ANODE_LOOK_ELC_DEFL_EQNS_V02

This document provides equations to calculate the High Rate Science (HRS) elevations angles for JADE-E when the deflectors are utilized : version 02. This is a somewhat simplified version of what the onboard software does. Later versions may improve this to be closer to the onboard software.

The first parts are descriptions of the process, the last part is the IDL code used to represent the process (as used in the JADE product code for this version), albeit it with some cosmetic differences to look better on a page and not wrap lines, e.g. `in_file.MAG_VECTOR[*,0]^2` instead of `in_file.MAG_VECTOR[*,0]*in_file.MAG_VECTOR[*,0]`).

The electron deflection equations only affect elevation angles of the field-of-view (FOV). For azimuthal angles (with or without deflection) see the relevant version of file ANODE_LOOK_ELC_DEFL_NONE_V*nn*. To know which version (for either elevation or azimuth) was used for a given Level 3 JADE record, look at the SOURCE_JADE_CALIB object which is a version number, *mmmmm*, which corresponds to the file JAD_L30_CALIB_LIST_*mmmmm*.TXT. This file then says which versions of other files (such as file ANODE_LOOK_ELC_DEFL_NONE_V*nn* or ANODE_LOOK_ELC_DEFL_EQNS_V*nn*) were used.

Note: Electron deflectors for HRS were not used for science data during Cruise, nor JOI, their first science use was at Perijove 1 (although they had previously been used during operations check-out tests, but those are not for science use). Any HRS before that time (or if the JADE Level 2 data record has MAG_VECTOR = [0, 0, 0]) did not deflect, and the non-deflection elevation angles should be used, which are found in the relevant version of file ANODE_LOOK_ELC_DEFL_NONE_V*nn* (see JAD_L30_CALIB_LIST_*mmmmm*.TXT for which *nn*).

Methodology summary

There are two different methods to calculate the deflection, named in this document as *simple* and *complex*. The simple method is used in preference, and has an energy-independent deflection angle . If the JADE deflectors (either the low gain or high gain part) reach their maximum voltage (DAC = 4095), then the complex method is also used, giving an energy-dependent deflection angle. In the IDL code the complex method is flagged to only return deflections when DAC = 4095, and otherwise return fill values (=65535 degs), which makes the merging of the simple and complex results easier: use the simple method results, and over-write elements with the complex angles where the complex angle is not 65535 (see IDL function "**`_hrs_electron_deflection_merge`**").

Complex Deflection Description

The following description is mostly a *cut & paste* from internal JADE document "from_B_to_elevation_20160914.doc". The "Broadcast magnetic field" is in spacecraft co-ordinates, and is uncalibrated. It does not list the FWHM of Elevation angle for a given deflection, see the 'FWHM of Elevation Angle' section of this PDF.

Calculated deflection angle is **dependent** on JADE energy step (or eV/q)!
IDL code for this is function "`_hrs_electron_deflection`".

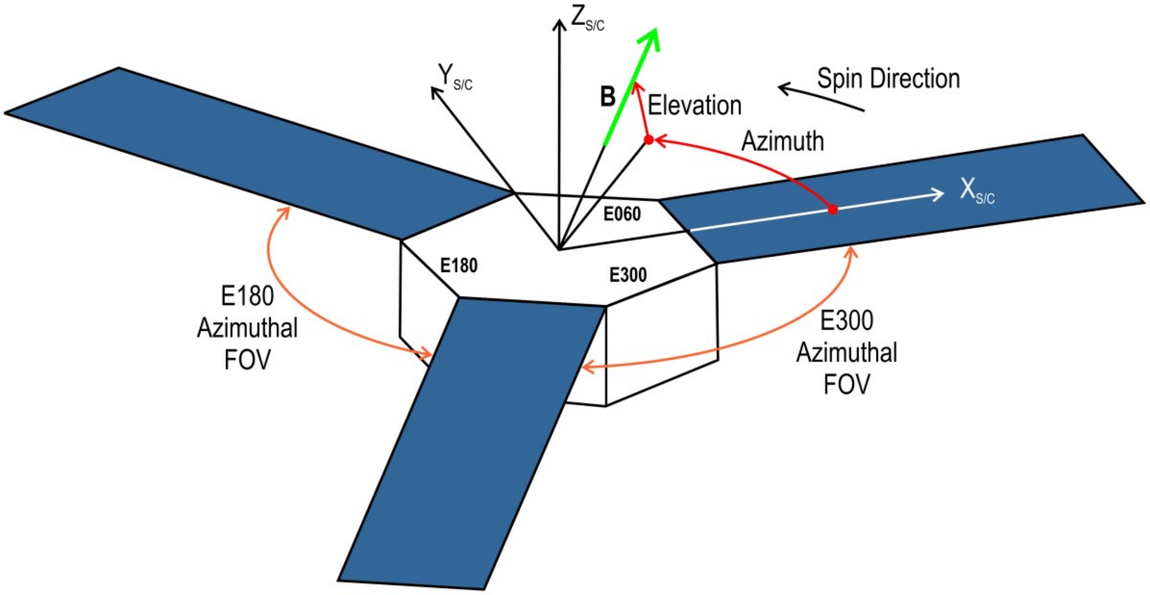
JADE-E elevation of look direction in HRS

Frédéric Allegrini, Sept. 1, 2016

Rev. 1, Sept. 14, 2016: corrected algorithm in step 0

At the beginning of every energy sweep, JADE uses the propagated broadcast magnetic field to determine the elevation of the look direction and which sensor uses its deflectors:

Broadcast magnetic field: B = (Bx, By, Bz) (1)



Step 0: determine the sign of the deflection angle (positive or negative) for each sensor.

