Supporting Information

Optimization of TROSY- and anti-TROSY-based ^{15}N CPMG relaxation dispersion experiments through phase cycling

Yingxian Cui,[1] Yangzhuoyue Jin,[1] Yu Hou,[1] Xiaoxu Han,[1] Haiyan Cao,[1]
Lewis E. Kay,[2,3,4,5] and Tairan Yuwen[1]

[1]State Key Laboratory of Natural and Biomimetic Drugs, Department of Pharmaceutical Analysis, School of Pharmaceutical Sciences, Peking University, Beijing 100191, China
[2]Department of Molecular Genetics, University of Toronto, Toronto, Ontario M5S 1A8, Canada
[3]Department of Biochemistry, University of Toronto, Toronto, Ontario M5S 1A8, Canada
[4]Department of Chemistry, University of Toronto, Toronto, Ontario M5S 3H6, Canada
[5]Program in Molecular Medicine, Hospital for Sick Children Research Institute, Toronto, Ontario M5G 1X8, Canada

Table of Contents

```
/*  N15CPMG_trosy_anti_ty_800_cp

    Pseudo 4D for recording trosy/anti-trosy CPMG pulse scheme (15N): Uses l3 as
      variable loop counter to select for trosy (l3%2==0) or anti-trosy (l3%2==1)

    Includes option for gd coherence transfer selection and use of reburp pulse to
      select only the amide 15N spins

    Includes option to apply reburp instead of single 180 in the middle of P-element

    Note:
      1. Identical trosy selection can be achieved by inverting the phase of the
         first 1H 90 after t1 along with the final 15N 90, and then inverting the
         first 4 or last 4 receiver phases. Addition of coherence transfer gradients
         does not change the cycle
      2. ncyc_cp can be odd or even
      3. If gd_sel is used then the min phase cycle is 2
         If gd_sel is not used then the min phase cycle is 4

    Set up: F3 set to 2 when recording trosy, anti-trosy CPMG
            F2 set to the number of nu-cpmg values = number of values in VC list
            F1 set to number of t1 (15N points) real + imag
    Order of acq: t4,t3,t2,t1  - this is always the order for 4Ds

    Written by LEK and TY on June 6th, 2017 based on N15CPMG_NH_trosy_anti_lek_600_v2,
    implement [0013] phase cycle for CPMG pulses instead of constant phase

    Recommended:
      1. Use gd_sel to achieve better TROSY selection
      2. Use s3e_flg if possible, especially on protonated proteins
      3. Use reb_flg to achieve even larger offset coverage
      4. Use N_comp for heating compensation
      5. Use Hheat to achieve consistent heating effects with CW-CPMG (if desired)
      6. Set the number of scans (NS) as multiples of 2^k, where k is as large as
         possible (up to k=4)

    Modified by TY on Oct 26, 2023 to add -Dshp_flg option for using shaped pulses in
    CPMG pulse train

    Modified by TY on Jan 20, 2024 to add -Dc_dec option, with this option 13C AFPs
    are applied during Trelax, and d9 is used for calculating the number of 13C AFPs
    during tau_cp, based on "trunc(tau_cp/d9)"

*/

prosol relations=<triple>

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

/***********************/
/*   Define phases     */
/***********************/
#define zero ph=0.0
#define one ph=90.0
#define two ph=180.0
#define three ph=270.0

/************************/
/*    Define delays     */
/************************/
define delay hscuba      /* length of 1/2 scuba delay */
  "hscuba=30m"

define delay taua        /* 1 / 4J(XH)                */
  "taua=d3"              /* 1s/(4*105) use JNH=105 to decreas the tauxh duration */

define delay taub        /* 1 / 4J(XH)                */
```

```
  "taub=d5"               /* use JNH=95 for tauxh duration */

define delay BigT1
  "BigT1=d14"

define delay BigT
  "BigT=d15"

define delay time_T2
  "time_T2=d17"

#ifdef Hheat
  define delay time_Hheat
    "time_Hheat=d18"
#endif

define delay time_eq
  "time_eq=d19"

define delay tauCPMG
define delay tauCPMG0
define delay tauCPMG1
define delay tauCPMG2
define delay tauCPMG3

"d11=30m"
"in0=inf1/2"              /* t1/2 increment */
"TAU2=0.2u"

#ifdef water
  "DELTA11=aq"
  "DELTA12=larger(d13-2u-p50-d16-20u-4u,TAU2)"

  define pulse pwh_tap
    "pwh_tap=p1*cnst5/90"
#endif

/************************/
/*   Define pulses      */
