Supplementary Information (SI) for Analyst.

Supporting Information

Improved NMR-based diffusion measurements for inorganic ions

Maria Clara D. N. G. Leal, Binhan Yu, Tianzhi Wang, and Junji Iwahara

Department of Biochemistry & Molecular Biology, Sealy Center for Structural Biology & Molecular Biophysics, University of Texas Medical Branch, Galveston, Texas 77555-1068, USA

Contents

S1. Pulse program for the BPP spin-echo method for Bruker NMR instruments

This pulse program is for the pulse sequence shown in Figure 1B. It was used with a Bruker Avance III spectrometer operated with TopSpin version 3.6.

```
;BPP spin-echo sequence
;Reference: Leal et al. (2026) Analyst
;
;$CLASS=HighRes
;$DIM=2D
;$TYPE=
;$SUBTYPE=
;$COMMENT=

#include <Avance.incl>
#include <Grad.incl>
#include <Delay.incl>

define list<gradient> diff=<Difframp>

"p2=p1*2"

1 ze
2 d1
  50u UNBLKGRAD
  p1 ph1
  p30:gp6*diff
  d16
  p2 ph2
  p30:gp6*-1*diff
  d16
  p30:gp6*diff
  d16
  p2 ph4
  p30:gp6*-1*diff
  d16
  4u BLKGRAD
  go=2 ph31
  d1 mc #0 to 2 F1QF(calgrad(diff))
exit

ph1= 0
ph2= 0 1 2 3
ph4= 0 0 0 0  1 1 1 1  2 2 2 2  3 3 3 3
ph31=0 2 0 2  2 0 2 0

;pl1: f1 channel - power level for pulse (default)
;p1 : f1 channel -  90 degree high power pulse
;p2 : f1 channel - 180 degree high power pulse
;p19: gradient pulse 2 (spoil gradient)
;p30: gradient pulse (little DELTA * 0.5)
;d1 : relaxation delay; 1-5 * T1
;d16: delay for gradient recovery
;ns : 8 * n
;ds : 4 * m
;td1: number of experiments
;FnMODE: QF
;        use xf2 and DOSY processing
;use gradient power:    gp 6
;                       100
; For Eq. 1 of Leal et al. (2026) Analyst
; [big Delta] = 2.0*p30 + 2.0*d16 + 2.0*p1
; [little delta] = 2.0*p30
:
;for z-only gradients:
;gpz6: 100%
;use gradient files:
;gpnam6: SMSQ10.100
;use AU-program dosy to calculate gradient-file Difframp
```

S2. Python program to determine ionic diffusion coefficients from BPP spin-echo data

This Python program, BPPdiffusion.py, reads the BPP diffusion data processed by the 'xf2' command of the Bruker TopSpin software and determines an ionic diffusion coefficient. The BPP spin-echo data are supposed to be recorded with the pulse program shown on the prior page, whereas the BPP stimulated-echo data are supposed to be recorded with the pulse program 'stebpgp1s' in the Bruker's standard pulse program library. In each case, the TopSpin data directory should contain a 'difflist' file generated by the AU program 'dosy' to run the experiment.

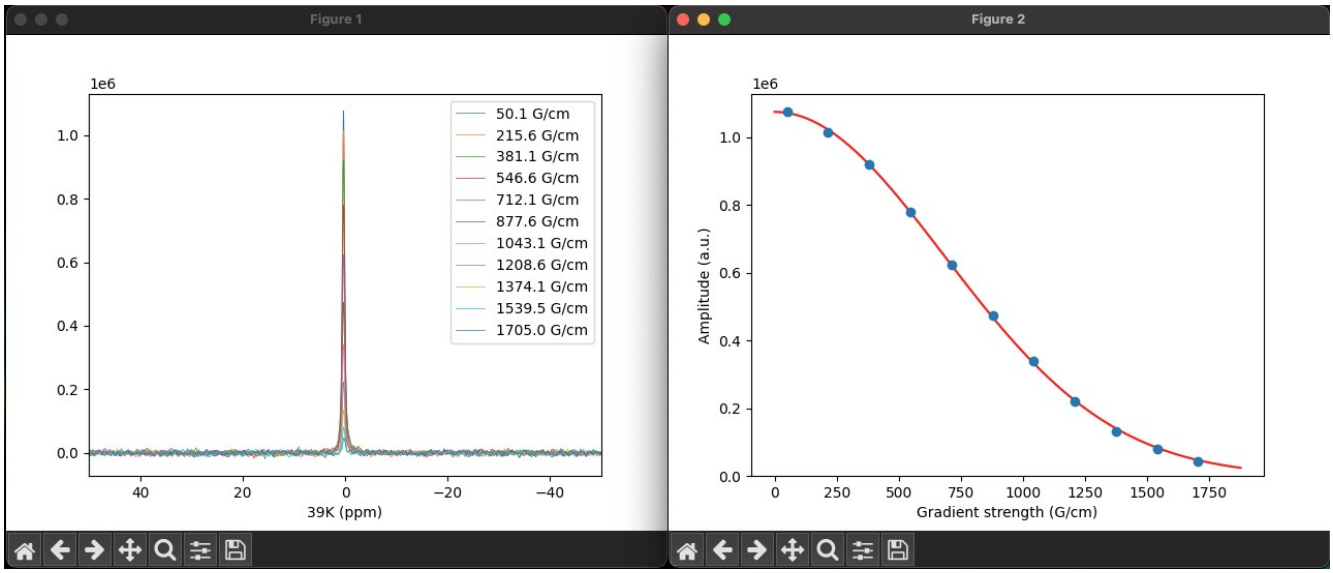
For example, for the BPP spin-echo data located at the directory ~/spec/avan750/Doty39K/120, the program can be executed as follows:

```
> BPPdiffusion.py -d ~/spec/avan750/Doty39K/120 -emode 1 -calibF 1.0722 -smode 1
```