At the beginning of the sweep, the propagated broadcast magnetic field falls onto an anode (called the "mag_anode"): if J is the structure containing the HRS data, then

mag_anode = J.MAG_LOOK_DIR (2)

Step 1: calculate elevation, δ, in degrees of B in S/C coordinates

$$\delta_B = \arctan\left(\frac{B_z}{\sqrt{B_x^2 + B_y^2}}\right) \quad (3)$$

$|\delta_B|$ is capped at 35°.
If $\delta_B > +35°$, then $\delta_B = +35°$
If $\delta_B < -35°$, then $\delta_B = -35°$

Step2: calculate voltage on deflector using

To determine which sensor is using the deflectors (UP or DN) we use the following table:

	E060	
	$\delta_B > 0$	$\delta_B < 0$
0≤mag_anode≤19	UP	DN
20≤ mag_anode≤43	DN	UP
44≤mag_anode≤47	UP	DN

	E180	
	$\delta_B > 0$	$\delta_B < 0$
0≤mag_anode≤11	DN	UP
12≤ mag_anode≤35	UP	DN
36≤mag_anode≤47	DN	UP

Reverse model: 5 parameters
Deflection voltage in kV:

$$V = E \cdot \left(b_0 + b_1\delta_B + b_2\delta_B^2 + b_3\delta_B^3\right) \quad (4)$$

V<0 if DFL DN is powered
V>0 if DFL UP is powered
V in kV and *E* in keV
E is from energy table

$$b_0 = 7.16 \cdot 10^{-4} + \gamma_0$$

$$b_1 = b_{10} + b_{11}E$$

$$b_2 = 5.85 \cdot 10^{-7} + \gamma_2$$

$$b_3 = -8.51 \cdot 10^{-8} + \gamma_3$$

Model	γ_0	b_{10}	b_{11}	γ_2	γ_3
E060 (FM2)	0.00158	6.425E-3	-5.73E-6	7.62E-7	-2.46E-8
E180 (FM3)	0.00237	6.448E-3	-5.87E-6	7.40E-7	6.15E-9

Step 3: convert Volts to DAC, round value, then convert back to Volts

V in volts is converted to a DAC value (Use absolute value of V):
DAC = ROUND((|V|−Offset) /Scale) (5)

Offset and scale in table below.
DAC is an integer between 0 and 4095.
If DAC > 4095, then DAC = 4095

V	**Supply**	**Scale**	**Offset**
≥300 V	E060_DFL_UP_HG	2.4308	42.8220
	E060_DFL_DN_HG	2.4513	42.7790
	E180_DFL_UP_HG	2.4431	43.4540
	E180_DFL_DN_HG	2.4415	48.2290
<300 V	E060_DFL_UP_LG	0.0728	1.2663
	E060_DFL_DN_LG	0.0727	1.3000
	E180_DFL_UP_LG	0.0728	1.4000
	E180_DFL_DN_LG	0.0741	1.6000

Convert rounded DAC back to voltage:

V (in volts) = DAC * Scale + Offset (6)
Change sign of V to be the same as in Eq. (4)

Step 4: calculate actual deflection using

Forward model: 5 parameters
Deflection angle in degrees (V in kV and E in keV):

$$\delta = a_0 + a_1\left(\frac{V}{E}\right) + a_2\left(\frac{V}{E}\right)^2 + a_3\left(\frac{V}{E}\right)^3 \qquad (7)$$

$$a_0 = -0.11 + \varepsilon_0 \qquad a_2 = -2.457 + \varepsilon_2$$

$$a_1 = 154.85 + 1.582 \cdot 10^{-4} E + \varepsilon_1 \qquad a_3 = 52.02 + \varepsilon_3$$

Model	ε_0 [º]	ε_1	ε_2	ε_3
E060 (FM2)	-0.24	0.71	-3.646	18.00
E180 (FM3)	-0.37	0.20	-3.299	-3.342

δ is the elevation for this time period, this energy, and all 16 anodes for this sensor

Step 5: apply correction due to offset between commanded and actual deflector voltage. This will be done in the form of a matrix or equations. Ignore for now.

The FWHM of Elevation angle is explained in a separate section of this PDF.