/************************/
define pulse dly_pg1      /* Messerle purge pulse */
  "dly_pg1=5m"

define pulse dly_pg2      /* Messerle purge pulse */
  "dly_pg2=dly_pg1/1.62"

define pulse pwh
  "pwh=p1"                /* 1H hard pulse at power level pl1 (tpwr) */

define pulse pwn
  "pwn=p3"                /* PW90 for N pulse at power level pl3 (dhpwr2)  */

define pulse pw_sl
  "pw_sl=p13"             /* 1H water selective pulse at power level pl13 (tpwrsl) */

define pulse pw_sl1
  "pw_sl1=p14"            /* 1H water selective pulse at power level pl14 (tpwrsl1) */

define pulse pwc_ad       /* 13C adiabatic pulse at power level spw22(d_ad) */

#if defined(c_flg) || defined(c_dec)
  "pwc_ad=p22"
#endif

#ifdef N_sel
  define pulse pwn_sl
    "pwn_sl=p32"
#endif
```

```
define pulse pwn_cp
  "pwn_cp=p33"            /* PW90 for 15N CPMG pulses */

#ifdef reb_flg
  define pulse pwn_reb
    "pwn_reb=pwn_cp*2.0/0.07981"   /* REBURP pulse length */

  "spoal34=0.5"
  "spoff34=0"
  "spw34=plw33"
#endif /*reb_flg*/

/*******************************/
/*   Define other parameters    */
/*******************************/
define loopcounter Nphase
  "Nphase=1"

define loopcounter ncyc_max  /* max value of ncyc used */
  "ncyc_max=l8"

define list<loopcounter> ncyc_cp=<$VCLIST>

/*****************************************************************/
/* f1180: Start t1 at half dwell to get -90/180 phase correction */
/*        in F1 (15N) dim - set zgoptns -Df1180                  */
/*****************************************************************/
#ifdef f1180
  "d0=(in0/2)"
#else
  "d0=(0.2u/2)"
#endif

/*********************************************************/
/* Assign cnsts to check validity of parameter ranges */
/*********************************************************/
#ifdef fsat
  "cnst10=plw10"          /* tsatpwr - set max at 0.00005W */
#endif

#ifdef mess_flg
  "cnst11=plw11"          /* tpwrmess pl11 - set max at 2.0W */
#endif

#ifdef Hheat
  "cnst12=plw12"          /* power level used for CW CPMG expt */
#endif

"cnst13=spw13"            /* tpwrsl - set max at 0.0015W" */
"spoal13=0"
"spoff13=0"

"cnst14=spw14"            /* tpwrsl1 - set max at 0.0015W" */
"spoal14=1"
"spoff14=0"

#if defined(c_flg) || defined(c_dec)
  "cnst22=spw22"          /* dpwr_ad - set max at 57W */
  "spoal22=0.5"
  "spoff22=0"

  "plw2=0"
#endif

#ifdef N_sel
  "cnst32=spw32"          /* power level for 15N selective pulse */
  "spoal32=0.5"
#endif
```

```
"cnst33=plw33"           /* power level for 15N CPMG Pulses  < 190 W */

#ifdef shp_flg
  "spw28=plw3"
  "spw29=plw3"
  "spw30=plw3"
  "spoal28=0.5"
  "spoff28=0"
  "spoal29=0.5"
  "spoff29=0"
  "spoal30=0.5"
  "spoff30=0"

  define pulse shp_cp
    "shp_cp=p36*(10000*pwn_cp/0.25s)"
  "spoal36=0.5"
  "spoff36=0"
  "spw36=plw33"
#endif

/**********************/
/* Define CPMG pulses */
/**********************/
#ifdef shp_flg
#define cpmg_20 (shp_cp:sp36 ph20):f3
#define cpmg_21 (shp_cp:sp36 ph21):f3
#define cpmg_22 (shp_cp:sp36 ph22):f3
#define cpmg_23 (shp_cp:sp36 ph23):f3
#define cpmg_24 (shp_cp:sp36 ph24):f3
#else
#define cpmg_20 (pwn_cp*2.0 ph20):f3
#define cpmg_21 (pwn_cp*2.0 ph21):f3
#define cpmg_22 (pwn_cp*2.0 ph22):f3
#define cpmg_23 (pwn_cp*2.0 ph23):f3
#define cpmg_24 (pwn_cp*2.0 ph24):f3
#endif
#define CPMG_15N_0 cpmg_20
#define CPMG_15N_F if "l20%2 == 0" {\n cpmg_21 \n}\n else {\n cpmg_22 \n}
#define CPMG_15N_R if "l20%2 == 0" {\n cpmg_23 \n}\n else {\n cpmg_24 \n}

#ifdef c_dec
#define AFP_13C (pwc_ad:sp22 ph26):f2
#else
#define AFP_13C
#endif

/************************/
/* Initialize variables */
/************************/
"l2=0"
"l20=0"
"l21=0"
"l22=0"
"l23=0"
"l24=0"
"l25=0"
"l26=0"
"l27=0"

"acqt0=0"
baseopt_echo

/*******************************/
/* BEGIN ACTUAL PULSE SEQUENCE */
/*******************************/
1 ze
/*****************************************************************/
/* Check validity of parameters and assign values to some of them */
/*****************************************************************/
```

```
#ifdef fsat
  if "cnst10 > 0.00005" {
    2u
    print "error: tsatpwr pl10 too large; < 0.00005W !!!"
