template', actionPerformed=self.OnedimPatern)
39          self.SaveDir = JButton('Choose Directory to save data',
            ↪  actionPerformed=self.DirChooser)
40          self.RunBTN = JButton('Start Acquisition!',
            ↪  actionPerformed=self.onRunt)
41          self.StopBTN = JButton('Stop Acquisition!',
            ↪  actionPerformed=self.onStop)
42          self.StopBTN.setEnabled(False)
43          spin2 = SpinnerNumberModel(100.0, 1.0, 999999.0, 1.0)
44          spin = SpinnerNumberModel(1.0, 1.0, 3.0, 1.0)
45          self.numofexp = JSpinner(spin)
46          self.steps = JSpinner(spin2)
47          self.twodionly = JCheckBox('2D acquisition only')
48          self.setTitle('TReNDS  Acquisition module for TopSpin with TR-DOSY')
            ↪  # create window with title
49          self.setSize(600, 700)  # set window size x, y
```

```
50          self.panel1 = JPanel()
51          self.panel2 = JPanel()
52          self.panel4 = JPanel()
53          self.panel1.setAlignmentX(Component.LEFT_ALIGNMENT)
54          self.panel2.setAlignmentX(Component.LEFT_ALIGNMENT)
55          self.panel4.setAlignmentX(Component.LEFT_ALIGNMENT)
56          layoutout1 = BoxLayout(self.panel1, BoxLayout.Y_AXIS)
57          layoutout2 = BoxLayout(self.panel2, BoxLayout.X_AXIS)
58          layoutout3 = BoxLayout(self.panel4, BoxLayout.X_AXIS)
59          self.panel1.setLayout(layoutout1)
60          self.panel2.setLayout(layoutout2)
61          self.panel4.setLayout(layoutout3)
62          self.setLayout(FlowLayout())  # layout manager for horizontal
            ↪  alignment
63          self.add(self.panel1)
64          self.panel1.add(self.panel2)
65          self.panel1.add(JLabel('\n'))
66          self.panel1.add(self.Patern1D)
67          self.panel1.add(self.twodionly)
68          self.panel1.add(JLabel('\n'))
69          self.panel1.add(Exp1)
70          self.panel1.add(JLabel('\n'))
71          self.panel1.add(self.samp)
72          self.panel1.add(self.Patern2D)
73          self.panel1.add(JLabel('\n'))
74          self.panel1.add(Exp2)
75          self.panel1.add(JLabel('\n'))
76          self.panel1.add(self.samp2)
```

```
77          self.panel1.add(self.Patern2D_1)
78          self.panel1.add(JLabel('\n'))
79          self.panel1.add(Exp3)
80          self.panel1.add(JLabel('\n'))
81          self.panel1.add(self.samp3)
82          self.panel1.add(self.Patern2D_2)
83          self.panel1.add(JLabel('\n'))
84          self.panel1.add(ExpTR)
85          self.panel1.add(JLabel('\n'))
86          self.panel1.add(self.sampTR)
87          self.panel1.add(self.Patern2D_tr)
88          self.panel1.add(JLabel('\n'))
89          self.panel1.add(Spacer2)
90          self.panel1.add(JLabel('\n'))
91          self.panel1.add(self.SaveDir)
92          self.panel1.add(Spacer1)
93          self.panel1.add(Spacer)
94          self.panel1.add(self.panel4)
95          self.panel4.add(self.RunBTN)
96          self.panel4.add(JLabel('\t'))
97          self.panel4.add(self.StopBTN)
98          self.panel2.add(stepsTXT)
99          self.panel2.add(self.steps)
100         self.panel2.add(numof)
101         self.panel2.add(self.numofexp)
102         self.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE)
103         self.setVisible(True)
104         self.setAlwaysOnTop(True)
```