Simple Deflection Description

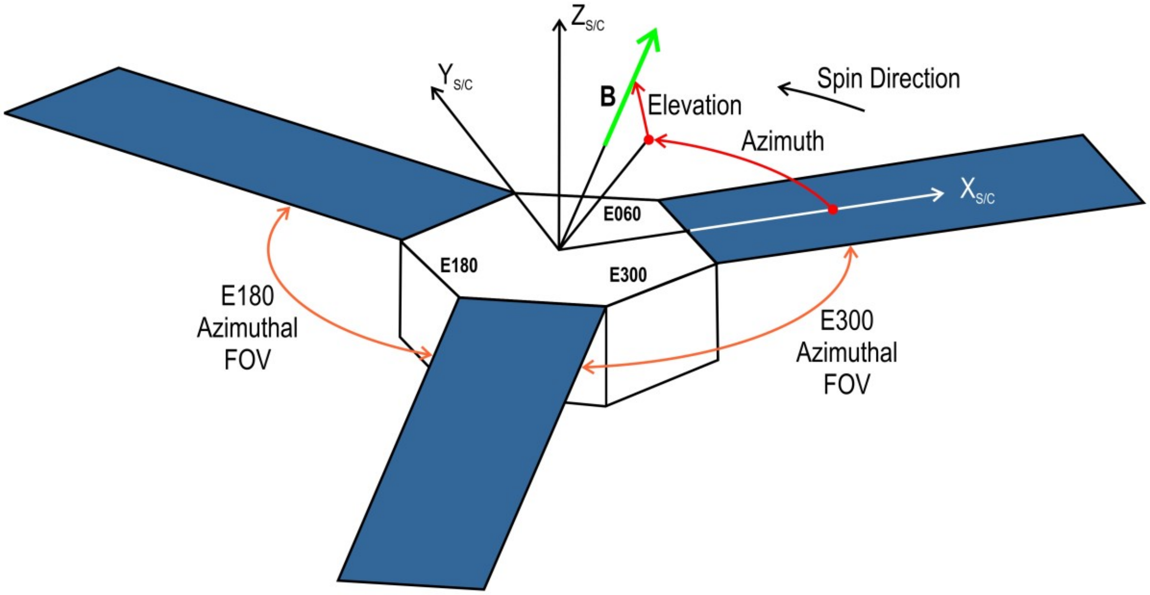
The following description is mostly a *cut & paste* from internal JADE document "from_B_to_elevation_20161020.doc". The "Broadcast magnetic field" is in spacecraft co-ordinates, and is uncalibrated. It does not list the FWHM of Elevation angle for a given deflection, see the 'FWHM of Elevation Angle' section of this PDF.

Calculated deflection angle is **independent** of JADE energy step (or eV/q)!
IDL code for this is function "`_hrs_electron_deflection_simple`".

JADE-E elevation of look direction in HRS
Frédéric Allegrini, Sept. 1, 2016

At the beginning of every energy sweep, JADE uses the propagated broadcast magnetic field to determine the elevation of the look direction and which sensor uses its deflectors:

Broadcast magnetic field: B = (Bx, By, Bz) (1)



Step 0: determine the sign of the deflection angle (positive or negative) for each sensor.

At the beginning of the sweep (every second), the propagated broadcast magnetic field falls onto an anode (called the "mag_anode", integer between 0 and 47): if J is the structure containing the HRS data, then

mag_anode = J.MAG_LOOK_DIR (2)

Step 1: calculate elevation, δ, in degrees of B in S/C coordinates

$$\delta_B = \arctan\left(\frac{B_z}{\sqrt{B_x^2 + B_y^2}}\right) \tag{3}$$

|δ_B| is capped at 35°.
If δ_B > +35°, then δ_B =+35°
If δ_B < -35°, then δ_B =-35°

To determine the targeted elevation (i.e., the angle that JADE-E is deflecting to), δ_D, for this second we use the following:

- For E060: if 20≤ mag_anode≤43, then δ_D = -δ_B, else δ_D = δ_B
- For E180: if 12≤ mag_anode≤35, then δ_D = δ_B, else δ_D = -δ_B

δ_D is the angle to use for now in the conversion of L2 to L3 JADE-E data. It is the same for all anodes and energies for a given second.

This shortened elevation calculation does not account for offsets from the different conversions within the instrument, errors due to HVPS not reaching targeted voltages, and magnetic field effects on the trajectories.

The steps from the previous version of this document (which took some of the offsets from conversions into account) and refinements to the method will be implemented later as needed.

The FWHM of Elevation angle is explained on the next page.

The FWHM of Elevation Angle

The internal JADE document "JADE-E_calibration_report_v1.docx" is based on ground calibration data and electro-optics simulations; relevant parts are cut & pasted in this section, and apply to both the Complex and Simple sections.

Equation 15 of "JADE-E_calibration_report_v1.docx" provides the elevation (*el*) resolution equations as shown below (units of degrees), and their Figure 24 (used as the source of the equations) is pasted below that:

$$el\ FWHM = \begin{cases} 29.9 + 0.758el & -35^\circ \leq el < -32.2^\circ \\ 3.55 - 0.0606el & -32.2^\circ \leq el < 23.4^\circ \\ -0.189 + 0.0993el & 23.4^\circ \leq el \leq 35^\circ \end{cases}$$