The option '-emode' should be 1 for BPP spin-echo data or 2 for BPP stimulated-echo data. The option '-calibF' is used for correction of the gradient strengths listed in the 'difflist' file. Each gradient strength will be multiplied by the correction factor specified via this option. The option '-smode' is used to select either peak heights ('-smode 1') or integrals ('-smode 2') for the signal intensities. When the integral is used, the options '-b1' and '-b2' should be used to specify the integral boundaries (e.g., '-b1 1.2 -b2 -0.5' when the integral should be calculated for the region between 1.2 and -0.5 ppm). For the above example, the outputs are:

```
11 gradients
 Original in difflist | Corrected
      46.771 G/cm     |     50.148 G/cm
     201.116 G/cm     |    215.637 G/cm
     355.461 G/cm     |    381.125 G/cm
     509.806 G/cm     |    546.614 G/cm
     664.151 G/cm     |    712.103 G/cm
     818.496 G/cm     |    877.591 G/cm
     972.841 G/cm     |   1043.080 G/cm
    1127.186 G/cm     |   1208.569 G/cm
    1281.531 G/cm     |   1374.058 G/cm
    1435.876 G/cm     |   1539.546 G/cm
    1590.221 G/cm     |   1705.035 G/cm

Other experimental parameters
 p1 =   32.00 us
p30 = 1500.00 us
d16 = 0.001500 s
Experiment type: BPP spin-echo (emode = 1)

Nuclei: 39K
Gyromagnetic ratio = 1.2499e+07 T-1 s-1

Use the following conditions for TopSpin 'vargrad' fitting to compare the results.
   Gamma: 198.932 Hz/G
  LITDEL: 3.000 msec
  BIGDEL: 5.314 msec
Note that TopSpin 'vargrad' fitting directly utilize difflist graidents without correction.

Bipolar gradient shape: SMSQ10.100
Sinnaeve parameters: lambda =  0.5000; kappa =  0.3495
=====================================================
Diffusion coefficient: 1.766e-09 +/- 1.269e-11 m2 s-1
=====================================================
```

The program also opens two separate windows showing the overlaid spectra and the data fitting:



The source code of BPPdiffusion.py is given below. A Jupyter version of this Python program is also available upon request to j.iwahara@utmb.edu.

```
#!/usr/bin/env python3
# coding: utf-8

"""
This script reads a BPP spin-echo NMR dataset collected and processed with Bruker TopSpin
and determine the diffusion coefficient for iorganic ions. The data must be processed with
the TopSpin command, 'xf2', before running this script. Thediffusion coefficinet is
determined as described in:
 - Leal, M.C.D.N.G, Yu, B., Wang, T., Iwahara, J. (2026) Magn. Reson. Chem. XXX, XXXX-XX
A BPP spin-echo pulse program for Bruker NMR instruments is also available in the Supporting
Information of this paper.

This script requires nmrglue, NumPy, matplotlib, and SciPy libraries for Python. If your
computer does not have them, you can install them by running:
 > python3 -m pip install nmrglue
 > python3 -m pip install numpy
 > python3 -m pip install matplotlib
 > python3 -m pip install scipy
"""
import argparse
import ast
import nmrglue as ng
import numpy as np
import matplotlib.pyplot as plt
from scipy.optimize import curve_fit

# ------------------------------------------------------
# Parse command-line arguments
# ------------------------------------------------------
parser = argparse.ArgumentParser(
    description="Process BPP diffusion NMR data."
)

parser.add_argument(
    "-dataDir", "-d",
    required=True,
    type=str,
    help="Directory containing the Bruker dataset (e.g., /path/to/expdir)"
)

parser.add_argument(
    "-emode", "-e",
    type=int,
    choices=[1, 2],
```

```
    required=True,
    help="1 = BPP spin-echo, 2 = BPP stimulated-echo"
)

parser.add_argument(
    "-calibF", "-c",
    type=float,
    default=1.0,
    help="PFG correction factor (default = 1 if no correction is needed)"
)

parser.add_argument(
    "-smode", "-s",
    type=int,
    choices=[1, 2],
    required=True,
    help="1 = use peak heights; 2 = use intergral for the selected range"
)

parser.add_argument(
    "-b1",
    type=float,
    default=None,
    help="Integral boundary (chemical shift in ppm): Required only if smode = 2"
)

parser.add_argument(
    "-b2",
    type=float,
    default=None,
    help="Integral boundary (chemical shift in ppm): Required only if smode = 2"
)

args = parser.parse_args()

# ---------------------------------------------------------
# Assign variables
# ---------------------------------------------------------
dataDir = args.dataDir
emode = args.emode
calibF = args.calibF
smode = args.smode
if args.smode == 2:
    if args.b1 is None:
        raise ValueError("smode = 2 requires -b1 and -b2 options")
    if args.b2 is None:
        raise ValueError("smode = 2 requires -b1 and -b2 options")
    rangeIntegral = [args.b1, args.b2]
else:
    rangeIntegral = None   # not used

# Nuclear gyromagnetic ratios
# The values gyromagnetic ratios are from Hennel & Klinowski "Fundamentals of Nuclear
Magnetic Resonance". The values are in 10^7 T-1 s-1 units. You can add other nuclei in the
same format here if necessary.

nucs = []; grs = [];
nucs.append('1H');    grs.append( 26.752196)
nucs.append('7Li');   grs.append( 10.39758)
nucs.append('13C');   grs.append(  6.72828)
nucs.append('15N');   grs.append( -2.712621)
nucs.append('23Na');  grs.append(  7.080416)
nucs.append('25Mg');  grs.append( -1.6389)
nucs.append('31P');   grs.append( 10.8394)
nucs.append('33S');   grs.append(  2.05568)
nucs.append('35Cl');  grs.append(  2.624196)
nucs.append('39K');   grs.append(  1.249928)
nucs.append('133Cs'); grs.append(  3.533253)

# Loading data
```

```
dataFile = dataDir + "/pdata/1"
diffList = dataDir + "/difflist"
dic, data = ng.bruker.read_pdata(dataFile)

dlist0 = np.loadtxt(diffList)
tn=len(dlist0)
dlist = dlist0 * calibF
print("%d gradients" % tn)
print(" Original in difflist | Corrected")
for n in range(tn):
    print("    %8.3f G/cm     |   %8.3f G/cm" % (dlist0[n], dlist[n]))

print("\nOther experimental parameters")
p1  = dic['acqus']['P'][1]*10**(-6);   # p1 in s
p30 = dic['acqus']['P'][30]*10**(-6);  # p30 in s