    goto HaltAcqu
  }
#endif

#ifdef mess_flg
  if "cnst11 > 2.0" {
    2u
    print "error:tpwrmess pl11 too large; < 2W !!!"
    goto HaltAcqu
  }
#endif

#ifdef Hheat
  if "cnst12 > 20" {
    2u
    print "error: tpwrml pl12 - used for Hheat is too large; < 20W !!!"
    goto HaltAcqu
  }
#endif

  if "cnst13 > 0.0015" {
    2u
    print "error: tpwrsl spw13 too large; < 0.0015W !!!"
    goto HaltAcqu
  }

  if "cnst14 > 0.0015" {
    2u
    print "error: tpwrsl1 spw14 too large; < 0.0015W !!!"
    goto HaltAcqu
  }

#if defined(c_flg) || defined(c_dec)
  if "cnst22 > 80" {
    2u
    print "error: d_ad spw22 is too large; < 80W !!!"
    goto HaltAcqu
  }
#endif

#ifdef N_sel
  if "cnst32 > 120" {
    2u
    print "error: spw32 is too large; < 120W !!!"
    goto HaltAcqu
  }
#endif

  if "cnst33 > 240" {
    2u
    print "error: pl33 for 15N CPMG is too large; < 240W !!!"
    goto HaltAcqu
  }

  if "time_T2 > 101m" {
    2u
    print "error: time_T2 is too large; < 100ms !!!"
    goto HaltAcqu
  }

#ifdef Hheat
  if "time_Hheat > 101m" {
    2u
    print "error: time_Hheat is likely too large; < 100ms !!!"
    goto HaltAcqu
```

```
  }
#endif

  if "d1 < 1.0s" {
    2u
    print "error: d1 must be at least 1s !!!"
    goto HaltAcqu
  }

#ifdef shp_flg
  if "cnst6 < 0 || cnst6 > 1" {
    2u
    print "error: cnst6 should be in the range of 0 and 1 !!!"
    goto HaltAcqu
  }
#endif

#ifdef c_dec
#ifdef shp_flg
  if "d9 < pwc_ad + pwn_cp*2.0/PI + shp_cp*0.5*cnst6 + 2u" {
#else
  if "d9 < pwc_ad + pwn_cp*2.0/PI + pwn_cp*0.75 + 2u" {
#endif
    2u
    print "error: d9 is too short, should be longer than pwc_ad !!!"
    goto HaltAcqu
  }
#endif

2 d11
/************************/
/* Update list pointers */
/************************/
  2u
  "ncyc_cp.idx=l2"
  2u rpp20 rpp21 rpp22 rpp23 rpp24

  4u pl3:f3
  /* Optional, help to obtain slightly more Nz magnetization */
  (pwn*2.0 ph26):f3

/************************************************/
/* Continue with power checks that are run time */
/************************************************/
  ;l20 even for nsdone%8=0,2,5,7, odd for nsdone%8=1,3,4,6
  "l20 = trunc(trunc((nsdone+0.3)/4)+nsdone+0.3)"
  2u

  "l21 = (trunc(ncyc_cp + 0.3))"
  "l22 = (trunc(ncyc_cp - 1 + 0.3))"
  2u

  if "l21 > 80" {
    2u
    print "error: ncyc_cp must be < 81 !!!"
    goto HaltAcqu
  }

  if "ncyc_max < 2" {
    2u
    print "error: l8 must be set to the max ncyc value used !!!"
    goto HaltAcqu
  }

  if "l21 > ncyc_max" {
    2u
    print "error: ncyc_max must be greater than or equal to ncyc_cp !!!"
    goto HaltAcqu
  }
```

```
  if "l21 > 0" {
    "l25 = (trunc((ncyc_max - l21 + 1)*2.0 + 0.3))"
  }
  else {
    "l25 = (trunc(ncyc_max*2.0 + 0.3))"
  }
  2u

  if "l21 > 0" {
    "tauCPMG0 = (time_T2*0.25)/l21"
#ifdef shp_flg
    "tauCPMG = (time_T2*0.25)/l21 - shp_cp*0.5*cnst6"
#else
    "tauCPMG = (time_T2*0.25)/l21 - pwn_cp*0.75"
#endif
    if "tauCPMG + pwn_cp*0.75 < 120.0u" {
      2u
      print "error: tauCPMG < 120u too short for CPMG !!!"