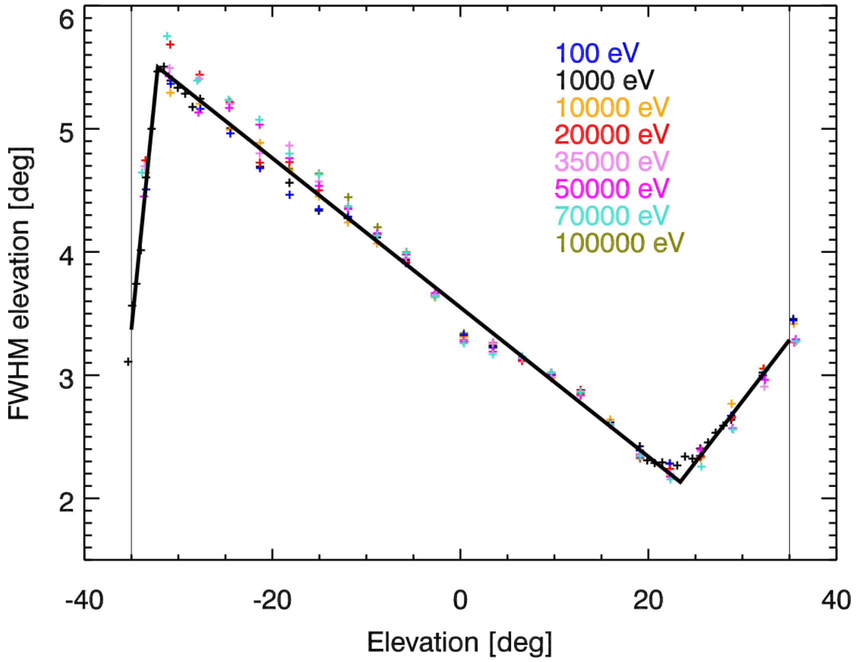


Figure 24. FWHM of the elevation angle distribution as a function of elevation angle from simulations at energies from 102 to 105 eV. The thick lines are fits using Eq. 15.

These equations are used in IDL for the *`DIM2_ELEVATION_UPPER`* and *`DIM2_ELEVATION_UPPER`* objects of JADE Level 3 files.

IDL code used to populate JADE Level 3 files

The code below was used to calculation the center elevation angles (DIM2_ELEVATION) only if there was deflection.

In_file is a structure of the objects, which includes fields of:

- MAG_VECTOR:
  A vector (size 3) of the magnetic field in spacecraft co-ordinates (JADE despun spacecraft or regular spacecraft, does not matter, as identical +z axis)
- MAG_LOOK_DIR:
  Mag_look_dir = MAG Look direction (0 to 47) from JADE-E HRS Level 2 files

Specific IDL coding methods:

- IDL uses $ to continue on next line.
- Doubles are denoted with a d, e.g. 35d = double(35).
- Signed long integers (4-bytes) are denoted with an L (e.g. -1L = long(-1)).
- Numbers without a d or an L are signed short integers (2-bytes).
- EQ = equal, NE = not equal.
- GT = greater than, GE = greater than or equal.
- LT = less than, LE = less than or equal.

The **_hrs_electron_deflection_simple** code is basically the same as from ANODE_LOOK_ELC_DEFL_EQNS_V01, with a few minor cosmetic changes, which include: Mag_look_dir is now L2.MAG_LOOK_DIR, and the MAG_VECTOR fill value is 9990000.0 and not 2147483646 (long integer) as in the V01 code, not that we had any MAG_VECTOR data of either fill value in data at the time.

Version 02 code starts here:

```
; Now do HRS Electron Deflection... if HRS Electrons
IF (ISELCHRS EQ 1) THEN BEGIN
  ; Now add Elevation_delta on to no-deflection elevation - but only if not fill.
  ; If HRS_ALL data was used, the E300 ones will be 65535 so not altered.

  DO_DAC4095_Only = 1
  Elevation_delta_high = _hrs_electron_deflection( L3, L2.MAG_LOOK_DIR, DO_DAC4095_Only)
  ; returns -1 if too early for deflection, i.e. cruise
  Elevation_delta_low  = _hrs_electron_deflection_simple(L3, L2.MAG_LOOK_DIR)
  ; returns -1 if too early for deflection, i.e. cruise
  Elevation_delta = _hrs_electron_deflection_merge(L3.MAG_VECTOR, Elevation_delta_high,
Elevation_delta_low, DO_DAC4095_Only)


  IF ((Elevation_delta[0] EQ -1) AND (N_ELEMENTS(Elevation_delta) EQ 1)) EQ 0 THEN BEGIN
    ind = WHERE(Elevation_delta LT 65534d,/NULL)
    ; using 65534 to avoid rounding issues, expect values < 35, so very safe
    IF N_ELEMENTS(ind) GT 0 THEN BEGIN
      ; First do center values:
      L3.DIM2_ELEVATION[ind] = Elevation_delta[ind]

      ; Work out Delta's, first assume all is in the middle range, then do the two end
ranges, if any.
      Elevation_delta_FWHM = 3.55d - 0.0606d * Elevation_delta
      ind1 = WHERE( Elevation_delta LT -32.2d,/NULL)
      IF (N_ELEMENTS(ind1) GT 0) THEN $
          Elevation_delta_FWHM[ind1] =   29.9d + 0.758d * Elevation_delta[ind1]
      ind1 = WHERE( (Elevation_delta GE 23.4d) AND (Elevation_delta LT 65534d), /NULL)
                                                   ; need to ignore fill values
      IF (N_ELEMENTS(ind1) GT 0) THEN $
          Elevation_delta_FWHM[ind1] = -0.189d +0.0993d * Elevation_delta[ind1]
      ; Divide by 2 to get half FWHM for upper and lower
      Elevation_delta_FWHM = Elevation_delta_FWHM / 2d
      ; base the delta's off the new center energy
      L3.DIM2_ELEVATION_UPPER[ind] = L3.DIM2_ELEVATION[ind] + Elevation_delta_FWHM[ind]
      L3.DIM2_ELEVATION_LOWER[ind] = L3.DIM2_ELEVATION[ind] - Elevation_delta_FWHM[ind]
    ENDIF
  ENDIF
ENDIF
```

The sub-functions are on the following pages.