```
105
106     def FileChooser(self, e):
107         self.setAlwaysOnTop(False)
108
109         chooseFile = JFileChooser()
110         filter1 = FileNameExtensionFilter("sampling schedule", [".sch"])
111         chooseFile.addChoosableFileFilter(filter1)
112
113         ret = chooseFile.showDialog(JPanel(), "Choose file")
114
115         if ret == JFileChooser.APPROVE_OPTION:
116             file = chooseFile.getSelectedFile()
117             self.sampfile = file.getCanonicalPath()
118             mes = "Sampling file: " + self.sampfile + " have been chosen"
119             SHOW_STATUS(message=mes)
120             self.samp.setText(mes)
121         self.setAlwaysOnTop(True)
122
123     def FileChooser2(self, e):
124         self.setAlwaysOnTop(False)
125
126         chooseFile = JFileChooser()
127         filter1 = FileNameExtensionFilter("sampling schedule", [".sch"])
128         chooseFile.addChoosableFileFilter(filter1)
129
130         ret = chooseFile.showDialog(JPanel(), "Choose file")
131
132         if ret == JFileChooser.APPROVE_OPTION:
```

```
133             file = chooseFile.getSelectedFile()
134             self.sampfile2 = file.getCanonicalPath()
135             mes = "Sampling file: " + self.sampfile + " have been chosen"
136             SHOW_STATUS(message=mes)
137             self.samp2.setText(mes)
138         self.setAlwaysOnTop(True)
139 
140     def FileChooser3(self, e):
141         self.setAlwaysOnTop(False)
142 
143         chooseFile = JFileChooser()
144         filter1 = FileNameExtensionFilter("sampling schedule", [".sch"])
145         chooseFile.addChoosableFileFilter(filter1)
146 
147         ret = chooseFile.showDialog(JPanel(), "Choose file")
148 
149         if ret == JFileChooser.APPROVE_OPTION:
150             file = chooseFile.getSelectedFile()
151             self.sampfile3 = file.getCanonicalPath()
152             mes = "Sampling file: " + self.sampfile + " have been chosen"
153             SHOW_STATUS(message=mes)
154             self.samp3.setText(mes)
155         self.setAlwaysOnTop(True)
156 
157     def FileChooserTR(self, e):
158         self.setAlwaysOnTop(False)
159 
160         chooseFile = JFileChooser()
```

```
161         filter1 = FileNameExtensionFilter("gradient schedule", [".txt"])
162         chooseFile.addChoosableFileFilter(filter1)
163 
164         ret = chooseFile.showDialog(JPanel(), "Choose file")
165 
166         if ret == JFileChooser.APPROVE_OPTION:
167             file = chooseFile.getSelectedFile()
168             self.sampfileTR = file.getCanonicalPath()
169             mes = "Sampling file: " + self.sampfileTR + " have been chosen"
170             SHOW_STATUS(message=mes)
171             self.sampTR.setText(mes)
172         self.setAlwaysOnTop(True)
173     def DirChooser(self, e):
174         self.setAlwaysOnTop(False)
175 
176         chooseFile = JFileChooser()
177         chooseFile.setFileSelectionMode(JFileChooser.DIRECTORIES_ONLY)
178         filter1 = FileNameExtensionFilter("sampling schedule", [".sch"])
179         chooseFile.addChoosableFileFilter(filter1)
180 
181         ret = chooseFile.showDialog(JPanel(), "Choose file")
182 
183         if ret == JFileChooser.APPROVE_OPTION:
184             file = chooseFile.getSelectedFile()
185 
186             self.dirsave = file.getCanonicalPath()
187             mes = "Acquired data will be saved in : " + self.dirsave + "
                ↪  directory"
```