      goto HaltAcqu
    }

    2u
    "tauCPMG1 = tauCPMG - pwn_cp*2.0/PI"
  }
  else {
    "tauCPMG0 = time_T2*0.25"
    "tauCPMG = time_T2*0.25"
    2u
    "tauCPMG1 = tauCPMG"
  }
  2u

  if "l25 > 0" {
#ifdef shp_flg
    "tauCPMG2 = (time_T2*0.5)/l25 - shp_cp*0.5"
#else
    "tauCPMG2 = (time_T2*0.5)/l25 - pwn_cp"
#endif
  }
  else {
    "tauCPMG2 = 0"
  }
  2u

#ifdef c_dec
  "l23 = (trunc((tauCPMG0 + 1u)/d9))"
  2u
  "l24 = (trunc(l23*2.0 + 0.3))"
  "l27 = (trunc((trunc((time_T2 + 4u)/(d9*4.0)))*4.0 + 2 + 0.3))"
  2u
  "l26 = (trunc(l27*0.5 - l21*l23*2.0 + 0.3))"
#else
  "l23 = 0"
  2u
  "l24 = 0"
  "l27 = 0"
  2u
  "l26 = 0"
#endif
  2u

#ifdef c_dec
  if "l26 > 0" {
    "tauCPMG3 = (time_T2*0.25)/l26 - pwc_ad*0.5"
  }
  else {
    "tauCPMG3 = 0"
  }
```

```
  2u
#endif

#ifdef shp_flg
  if "l21 > 0" {
    "DELTA3 = (ncyc_max - l21) * shp_cp * 2.0 * (1.0 - cnst6) + 2u"
  }
  else {
    "DELTA3 = (ncyc_max - 1) * shp_cp * 2.0 * (1.0 - cnst6) + 2u"
  }
#else
  if "l21 > 0" {
    "DELTA3 = (ncyc_max - l21) * pwn_cp + 2u"
  }
  else {
    "DELTA3 = (ncyc_max - 1) * pwn_cp + 2u"
  }
#endif
  2u

/*********************/
/* 1H Heating period */
/*********************/
#ifdef Hheat
  /* shaped pulse */
  4u
  (pw_sl1:sp14 ph27):f1
  4u
  /* shaped pulse */

  4u pl12:f1
  time_Hheat cw:f1 ph26
  2u do:f1

  /* shaped pulse */
  4u
  (pw_sl:sp13 ph29):f1
  4u
  /* shaped pulse */

  20u UNBLKGRAD

  2u
  p50:gp0
  d16

  4u BLKGRAD
#endif /*Hheat*/

/***********************************************/
/* Option for Messerle purge zgoptn -Dmess_flg */
/***********************************************/
#ifdef mess_flg
  4u pl11:f1
  (dly_pg1 ph26):f1
  2u
  (dly_pg2 ph27):f1
#endif /*mess_flg*/

/*******************************************/
/* Heating compensation for 15N CPMG pulses */
/*******************************************/
#ifdef N_comp
  4u pl33:f3

  if "l25 > 0" {
11  tauCPMG2
    CPMG_15N_0
    tauCPMG2 ipp20
```

```
    lo to 11 times l25
  }
  else {
    time_T2
  }

  20u UNBLKGRAD

  2u
  p50:gp0*0.5
  d16

  4u pl3:f3
  (pwn ph26):f3

  2u
  p50:gp0
  d16

  4u BLKGRAD
#endif /*N_comp*/

#ifdef c_dec
  if "l26 > 0" {
13  tauCPMG3
    AFP_13C
    tauCPMG3
    lo to 13 times l26
  }
  else {
    time_T2*0.5
  }
#endif /*c_dec*/

/************************/
/* Presaturation Period */
/************************/
#ifdef N_comp
  "DELTA1 = d1 - time_T2"
#else
  "DELTA1 = d1"
#endif /*N_comp*/

#ifdef c_dec
  "DELTA2 = DELTA1 - time_T2"
#else
  "DELTA2 = DELTA1"
#endif /*c_dec*/

#ifdef fsat
  4u pl10:f1
  DELTA2 cw:f1 ph26
  2u do:f1
  4u pl1:f1

#ifdef fscuba
  hscuba
  (pwh ph26 pwh*2.0 ph27 pwh ph26):f1
  hscuba
#endif /*fscuba*/

#else
  4u pl1:f1
  DELTA2
#endif /*fsat*/

  20u UNBLKGRAD          /* unblank gradient amp and put lock in hold mode */

#ifdef c_dec
```

```
  if "l26 > 0" {
14  tauCPMG3
    AFP_13C
    tauCPMG3
    lo to 14 times l26
  }
  else {
    time_T2*0.5
  }

  2u
  p50:gp0
  d16
#endif

/*************************************/
/* Start H->N magnetization transfer */
/*************************************/
  /* shaped pulse */
  4u
  (pw_sl1:sp14 ph29):f1
  4u pl1:f1
  /* shaped pulse */

  if "l3 % 2 == 0" {
    (pwh ph26):f1
  }
  else {
    (pwh ph28):f1
  }

  2u
  p51:gp1
  d16

#ifdef N_sel
  "DELTA = taua - 2u - p51 - d16 - pwn_sl*0.475"
  DELTA