_hrs_electron_deflection_merge

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
FUNCTION _hrs_electron_deflection_merge, L3_MAG_VECTOR, Elevation_delta_high,
Elevation_delta_low, DO_DAC4095_Only

  COMPILE_OPT HIDDEN
  ON_ERROR,2

  ; Check that neither are -1 (meaning empty structure_
  IF (Elevation_delta_high[0] EQ -1) AND (N_ELEMENTS(Elevation_delta_high) EQ 1) AND $
     (Elevation_delta_low[ 0] EQ -1) AND (N_ELEMENTS(Elevation_delta_low ) EQ 1)     $
     THEN BEGIN
    RETURN,Elevation_delta_high  ; If both are -1, return -1
  ENDIF

  ; Check if just one is -1, if so return the other.
  IF (Elevation_delta_high[0] EQ -1) AND (N_ELEMENTS(Elevation_delta_high) EQ 1) THEN $
     RETURN, Elevation_delta_low
  IF (Elevation_delta_low[ 0] EQ -1) AND (N_ELEMENTS(Elevation_delta_low ) EQ 1) THEN $
     RETURN, Elevation_delta_high

  IF DO_DAC4095_Only EQ 1 THEN BEGIN
    ;Elevation_delta_high should be all fill values unless DAC = 4095, in which case
replace just those.
    ind = WHERE(Elevation_delta_high LT 65534d, n_ind, /NULL )
    ; 65535 is the fill value, using 65534 to avoid any rounding issues
    IF n_ind EQ 0 THEN RETURN, Elevation_delta_low ; all of Elevation_delta_high was fill
values

    Elevation_delta      = Elevation_delta_low       ; set all to the low
    Elevation_delta[ind] = Elevation_delta_high[ind] ; overwrite the highs where DAC = 4095

  ENDIF ELSE BEGIN
    ; DO FULL MERGE BASED ON DELTAB LIMIT IN WHERE STATEMENT BELOW
    rads2degs = 180d/!DPI
    deltaB = rads2degs * ATAN(L3_MAG_VECTOR[*,2] / $
             SQRT(L3_MAG_VECTOR[*,0]*L3_MAG_VECTOR[*,0] + $
                  L3_MAG_VECTOR[*,1]*L3_MAG_VECTOR[*,1] ) ) ; in degrees
    abs_deltaB = abs(deltaB)

    ind_LE = WHERE(abs_deltaB LE 10d, n_ind_LE, $
                   COMPLEMENT=ind_GT, NCOMPLEMENT=n_ind_GT, /NULL )
    IF n_ind_LE EQ 0 THEN RETURN,Elevation_delta_high
    IF n_ind_GT EQ 0 THEN RETURN,Elevation_delta_low

    ; if here, we must merge the two.
    Elevation_delta             = Elevation_delta_low              ; set all to the low
    Elevation_delta[ind_GT,*,*] = Elevation_delta_high[ind_GT,*,*] ; overwrite the highs
  ENDELSE

  RETURN, Elevation_delta
END
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

_hrs_electron_deflection_simple (‘simple’)

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
FUNCTION _hrs_electron_deflection_simple, in_file, L2_mag_look_dir

  COMPILE_OPT HIDDEN
  ON_ERROR,2

  IF in_file.PACKETID[0] NE 142 THEN $
    MESSAGE, $
     'ERROR: _hrs_electron_deflection should only be applied to HRS Electron datasets'
  IF TOTAL(STRCMP(TAG_NAMES(in_file),'DIM2_ELEVATION')) EQ 0 THEN $
    MESSAGE, 'Cannot run this in Level 2 data, needs to be (nearly) level 3)'
    ; needs in_file.DIM2_ELEVATION
  IF in_file.TIMESTAMP_WHOLE[0] LT 520862570 THEN BEGIN
    ; IF DATE BEFORE JOI THEN RETURN, DEFLECTION WAS NEVER ON
    ; 520862570 = 2016-07-04, JOI
    ; First HRS electron data with deflectors was 2016-240 (2016-Aug-27)
    PRINT,'Time stamp before JOI, was no HRS deflection to be done.  Returning as is.'
    RETURN,-1
  ENDIF
  ; IF AFTER JOI, ONLY E060 and E180 where on, never E300

  IF (ABS(MAX(in_file.DIM2_ELEVATION)) GT 0.5d) THEN BEGIN
    PRINT,'-> WARNING: LOOKS LIKE YOU ALREADY CORRECTED THE ELEVATIONS, RETURNING AS IS!'
    RETURN,in_file.DIM2_ELEVATION
  ENDIF

  ; pre-allocate array of fill values for elevation delta
  Elevation_delta    = in_file.DIM2_ELEVATION ; get right size, be it 32 or 48 anodes
  Elevation_delta[*] = 65535d ; make all fill

  ; Step 0  ; mag_anode must be 0-47 only - enforced before generation of Level 2 files!
  mag_anode = ROUND( L2_mag_look_dir )
  ; Rounding just to make integer and avoid rounding errors.