d16 = dic['acqus']['D'][16];           # d16 in s
d20 = dic['acqus']['D'][20];           # d20 in s
print(" p1 = %7.2f us" % dic['acqus']['P'][1])
print("p30 = %7.2f us" % dic['acqus']['P'][30])
print("d16 = %7.6f s"  % dic['acqus']['D'][16])
if (emode == 1):
    print("Experiment type: BPP spin-echo (emode = %d)" % emode)
    bD = (p30 + d16)*2.0 + p1*2.0
elif (emode == 2):
    print("d20 = %7.6f s"  % dic['acqus']['D'][20])
    print("Experiment type: BPP stimulated-echo (emode = %d)" % emode)
    bD = d20
else:
    raise ValueError("emode must be either 1 (for BPP spin-echo) or 2 (for BPP stimulated
echo)")

ld = p30*2.0
ta = d16
nu=dic['acqus']['NUC1']
ga = 0.0001;
for n in range(len(grs)):
    if (nu == nucs[n]):   ga = grs[n] * 10**7;
if (abs(ga) < 0.1):
    raise ValueError("No matching nuclei")
else:
    print("\nNuclei: %s" % nu)
    print("Gyromagnetic ratio = %7.4e T-1 s-1" % ga)
print("\nUse the following conditions for TopSpin 'vargrad' fitting to compare the
results.")
print("   Gamma: %.3f Hz/G" % (ga/(2.0*np.pi)*10**(-4)))
print("  LITDEL: %.3f msec" % (ld*10**3))
print("  BIGDEL: %.3f msec" % ((bD - ta/2.0)*10**3))
print("Note that TopSpin 'vargrad' fitting directly utilize difflist graidents without
correction.\n")


# Sinnaeve parameter setting
# Here, based on the information of gradient shape, we set Sinnaeve's parameters according
to Sinnaeve, D. (2012) Concepts Magn Reson 40A, 39-65. Here, the parameters lambda and
kappa are set. Note that the parameter sigma is already taken into account when difflist
is created by the AU script 'dosy' of topspin.

gpnam6 = dic['acqus']['GPNAM'][6]
print('Bipolar gradient shape: %s' % gpnam6)
base_gpnam6 = gpnam6.split('.')[0]
if (base_gpnam6 == 'SMSQ10'):
    kap = 422.0/1215.0 + 23.0/(1080.0*(np.pi**2)); # See Table 1 of Sinnaeve
    lam = 0.5;                                     # See Table 1 of Sinnaeve
elif (base_gpnam6 == 'SINE'):
    kap = 3.0/8.0; # See Table 1 of Sinnaeve
    lam = 1.0/2.0; # See Table 1 of Sinnaeve
elif (base_gpnam6 == 'RECT'):
    kap = 1.0/3.0; # See Table 1 of Sinnaeve
    lam = 1.0/2.0; # See Table 1 of Sinnaeve
else:
    raise ValueError("Unsupported gradient shape: %s" % gpnam6)
```

```
print("Sinnaeve parameters: lambda = %7.4f; kappa = %7.4f" % (lam, kap))


# Showing 1D slices

offs  = dic['procs']['OFFSET']
swppm = dic['procs']['SW_p']
e1 = offs; # Edge of the spectrum
e2 = offs-swppm/dic['procs']['SF']; # Edge of the spectrum
xs_ppm = np.linspace(e1,e2,dic['procs']['SI'])
plt.figure()
for n in range(tn):
    plt.plot(xs_ppm,data[n,:],lw = 0.5, label="%.1f G/cm" % dlist[n])
plt.xlabel("%s (ppm)" % nu)
plt.legend()
plt.xlim([max(xs_ppm), min(xs_ppm)]); # To reverse x-axis

# Model function for fitting

def mf(x, Dif, ao):
    q1 = (2.0*kap - 2.0*lam - 1.0)/4.0;            # For Eq.53 of Sinnaeve
    h = ao*np.exp(-Dif*(10**(-12)) *((ga*x*(10**(-2))*ld)**2) *(bD + q1*ld - ta/2.0))
    # 10**(-12) is for um2 s-1 [instead of m2 s-1]
    # 10**(-2) is for G/cm [instead of T/m]
    return h


# ### Nonlinear least-squares fitting and best-fit curve

sa = [];
for n in range(tn):
    ipk = np.argmax(data[0,:]); # Index for the peak position in the 1st spectrum
    ys = data[n,:];
    if (smode == 2) :
        sg = ys[(xs_ppm < max(rangeIntegral)) & (xs_ppm > min(rangeIntegral))]
        sa.append(sum(sg))
    elif (smode == 1) :
        sg = ys[ipk]
        sa.append(sg)
    else:
        raise ValueError("smode must be 1 or 2")

iao = sa[0]; iDif = 100.0;
ivs = [iDif, iao]
pfit,pcov = curve_fit(mf, dlist, sa, ivs)
sxs = np.linspace(0,max(dlist)*1.1,100);
bfc = mf(sxs, pfit[0],pfit[1])
Di = pfit[0]*10**(-12);              # in m2 s-1
eDi = np.sqrt(pcov[0,0])*10**(-12);  # in m2 s-1
print("=====================================================")
print("Diffusion coefficient: %.3e +/- %.3e m2 s-1" % (Di, eDi))
print("=====================================================")
plt.figure()
plt.plot(sxs,bfc,'r')
plt.plot(dlist,sa,'o')
plt.xlabel("Gradient strength (G/cm)")
plt.ylabel("Amplitude (a.u.)")
plt.ylim([0, max(sa)*1.05])
plt.show()
```