  (center (pwh*2.0 ph26):f1 (pwn_sl:sp32 ph26):f3)
#else /*N_sel*/
#ifdef shp_flg
  "DELTA = taua - 2u - p51 - d16 - p30*cnst30*0.5"
  DELTA

  (center (pwh*2.0 ph26):f1 (p30:sp30 ph26):f3)
#else /*shp_flg*/
  "DELTA = taua - 2u - p51 - d16"
  DELTA

  (center (pwh*2.0 ph26):f1 (pwn*2.0 ph26):f3)
#endif /*shp_flg*/
#endif /*N_sel*/

  DELTA pl3:f3

  2u
  p51:gp1
  d16

  (pwh ph27):f1

  2u
  p52:gp2
  d16

#ifdef s3e_flg
  (pwn ph26):f3
  2u
```

```
  2u
  p51:gp1
  d16

#ifdef shp_flg
  "DELTA = taub*0.5 - 2u - 2u - p51 - d16 - p29*cnst29*0.5"
  DELTA

  (center (pwh ph27 pwh*2.0 ph26 pwh ph27):f1 (p29:sp29 ph26):f3)

  "DELTA = taub*0.5 - p29*cnst29*0.5 - 2u - p51 - d16 - pwh*4.0 - 2u"
#else /*shp_flg*/
  "DELTA = taub*0.5 - 2u - 2u - p51 - d16"
  DELTA

  (center (pwh ph27 pwh*2.0 ph26 pwh ph27):f1 (pwn ph27 pwn*2.0 ph26 pwn ph27):f3)

  "DELTA = taub*0.5 - 2u - p51 - d16 - pwh*4.0 - 2u"
#endif /*shp_flg*/

  2u
  p51:gp1
  d16

  DELTA pl3:f3

  (pwh ph27 pwh*2.0 ph26 pwh ph27):f1
  2u

  if "l3 % 2 == 0" {
    (pwn ph8):f3
  }
  else {
    (pwn ph9):f3
  }

  2u
  p52:gp2
  d16
#endif /*s3e_flg*/

  time_eq

/*************************************/
/* Start of first half CPMG period */
/*************************************/
  4u pl33:f3
  (pwn_cp ph10):f3

  if "l21 > 0" {
    if "l23 > 0" {
      "DELTA = ((tauCPMG1-2u)/l23 - pwc_ad)*0.5"
3     DELTA
      AFP_13C
      DELTA
      lo to 3 times l23
    }
    else {
      "DELTA = tauCPMG1-2u"
      DELTA
    }

    2u
    CPMG_15N_F

    if "l22 > 0" {
4     2u ipp21 ipp22 ipp23 ipp24
```

```
      if "l24 > 0" {
        "DELTA = ((tauCPMG*2.0-2u-2u)/l24 - pwc_ad)*0.5"
5       DELTA
        AFP_13C
        DELTA
        lo to 5 times l24
      }
      else {
        "DELTA = tauCPMG*2.0-2u-2u"
        DELTA
      }

      2u
      CPMG_15N_F
      lo to 4 times l22
    }

    2u pl33:f3

    if "l23 > 0" {
      "DELTA = ((tauCPMG1-2u)/l23 - pwc_ad)*0.5"
6     DELTA
      AFP_13C
      DELTA
      lo to 6 times l23
    }
    else {
      "DELTA = tauCPMG1-2u"
      DELTA
    }
  }
  else {
    2u
    CPMG_15N_F

    2u pl33:f3
  }

/*******************************/
/* Modified central P-element */
/*******************************/
  (pwn_cp ph11):f3
  2u
  (pwh ph27 pwh*2.0 ph26 pwh ph27):f1

  2u
  p53:gp3
  d16

#ifdef shp_flg
  "DELTA = taub - pwn_cp*2.0/PI - 2u - pwh*4.0 - 2u - p53 - d16 - p29*cnst29*0.5"
  DELTA

  if "l3 % 2 == 0" {
    (center (pwh ph27 pwh*2.0 ph26 pwh ph27):f1 (p29:sp29 ph14):f3)
  }
  else {
    (center (pwh ph27 pwh*2.0 ph26 pwh ph27):f1 (p29:sp29 ph13):f3)
  }

  "DELTA = taub - p29*cnst29*0.5 - 2u - p53 - d16 - pwn_cp*2.0/PI"
#else /*shp_flg*/
#ifdef reb_flg
  "DELTA = taub - pwn_cp*2.0/PI - 2u - pwh*4.0 - 2u - p53 - d16 - pwn_reb*0.475"
  DELTA

  if "l3 % 2 == 0" {
    (center (pwh ph27 pwh*2.0 ph26 pwh ph27):f1 (pwn_reb:sp34 ph14):f3)
  }
```

```
  else {
    (center (pwh ph27 pwh*2.0 ph26 pwh ph27):f1 (pwn_reb:sp34 ph13):f3)
  }

  "DELTA = taub - pwn_reb*0.475 - 2u - p53 - d16 - pwn_cp*2.0/PI"
#else /*reb_flg*/
  "DELTA = taub - pwn_cp*2.0/PI - 2u - pwh*4.0 - 2u - p53 - d16 - larger(pwh,pwn_cp)*2
.0/PI"
  DELTA

  if "l3 % 2 == 0" {
    (center (pwh ph27 pwh*2.0 ph26 pwh ph27):f1 (pwn_cp*2.0 ph14):f3)
  }
  else {
    (center (pwh ph27 pwh*2.0 ph26 pwh ph27):f1 (pwn_cp*2.0 ph13):f3)
  }

  "DELTA = taub - larger(pwh,pwn_cp)*2.0/PI - 2u - p53 - d16 - pwn_cp*2.0/PI"