  ; Step 1, find elevation in degrees and cap at +/- 35 degs
  rads2degs = 180d/!DPI
  deltaB = rads2degs * ATAN(in_file.MAG_VECTOR[*,2] / $
           SQRT(in_file.MAG_VECTOR[*,0]*in_file.MAG_VECTOR[*,0] + $
                in_file.MAG_VECTOR[*,1]*in_file.MAG_VECTOR[*,1] ) ) ; in degrees
  deltaB( WHERE(deltaB GT  35d ,/NULL) ) =  35d ; upper cap
  deltaB( WHERE(deltaB LT -35d ,/NULL) ) = -35d ; lower cap

  ; Step 2 - assign elevation based on anode and mag_look_dir
  ; Do For loop through array as MAG vector can change
  FOR rec = 0L,(N_ELEMENTS(in_file.T) - 1L)  DO BEGIN
    ; ignore if MAG vector is all zeros or fills
    IF ((in_file.MAG_VECTOR[rec,0] EQ 0L) AND (in_file.MAG_VECTOR[rec,1] EQ 0L) AND $
        (in_file.MAG_VECTOR[rec,2] EQ 0L)) THEN $
        CONTINUE ; if no MAG vector (all zeros), skip (leave Elevation_delta as 65535)

    IF ((in_file.MAG_VECTOR[rec,0] EQ 9990000.0) AND (in_file.MAG_VECTOR[rec,1] EQ $
         9990000.0  ) AND (in_file.MAG_VECTOR[rec,2] EQ 9990000.0  )) THEN $
         CONTINUE ; or if MAG vector contains any fills Level 3; 9990000.0

    ; Now do elevation, a line each for E060 and E180
    IF ((mag_anode[rec] GE 20) AND (mag_anode[rec] LE 43)) THEN $
      Elevation_delta[rec,*, 0:15] = -deltaB[rec] ELSE $
      Elevation_delta[rec,*, 0:15] =  deltaB[rec] ; for E060
    IF ((mag_anode[rec] GE 12) AND (mag_anode[rec] LE 35)) THEN $
      Elevation_delta[rec,*,16:31] =  deltaB[rec] ELSE $
      Elevation_delta[rec,*,16:31] = -deltaB[rec] ; for E180
    ; nothing to do for E300, it's all fills.
  ENDFOR

  RETURN, Elevation_delta
END
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

_hrs_electron_deflection ('complex')

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
FUNCTION _hrs_electron_deflection, in_file, L2_mag_look_dir, DO_DAC4095_Only

  COMPILE_OPT HIDDEN
  ON_ERROR,2

  IF in_file.PACKETID[0] NE 142 THEN MESSAGE,'ERROR: _hrs_electron_deflection should only
be applied to HRS Electron datasets'
  IF TOTAL(STRCMP(TAG_NAMES(in_file),'DIM2_ELEVATION')) EQ 0 THEN $
    MESSAGE,'Cannot run this in Level 2 data, needs to be (nearly) level 3)'
  IF in_file.TIMESTAMP_WHOLE[0] LT 520862570 THEN BEGIN
    ; IF DATE BEFORE JOI THEN RETURN, DEFLECTION WAS NEVER ON
    ; 520862570 = 2016-07-04, JOI
    ; First HRS electron data with deflectors was 2016-240 (2016-Aug-27)
    PRINT,'Time stamp before JOI, was no HRS deflection to be done.  Returning as is.'
    RETURN,-1
  ENDIF
  ; IF AFTER JOI, ONLY E060 and E180 where on, never E300

  IF (ABS(MAX(in_file.DIM2_ELEVATION)) GT 0.5d) THEN BEGIN
    PRINT,'-> WARNING: LOOKS LIKE YOU ALREADY CORRECTED THE ELEVATIONS, RETURNING AS IS!'
    RETURN,in_file.DIM2_ELEVATION
  ENDIF

  ; This constants are size 3 for E060, E180 and E300 respectively.
  ; Faster if I remove E300 values as never used?
  ; Made size 2, removing E300 so that I get an error if we ever have a situation where
E300 is on.
  g0 = [ 0.00158d, 0.00237d];, 0.00145d]
  g2 = [ 7.62d-7 , 7.40d-7 ];, 1.29d-6 ]
  g3 = [-2.46d-8 , 6.15d-9 ];, 2.97d-8 ]
  b0 =  7.16d-4 + g0
  b2 =  5.85d-7 + g2
  b3 = -8.51d-8 + g3
  b10= [ 6.425d-3, 6.448d-3];, 6.268d-3]
  b11= [-5.73d-6 ,-5.87d-6 ];,-5.61d-6 ]

  e0 = [-0.24d ,  -0.37d ];,   -0.23d ]
  e1 = [ 0.71d ,   0.20d ];,    4.65d ]
  e2 = [-3.646d,  -3.299d];,   -5.814d]
  e3 = [18.00d ,  -3.342d];,  -13.73d ]
  a0 = -0.11d   + e0
  a2 = -2.457d  + e2
  a3 = 52.02d   + e3