```
188             SHOW_STATUS(message=mes)
189             self.SaveDir.setText(mes)
190             self.setAlwaysOnTop(True)
191
192             # def TwoDimTemp(e):
193             # def TwoDimTemp(e):
194
195     def TwodimPatern(self, e):
196         self.setAlwaysOnTop(False)
197
198         self.twodimpath = DATASET_DIALOG(
199             "Choose 2D experiment pattern (DO NOT USE FIND OPTION!)")  #
                ↪  PROBLEM WITH FIND BUTTON!!!
200         mes = '2D template is: ' + self.twodimpath[3] + '/' +
            ↪  self.twodimpath[0] + '/' + self.twodimpath[1] + '/' + \
201               self.twodimpath[2]
202         SHOW_STATUS(message=mes)
203         # DATASET_DIALOG("Choose 1D experiment pattern", CURDATA())
204         self.Patern2D.setText(mes)
205         self.setAlwaysOnTop(True)
206
207     def TwodimPatern2(self, e):
208         self.setAlwaysOnTop(False)
209
210         self.twodimpath2 = DATASET_DIALOG(
211             "Choose 2D experiment pattern (DO NOT USE FIND OPTION!)")  #
                ↪  PROBLEM WITH FIND BUTTON!!!
```

```
212         mes = '2D template is: ' + self.twodimpath2[3] + '/' +
            ↪  self.twodimpath2[0] + '/' + self.twodimpath2[1] + '/' + \
213               self.twodimpath2[2]
214         SHOW_STATUS(message=mes)
215         # DATASET_DIALOG("Choose 1D experiment pattern", CURDATA())
216         self.Patern2D_1.setText(mes)
217         self.setAlwaysOnTop(True)
218
219     def TwodimPaternTR(self, e):
220         self.setAlwaysOnTop(False)
221
222         self.twodimpathTR = DATASET_DIALOG(
223             "Choose 2D experiment pattern (DO NOT USE FIND OPTION!)")  #
                ↪  PROBLEM WITH FIND BUTTON!!!
224         mes = '2D template is: ' + self.twodimpathTR[3] + '/' +
            ↪  self.twodimpathTR[0] + '/' + self.twodimpathTR[1] + '/' + \
225               self.twodimpathTR[2]
226         SHOW_STATUS(message=mes)
227         # DATASET_DIALOG("Choose 1D experiment pattern", CURDATA())
228         self.Patern2D_tr.setText(mes)
229         self.setAlwaysOnTop(True)
230     def TwodimPatern3(self, e):
231         self.setAlwaysOnTop(False)
232
233         self.twodimpath3 = DATASET_DIALOG(
234             "Choose 2D experiment pattern (DO NOT USE FIND OPTION!)")  #
                ↪  PROBLEM WITH FIND BUTTON!!!
```

```
235         mes = '2D template is: ' + self.twodimpath3[3] + '/' +
            ↪  self.twodimpath3[0] + '/' + self.twodimpath3[1] + '/' + \
236               self.twodimpath3[2]
237         SHOW_STATUS(message=mes)
238         # DATASET_DIALOG("Choose 1D experiment pattern", CURDATA())
239         self.Patern2D_2.setText(mes)
240         self.setAlwaysOnTop(True)
241
242         # self.twodimpath=DATASET_DIALOG("Choose 1D experiment pattern",
            ↪  CURDATA())
243
244     def OnedimPatern(self, e):
245         self.setAlwaysOnTop(False)
246
247         self.onedimpath = DATASET_DIALOG("Choose 1D experiment pattern (DO
            ↪  NOT USE FIND OPTION!)")
248         mes = '1D template is: ' + self.onedimpath[3] + '/' +
            ↪  self.onedimpath[0] + '/' + self.onedimpath[1] + '/' + \
249               self.onedimpath[2]
250         SHOW_STATUS(message=mes)
251         # DATASET_DIALOG("Choose 1D experiment pattern", CURDATA())
252         # self.twodimpath=DATASET_DIALOG("Choose 1D experiment pattern",
            ↪  CURDATA())
253         self.Patern1D.setText(mes)
254         self.setAlwaysOnTop(True)
255
256     def onStop(self, e):
257         self.eventkill.set()
```