#endif /*reb_flg*/
#endif /*shp_flg*/

  2u
  p53:gp3
  d16

  DELTA pl33:f3

  (pwn_cp ph12):f3

/*************************************/
/* Start of second half CPMG period */
/*************************************/
  if "l21 > 0" {
    if "l23 > 0" {
      "DELTA = ((tauCPMG1-2u)/l23 - pwc_ad)*0.5"
7     DELTA
      AFP_13C
      DELTA
      lo to 7 times l23
    }
    else {
      "DELTA = tauCPMG1-2u"
      DELTA
    }

    2u
    CPMG_15N_R

    if "l22 > 0" {
8     2u dpp21 dpp22 dpp23 dpp24

      if "l24 > 0" {
        "DELTA = ((tauCPMG*2.0-2u-2u)/l24 - pwc_ad)*0.5"
9       DELTA
        AFP_13C
        DELTA
        lo to 9 times l24
      }
      else {
        "DELTA = tauCPMG*2.0-2u-2u"
        DELTA
      }

      2u
      CPMG_15N_R
      lo to 8 times l22
    }

    2u pl33:f3
```

```
    if "l23 > 0" {
      "DELTA = ((tauCPMG1-2u)/l23 - pwc_ad)*0.5"
10    DELTA
      AFP_13C
      DELTA
      lo to 10 times l23
    }
    else {
      "DELTA = tauCPMG1-2u"
      DELTA
    }
  }
  else {
    2u
    CPMG_15N_R
    2u pl33:f3
  }

  (pwn_cp ph27):f3
  DELTA3

  time_eq

/*********************************/
/* Calibrate 15N pulse under load */
/*********************************/
#ifdef N_calib
  4u pl35:f3
  (p35 ph26):f3

  2u
  p54:gp4
  d16
#endif /*N_calib*/

/*********************************************/
/* Select TROSY (l3=0) or anti-TROSY (l3=1) */
/*********************************************/
  4u pl3:f3

  if "l3 % 2 == 0" {
    "DELTA = 4u + pw_sl1 + 4u + pwh*4.0 + 4u + pw_sl + 4u"
    DELTA
  }
  else {
    /* shaped pulse */
    4u
    (pw_sl1:sp14 ph29):f1
    4u pl1:f1
    /* shaped pulse */

    (pwh ph27 pwh*2.0 ph26 pwh ph27):f1

    /* shaped pulse */
    4u
    (pw_sl:sp13 ph27):f1
    4u pl1:f1
    /* shaped pulse */
  }

  2u
  p55:gp5
  d16

/****************/
/* 15N labeling */
/****************/
  if "Nphase %2 == 1" {
```

```
    (pwn ph1):f3
  }
  else {
    (pwn ph2):f3
  }

#ifdef c_flg
  if "d0 - pwc_ad*0.5 > 2u" {
    "DELTA = larger(d0 - pwc_ad*0.5, TAU2)"
    DELTA
    (pwc_ad:sp22 ph26):f2
    DELTA
  }
  else {
    d0
    d0
  }
#else
  d0
  d0
#endif /*c_flg*/

/*********************************************/
/*   Encoding coherence selection gradient   */
/*********************************************/
#ifdef gd_sel
  2u
  p56:gp6*EA*-1.0
  d16

  "DELTA = BigT - 2u - p56 - d16"
  DELTA

#ifdef shp_flg
  (p28:sp28 ph26):f3
#else
  (pwn*2.0 ph26):f3
#endif

  2u
  p56:gp6*EA
  d16

  "DELTA = BigT - 2u - p56 - d16 + pwn*4.0/PI + 2u"
#else
  "DELTA = de"
#endif
  DELTA

/*************************************/
/* N->H back transfer through ST2-PT */
/*************************************/
  (pwh ph3):f1

  /* shaped pulse */
  4u
  (pw_sl:sp13 ph4):f1
  4u pl1:f1
  /* end shaped pulse */

  2u
  p57:gp7
  d16

#ifdef shp_flg
  "DELTA = taua - 4u - pw_sl - 4u - 2u - p57 - d16 - p29*cnst29*0.5"
  DELTA

  (center (pwh*2.0 ph26):f1 (p29:sp29 ph26):f3)
```

```
#else /*shp_flg*/
  "DELTA = taua - 4u - pw_sl - 4u - 2u - p57 - d16"
  DELTA

  (center (pwh*2.0 ph26):f1 (pwn*2.0 ph26):f3)
#endif /*shp_flg*/

  2u
  p57:gp7
  d16

  DELTA pl3:f3

  /* shaped pulse */
  4u
  (pw_sl:sp13 ph26):f1
  4u pl1:f1
  /* end shaped pulse */

  (pwh ph26):f1

#ifdef gd_sel
  2u
#else
  "DELTA = de"
  DELTA
#endif

  (pwn ph5):f3

  2u
  p58:gp8
  d16