  Scale  = DBLARR(1,64,/NOZERO) ; pre-allocate for later
  Offset = DBLARR(1,64,/NOZERO) ; pre-allocate for later

  rads2degs = 180d/!DPI


  ; pre-allocate array of fill values for elevation delta
  Elevation_delta    = in_file.DIM2_ELEVATION ; get right size, be it 32 or 48 anodes
  Elevation_delta[*] = 65535d ; make all fill

  ; Convert Energy table of Sweep to keV
  keV = in_file.DIM1_E / 1000d ; keV of Energy table


  ; Step 0  ; mag_anode must be 0-47 only - enforced before generation of Level 2 files!
  mag_anode = ROUND( L2_mag_look_dir )
  ;  Rounding just to make integer and avoid rounding errors.


  ; Step 1a, find elevation in degrees and cap at +/- 35 degs
  deltaB = rads2degs * ATAN(in_file.MAG_VECTOR[*,2] / $
         SQRT(in_file.MAG_VECTOR[*,0]*in_file.MAG_VECTOR[*,0] + $
              in_file.MAG_VECTOR[*,1]*in_file.MAG_VECTOR[*,1] ) ) ; in degrees
  deltaB( WHERE(deltaB GT  35d ,/NULL) ) =  35d
  deltaB( WHERE(deltaB LT -35d ,/NULL) ) = -35d
  ; Step 1b done inside For loop

  ; Do For loop through array as MAG vector can change
  nrec_minus1 = N_ELEMENTS(in_file.T) - 1L
  FOR rec = 0L,nrec_minus1  DO BEGIN
    ; ignore if MAG vector is all zeros or fills
    IF ((in_file.MAG_VECTOR[rec,0] EQ 0L) AND (in_file.MAG_VECTOR[rec,1] EQ 0L) AND $
        (in_file.MAG_VECTOR[rec,2] EQ 0L) ) THEN CONTINUE
        ; if no MAG vector (all zeros), skip (leave as 65535)

    IF ((in_file.MAG_VECTOR[rec,0] EQ 9990000.0) AND (in_file.MAG_VECTOR[rec,1] EQ $
         9990000.0  ) AND (in_file.MAG_VECTOR[rec,2] EQ 9990000.0  )) THEN CONTINUE
         ; Level 3 fill value; or if MAG vector contains any fills Level 3; 9990000.0

    ; Do For loop through sensors, although only the first 2
    FOR S = 0,1 DO BEGIN ; no S = 2, will never do for E300
      Start_index = S * 16
      ; E060 starts at anode 0, E180 starts at anode 16, and E300 at anode 32

      ; Step 1a - done outside of these for loops above as can do once on whole array.
      ; Step 1b - figure out UP or DOWN

      IF S EQ 0 THEN BEGIN
        CASE 1 OF ; for E060
          ((mag_anode[rec] GE 20) AND (mag_anode[rec] LE 43)) : sign = -1
          ELSE : sign =  1
        ENDCASE
      ENDIF ELSE BEGIN ; IF S EQ 1
        CASE 1 OF ; for E180
          ((mag_anode[rec] GE 12) AND (mag_anode[rec] LE 35)) : sign = 1
          ELSE : sign =  -1
        ENDCASE
      ENDELSE

      dB = deltaB[rec]
      IF (sign EQ -1) THEN dB = -dB
      ; also change the V_in line to use dB instead of deltaB[rec]

      ; Step 2 - Calculate voltage on deflector based on Energy (eV) of J.DIM1_E
      E = keV[rec,*,Start_index] ; keV, per anode. - same for all anodes, so just using
the first one

      b1     = b10[S] + b11[S] * E
      V_in   = E * (b0[S] + b1*dB + b2[S]*dB*dB + b3[S]*dB*dB*dB) ; in kV
      ; V_in size is 1x64

      ; Assume all are same sign, so V_in[0] represents up or down.