```
258         self.StopBTN.setEnabled(False)
259         self.RunBTN.setEnabled(True)
260         self.RunBTN.setText("Start Acquisition")
261 
262     def onRunt(self, e):
263         self.StopBTN.setEnabled(True)
264         twoditemp = self.twodimpath
265         if self.numofexp.getValue() < 3:
266             self.sampfile3 = ''
267             self.twodimpath3 = ''
268         if self.numofexp.getValue() < 2:
269             self.sampfile2 = ''
270             self.twodimpath2 = ''
271         if self.twodionly.isSelected():
272             self.onedimpath = ''
273         self.eventkill = threading.Event()
274         self.first = threading.Thread(target=self.acquisitionloop,
275                                       args=(self.steps.getValue(),
                                      ↪  self.sampfile,self.twodimpathTR,
                                      ↪  self.twodimpath, self.onedimpath,
276                                            self.dirsave,
                                            ↪  self.twodionly.isSelected(),
                                            ↪  self.sampfile2,
                                            ↪  self.sampfile3,
                                            ↪  self.sampfileTR,
277                                           self.twodimpath2,
                                            ↪  self.twodimpath3,
                                            ↪  self.numofexp.getValue(),
```

```
278                                             self.eventkill))
279         # first.setDaemon(True)
280         self.first.start()
281 
282     def acquisitionloop(self, steps, sampfile, twoditempTR, twoditemp,
    ↪  oneditemp, SaveDir, TwoDiOnly, sampfile2, sampfile3, sampfileTR,
283                         twoditemp2, twoditemp3, numofexp, eventkill):
284         self.RunBTN.setEnabled(False)
285         self.create_interleave_file_structure(SaveDir, numofexp)  # creates
        ↪  subfolders like Acquired/1D Acquired/2D
286         # with open(sampfile) as f:
287         #     schedule1 = f.readlines()
288         f = open(sampfile, 'r')
289         schedule1 = f.readlines()
290         f.close()
291         f = open(sampfileTR, 'r')
292         scheduleTR = f.readlines()
293         f.close()
294         shutil.copy2(sampfileTR, SaveDir + '/Acquired/DOSY/gradients.txt')
295 
296         shutil.copy2(sampfile, SaveDir + '/Acquired/2D/sampling.sch')
297         if int(numofexp) == 3:
298             shutil.copy2(sampfile3, SaveDir + '/Acquired/2D_2/sampling.sch')
299             # with open(sampfile3) as f:
300             #     schedule3 = f.readlines()
301             f = open(sampfile3, 'r')
302             schedule3 = f.readlines()
303             f.close()
```

```
304         if int(numofexp) == 3 or int(numofexp) == 2:
305             shutil.copy2(sampfile2, SaveDir + '/Acquired/2D_1/sampling.sch')
306             # with open(sampfile2) as f:
307             #     schedule2 = f.readlines()
308             f = open(sampfile2, 'r')
309             schedule2 = f.readlines()
310             f.close()
311         W = 0
312         itr=0
313         for i in xrange(2, 2 * int(steps) + 2, 2):
314
315
316             if not eventkill.wait(1):
317                 # txtbtn = "Acq. running. Step  " + str(i / 2) + "/" +
                    ↪  str(steps)
318                 # self.RunBTN.setText(txtbtn)
319                 for k in range(0, int(numofexp)):
320                     mess = "Step" + str(i / 2)
321                     #SHOW_STATUS(message=str(schedule[i]))
322
323                     time.sleep(5)
324                     txtbtn = "Acq. running. Step  " + str(i / 2) + "/" +
                        ↪  str(steps)
325                     self.RunBTN.setText(txtbtn)
326                     # start 2D
327                     if k == 0:
328                         schedule = schedule1
```