#ifdef gd_sel
#ifdef N_sel
  "DELTA = taua - 2u - p58 - d16 - pwn_sl*0.475"
  DELTA

  (center (pwh*2.0 ph26):f1 (pwn_sl:sp32 ph26):f3)
#else /*N_sel*/
#ifdef shp_flg
  "DELTA = taua - 2u - p58 - d16 - p29*cnst29*0.5"
  DELTA

  (center (pwh*2.0 ph26):f1 (p29:sp29 ph26):f3)
#else /*shp_flg*/
  "DELTA = taua - 2u - p58 - d16"
  DELTA

  (center (pwh*2.0 ph26):f1 (pwn*2.0 ph26):f3)
#endif /*shp_flg*/
#endif /*N_sel*/
#else /*gd_sel*/
#ifdef shp_flg
  "DELTA = taua - 2u - p58 - d16 - 4u - pw_sl1 - 4u - p29*cnst29*0.5"
  DELTA

  /* shaped pulse */
  4u
  (pw_sl1:sp14 ph6:r):f1
  4u pl1:f1
  /* end shaped pulse */

  (center (pwh*2.0 ph26):f1 (p29:sp29 ph26):f3)

  /* shaped pulse */
  4u
  (pw_sl:sp13 ph7:r):f1
```

```
  4u pl1:f1
  /* end shaped pulse */

  "DELTA = taua - p29*cnst29*0.5 - 4u - pw_sl - 4u - 2u - p58 - d16 - 4u"
#else /*shp_flg*/
  "DELTA = taua - 2u - p58 - d16 - 4u - pw_sl1 - 4u"
  DELTA

  /* shaped pulse */
  4u
  (pw_sl1:sp14 ph6:r):f1
  4u pl1:f1
  /* end shaped pulse */

  (center (pwh*2.0 ph26):f1 (pwn*2.0 ph26):f3)

  /* shaped pulse */
  4u
  (pw_sl:sp13 ph7:r):f1
  4u pl1:f1
  /* end shaped pulse */

  "DELTA = taua - 4u - pw_sl - 4u - 2u - p58 - d16 - 4u"
#endif /*shp_flg*/
#endif /*gd_sel*/

  2u
  p58:gp8
  d16

  DELTA pl3:f3

#ifdef gd_sel
  (pwn ph28):f3

  "DELTA = BigT1 + pwh*2.0/PI + 2u"
  DELTA

  (pwh*2.0 ph26):f1

  2u
  p59:gp9*-1.0
  d16

  "DELTA = BigT1 - 2u - p59 - d16 - 4u - de"
  DELTA

  4u BLKGRAD
#else
  4u BLKGRAD

  (pwn ph28):f3

  "DELTA = pwh*2.0/PI"
  DELTA
#endif /*gd_sel*/

/*******************************************/
/*   Test how much water left              */
/*   set zgoptn -Dwater and                */
/*   use small value of rg                 */
/*   set <10 deg flip angle for tap pulse  */
/*******************************************/
#ifdef water
  if "nsdone == 0" {
    DELTA11

    20u UNBLKGRAD
```

```
    2u
    p50:gp0
    d16

    4u BLKGRAD

    DELTA12
    (pwh_tap ph26):f1
  }
#endif /*water*/

/*************************************/
/*   Signal detection and looping      */
/*************************************/
  go=2 ph31
  d11 mc #0 to 2
    F3QF(calclc(l3, 1))
    F2QF(calclc(l2, 1))
    F1EA(calgrad(EA) & calph(ph3, +180) & calph(ph4, +180) & calph(ph5, +180) & calclc
(Nphase, 1), caldel(d0, +in0) & calph(ph1, +180) & calph(ph2, +180) & calph(ph31, +180
))

HaltAcqu, 1m
exit

ph1=1 1 2 2 3 3 0 0
ph2=1 1 0 0 3 3 2 2
ph3=1
ph4=3
ph5=3
ph6=2
ph7=2
ph8=(8) 1
ph9=(8) 7
ph10=0 2
ph11=1 3 3 1
ph12=0 2 2 0
ph13={{1}*8}^2
ph14={{0}*8}^2
ph20=0 1 0 1
ph21=1 1 0 2
ph22=0 2 3 3
ph23=0 0 1 3
ph24=1 3 2 2
ph26=0
ph27=1
ph28=2
ph29=3
ph31=3 1 2 0 1 3 0 2

;d1: repetition delay d1
;d3: taua < 1/4J(NH) 2.38ms
;d5: taub = 1/4J(NH) 2.68ms
;d9: cutoff delay for tau_cp, above which 13C AFPs are applied. Suggest 5ms
;d11: delay for disk i/o, 30ms
;d13: delay time prior to tap reading pulse
;d14: BigT1 set to 500us
;d15: BigT set to 850us
;d16: gradient recovery delay 200us
;d17: time_T2 typically between 20 - 40 ms Presently has a cap at 100 ms
;d18: time_Hheat used in case 1H heating is applied
;d19: time_eq typically about 2-5ms
;pl1: tpwr - power level for 1H pulses
;pl3: dhpwr2 - power level for N hard pulses
;pl10: tsatpwr - power level for water presat
;pl12: power level for 1H heat to achieve same level of heating as in 15N CW CPMG