```
329                         self.copydataTo(twoditemp, SaveDir + '/Acquired',
                            ↪  '2D', i / 2)
330                         self.openFolderPath(SaveDir + '/Acquired/2D/' + str(i
                            ↪  / 2) + '/pdata/' + twoditemp[2])
331                         time.sleep(5)
332                         fname = SaveDir + '/Acquired/2D/' + str(i / 2) +
                            ↪  '/nusILlist'
333                     elif k == 1:
334                         schedule = schedule2
335                         self.copydataTo(twoditemp2, SaveDir + '/Acquired',
                            ↪  '2D_1', i / 2)
336                         self.openFolderPath(SaveDir + '/Acquired/2D_1/' +
                            ↪  str(i / 2) + '/pdata/' + twoditemp2[2])
337                         time.sleep(5)
338                         fname = SaveDir + '/Acquired/2D_1/' + str(i / 2) +
                            ↪  '/nusILlist'
339                     elif k == 2:
340                         schedule = schedule3
341                         self.copydataTo(twoditemp3, SaveDir + '/Acquired',
                            ↪  '2D_2', i / 2)
342                         self.openFolderPath(SaveDir + '/Acquired/2D_2/' +
                            ↪  str(i / 2) + '/pdata/' + twoditemp3[2])
343                         time.sleep(5)
344                         fname = SaveDir + '/Acquired/2D_2/' + str(i / 2) +
                            ↪  '/nusILlist'
345                     f = open(fname, 'w+')
346                     f.write(schedule[W])
```

```
347                     f.write(schedule[W + 1])  # retest it when the new
                        ↪ generator is ready
348                     f.close()
349                     try:
350                         topspin_path = sys.getEnviron()['XWINNMRHOME']  #
                            ↪ topspin < 3,1
351                     except:
352                         topspin_path = sys.registry['XWINNMRHOME']  # topspin
                            ↪ > 3.1
353                     f = open(topspin_path +
                        ↪ '/exp/stan/nmr/lists/vc/trendnls.txt', 'w+')
354                     f.write(schedule[W])
355                     f.write(schedule[W + 1])  # retest it when the new
                        ↪ generator is ready
356                     # self.RunBTN.setText(str(schedule[i/2]))
357                     f.close()
358                     self.sendcommand('fntype non-uniform_sampling')
359                     self.sendcommand('NUSpoints 2')
360                     self.sendcommand('DS 0')
361                     self.putpar('NUSLIST', 'trendnls.txt')  # ON test xcmd
                        ↪ gives error so putpar is safer
362                     time.sleep(5)
363                     self.sendcommand('zg')
364                     time.sleep(10)
365                     if k == 0:
366                         self.acqtest(i, '2D',
```

```
                                        SaveDir)  # KEEPS loop with sleep until
                                        ↪ audita.txt shows that acquisition is
                                        ↪ running
                    elif k == 1:
                        self.acqtest(i, '2D_1',
                                        SaveDir)  # KEEPS loop with sleep until
                                        ↪ audita.txt shows that acquisition is
                                        ↪ running
                    elif k == 2:
                        self.acqtest(i, '2D_2',
                                        SaveDir)  # KEEPS loop with sleep until
                                        ↪ audita.txt shows that acquisition is
                                        ↪ running

                W += 2
                if not TwoDiOnly:
                    self.copydataTo(oneditemp, SaveDir + '/Acquired', '1D', i
                    ↪ / 2)
                    self.openFolderPath(SaveDir + '/Acquired/1D/' + str(i /
                    ↪ 2) + '/pdata/' + oneditemp[2])
                    time.sleep(10)
                    self.sendcommand('zg')
                    time.sleep(10)
                    self.acqtest(i, '1D', SaveDir)
                # TR DOSY part
                self.copydataTo(twoditempTR, SaveDir + '/Acquired', 'DOSY', i
                ↪ / 2)
```

```
385                 self.openFolderPath(SaveDir + '/Acquired/DOSY/' + str(i / 2)
                    ↪  + '/pdata/' + twoditemp[2])
386                 time.sleep(5)
387
388                 self.sendcommand('GPZ6 ' +str(scheduleTR[itr][:-3]))
389                 time.sleep(5)
390                 itr+=1
391                 self.sendcommand('zg')
392                 time.sleep(10)
393
394                 self.acqtest(i, 'DOSY', SaveDir)
395
396                 # fname = SaveDir + '/Acquired/2D/' + str(i / 2) +
                    ↪  '/nusILlist'
397
398         self.RunBTN.setText("Acquisition Complete")
399         self.RunBTN.setEnabled(True)
400
401     def acqtest(self, i, Mode, SaveDir):
402         AcqActive = True
403         while AcqActive:
404             if 'acquisition in progress' in open(SaveDir + '/Acquired/' +
                ↪  Mode + '/' + str(
405                                 i / 2) + '/audita.txt').read():
406                 SHOW_STATUS('Acquisition in progress')
407                 time.sleep(10)
408             else:
409                 AcqActive = False
```