;pl33: power level for pwn_cp
;pl35: power level to calibrate 15N pulses under load. Set to 1000 dB when acquiring r
eal data
```

```
;sp13: tpwrsl - power level for water selective pulse pw_sl
;sp14: tpwrsl1 - power level for water selective pulse pw_sl1
;sp22: power level for 13C adiabatic pulse
;sp32: power level for 15N selective pulse
;spnam13: shape for pw_sl
;spnam14: shape for pw_sl1
;spnam22: shape for pwc_ad
;spnam28: shape for 15N broadband refocusing pulse applied alone
;spnam29: shape for 15N broadband refocusing pulse during INEPT
;spnam30: shape for 15N broadband inversion pulse during INEPT
;spnam32: shape for 15N selective pulse to remove Arg
;spnam34: Reburp.1000
;p1: pwh - 1H 90 degree pulse
;p3: pwn - 15N 90 degree pulse
;p13: pw_sl
;p14: pw_sl1
;p22: pwc_ad -  adiabatic C180 pulse width
;p28: pulse width for 15N broadband refocusing pulse applied alone
;p29: pulse width for 15N broadband refocusing pulse during INEPT
;p30: pulse width for 15N broadband inversion pulse during INEPT
;p32: pwn_sl for selective pulse on amide N15s
;p33: pwn_cp 90 15N pulse for CPMG Typically 40 ms
;p35: pulse used to calibrate under load SET TO 0 when start real expt
;p36: pulse duration for CPMG shaped pulse with maximum power level of 10kHz
;p50: gradient pulse 50                                  [1000 usec]
;p51: gradient pulse 51                                  [256 usec]
;p52: gradient pulse 52                                  [1000 usec]
;p53: gradient pulse 53                                  [300 usec]
;p54: gradient pulse 54                                  [1000 usec]
;p55: gradient pulse 55                                  [1000 usec]
;p56: gradient pulse 56                                  [625 usec]
;p57: gradient pulse 57                                  [256 usec]
;p58: gradient pulse 58                                  [256 usec]
;p59: gradient pulse 59                                  [256 usec]
;phcor6: phase correction for ph6 of pw_sl1
;phcor7: phase correction for ph7 of pw_sl
;cnst5: flip angle of tap reading pulse for testing water preservation
;cnst6: fraction of time magnetization on transverse plane during CPMG pulse
;cnst29: fraction of J-coupling active time during sp29
;cnst30: fraction of J-coupling active time during sp30
;l3: 0 for TROSY, 1 for anti-TROSY
;l8: ncyc_max (MUST BE SET PROPERLY!)
;vclist: variable counter list for ncyc_cp
;inf1: 1/SW(X) = 2*DW(X)
;in0: 1/(2*SW(x))=DW(X)
;nd0: 2
;ns: 4*n or 2*n (with -Dgd_sel)
;FnMODE: Echo-Antiecho in F1
;FnMODE: QF in F2
;FnMODE: QF in F3

;use gradient ratio:    gp 6 : gp 9
;                         80 : 39.6

;for z-only gradients:
;gpz0: 15%
;gpz1: 25%
;gpz2: -70%
;gpz3: 30%
;gpz4: 70%
;gpy5: 0% (Z-gradient) or 90% (XYZ-gradient)
;gpz5: 90%
;gpx6: 0% (Z-gradient) or 80% (XYZ-gradient)
;gpz6: 80% (Z-gradient) or 0% (XYZ-gradient)
;gpz7: 60%
;gpz8: 15%
;gpx9: 0% (Z-gradient) or 39.6% (XYZ-gradient), need to optimize
;gpz9: 39.6% (Z-gradient) or 0% (XYZ-gradient), need to optimize
```

```
;use gradient files:
;gpnam0: SMSQ10.32
;gpnam1: SMSQ10.32
;gpnam2: SMSQ10.32
;gpnam3: SMSQ10.32
;gpnam4: SMSQ10.32
;gpnam5: SMSQ10.32
;gpnam6: SMSQ10.32
;gpnam7: SMSQ10.32
;gpnam8: SMSQ10.32
;gpnam9: SMSQ10.32

;zgoptns: Df1180, Dc_flg, Dfsat, Dfscuba, Dmess_flg, Dgd_sel, DN_sel, DHheat, DN_comp,
 Dreb_flg, Ds3e_flg, Dwater, Dshp_flg, Dc_dec
```