```
                SHOW_STATUS('Acquisition step finished. Next step')

    def openFolder(self, folderM):

        com = 'RE(["'
        for l in range(0, 3):
            com += folderM[l] + '","'
        com += folderM[l + 1] + '"],"y")'
        SHOW_STATUS(message=com)
        EXEC_PYSCRIPT(com)

    def openFolderPath(self, folderM):
        com = 'RE_PATH("' + folderM + '","y")'
        EXEC_PYSCRIPT(com)

    def savePath(self, folder):
        com = 'WR_PATH("' + folder + '","y")'
        EXEC_PYSCRIPT(com)

    def sendcommand(self, command):
        com = 'XCMD("' + command + '",1)'
        EXEC_PYSCRIPT(com)

    def putpar(self, command, path):
        com = 'PUTPAR("' + command + '",' + '"' + path + '")'
        EXEC_PYSCRIPT(com)

```

```
437     def copydataTo(self, Source, Destination, Mode, counter):  # Mode 2D vs
        ↪  1D counter is from loop
438         Source = Source[3] + '/' + Source[0] + '/' + Source[1] + '/'
439         # SHOW_STATUS(message=Source)
440         # self.openFolder(self, Source)
441         Destination += '/' + Mode + '/' + str(counter)
442         # SHOW_STATUS(message=Destination)
443         # # time.sleep(5)
444         shutil.copytree(Source, Destination)
445 
446         # self.savePath(Destination)
447         # # time.sleep(5)
448         if Mode == '1D':
449             try:
450                 os.remove(Destination + '/fid')
451             except:
452                 pass
453         elif Mode == '2D':
454             try:
455                 os.remove(Destination + '/ser')
456             except:
457                 pass
458         SHOW_STATUS(message='Acq structure created in: ' + Destination)
459 
460 
461     def create_interleave_file_structure(self, Directory, numofexp):
462         # Directory=check_if_dir(Directory)
463         # Creates the folder structure for saving the data in Directory
```

```
464         if not (os.path.isdir(Directory)):
465             os.makedirs(Directory)
466             os.makedirs(Directory + '/Acquired')
467 
468             os.makedirs(Directory + '/Acquired/1D')
469             os.makedirs(Directory + '/Acquired/2D')
470             os.makedirs(Directory + '/Acquired/DOSY')
471 
472         else:
473 
474             if not (os.path.isdir(Directory + '/Acquired')):
475                 os.makedirs(Directory + '/Acquired')
476                 os.makedirs(Directory + '/Acquired/1D')
477                 os.makedirs(Directory + '/Acquired/2D')
478                 os.makedirs(Directory + '/Acquired/DOSY')
479 
480             else:
481                 os.rename(Directory + '/Acquired',
482                           Directory +
                          ↪  '/Acquired_back_at{:%Y_%b_%d_%H_%M_%S}'.format(
483                              datetime.datetime.now()))
484                 os.makedirs(Directory + '/Acquired')
485             if not (os.path.isdir(Directory + '/Acquired/1D')):
486                 os.makedirs(Directory + '/1D')
487             if not (os.path.isdir(Directory + '/Acquired/2D')):
488                 os.makedirs(Directory + '/Acquired/2D')
489         if int(numofexp) == 2 or int(numofexp) == 3:
490             if not (os.path.isdir(Directory + '/Acquired/2D_1')):
```

```
                os.makedirs(Directory + '/Acquired/2D_1')
        if int(numofexp) == 3:
            if not (os.path.isdir(Directory + '/Acquired/2D_2')):
                os.makedirs(Directory + '/Acquired/2D_2')


TopTReND()
```