      ; Step 3 - Convert Volts to DAC, round and back to Volts.
      ; Different if above/below 300 V, or 0.3 keV
      indV_GE_300 = WHERE( ABS(V_in) GE 0.300d , COMPLEMENT = indV_LT_300 , /NULL)
      IF (S EQ 0) THEN BEGIN
        ; Source of scale and offset values is default_parameters_v400.xlsx
        IF (sign EQ 1) THEN BEGIN ; E060 up
          Scale[ indV_GE_300] = 2.4308d  ; E060_DFL_UP_HG
          Offset[indV_GE_300] = 42.8220d ; E060_DFL_UP_HG
          Scale[ indV_LT_300] = 0.0728d  ; E060_DFL_UP_LG
          Offset[indV_LT_300] = 1.2663d  ; E060_DFL_UP_LG
        ENDIF ELSE BEGIN ; E060 down
          Scale[ indV_GE_300] = 2.4513d  ; E060_DFL_DN_HG
          Offset[indV_GE_300] = 42.7790d ; E060_DFL_DN_HG
          Scale[ indV_LT_300] = 0.0727d  ; E060_DFL_DN_LG
          Offset[indV_LT_300] = 1.3000d  ; E060_DFL_DN_LG
        ENDELSE
      ENDIF ELSE BEGIN
        ; if S EQ 1
        IF (sign EQ 1) THEN BEGIN ; E180 up
          Scale[ indV_GE_300] = 2.4431d  ; E180_DFL_UP_HG
          Offset[indV_GE_300] = 43.4540d ; E180_DFL_UP_HG
          Scale[ indV_LT_300] = 0.0728d  ; E180_DFL_UP_LG
          Offset[indV_LT_300] = 1.4000d  ; E180_DFL_UP_LG
        ENDIF ELSE BEGIN ; E180 down
          Scale[ indV_GE_300] = 2.4415d  ; E180_DFL_DN_HG
          Offset[indV_GE_300] = 48.2290d ; E180_DFL_DN_HG
          Scale[ indV_LT_300] = 0.0741d  ; E180_DFL_DN_LG
          Offset[indV_LT_300] = 1.6000d  ; E180_DFL_DN_LG
        ENDELSE
      ENDELSE

      DAC = ROUND( (ABS(V_in) * 1000d -  Offset) /Scale , /L64)
      ; V in volts, hence V_in * 1000d

      ; Fix upper limit at 4095
      ind_DAC = WHERE(DAC GE 4095,/NULL)   ; this is deliberately as needed later.
      DAC[   ind_DAC              ] = 4095 ; this is an integer
      DAC[WHERE(DAC LT    0,/NULL)] =    0 ; this is an integer - added for safety...

      ; Convert back and put in kV
      V_out = (DOUBLE(DAC) * Scale + Offset) / 1000d ; in kiloVolts

      ; Step 4 Calculate actual Deflection
      VoE = V_out / E ; positive
      IF V_in[0] LT 0d THEN VoE =  - VoE ; make negative if I had to - now using V_in for
sign.  V_in is size 1x64, assume first index is same sign for all

      ;FOR E = 0,63 DO IF V_in[E] LT 0d THEN VoE[E] =  - VoE[E] ; make negative if I had
      ;to - now using V_in for sign.  V_in is size 1x64, assume signs of index can vary
; ROB NOTE,
; Should the whole array be the same sign or not???
; Running tests on all HRS ELC data from PJ1 thorugh PJ11, V_in was only different signs
for 424 JADE-E records, when deltaB was between -0.5232 to -0.3575 degrees
; Am leaving it the original way (whole array, not a for loop) as quicker.
;
; Since this code (below with DO_DAC4095_Only EQ 1) only gives non-fill values when
; DAC = 4095 (large deflections) it does not apply for some deltaB’s of less than a
; degree where this problem could be.
;
; Rob NOTE END

      a1 = 154.85d + 1.582d-4 * E + e1[S]
      ; now fill in index 0 or 16:
      IF DO_DAC4095_Only EQ 1 THEN BEGIN
        IF N_ELEMENTS(ind_DAC) GT 0 THEN BEGIN
          ; ONLY set Elevation_delta if DAC == 4095
          Elevation_delta[rec,ind_DAC,Start_index] = $
            a0[S] + a1*VoE[ind_DAC] + a2[S]*VoE[ind_DAC]*VoE[ind_DAC] + $
            a3[S]*VoE[ind_DAC]*VoE[ind_DAC]*VoE[ind_DAC]
        ENDIF
      ENDIF ELSE BEGIN
        ; Fill in Elevation_delta for all DAC values
        Elevation_delta[rec,*,Start_index] = $
            a0[S] + a1*VoE + a2[S]*VoE*VoE + a3[S]*VoE*VoE*VoE
      ENDELSE
      ; fill in indexes 1-15 and 17-31 below
    ENDFOR
  ENDFOR

  ; Since E and deflection is same for all anodes, just calculated the first above and
copy out
  FOR z =  1,15 DO Elevation_delta[*,*,z] =Elevation_delta[*,*, 0]; make  0-15 same as  0
  FOR z = 17,31 DO Elevation_delta[*,*,z] =Elevation_delta[*,*,16]; made 16-31 same as 16
  ; nothing to do for E300, it's all fills.

  ; Frederic/Rob - do we need to check the output is not < -35  or > + 35?.  Printing for
now...
  IF MIN(Elevation_delta) LT 65534d THEN BEGIN
    MAX_Elevation_delta = MAX(Elevation_delta[WHERE(Elevation_delta LT 65534d,/NULL)], $
                          MIN = MIN_Elevation_delta) ; need to ignore fills
    IF MIN_Elevation_delta LT -35.2d THEN $
      PRINT,'--> WARNING: HRS Electron Deflection LT -35',MIN_Elevation_delta
      ; give a little leeway of -35.2 and not just -35
    IF MAX_Elevation_delta GT +35.2d THEN $
      PRINT,'--> WARNING: HRS Electron Deflection GT +35',MAX_Elevation_delta
      ; give a little leeway of  35.2 and not just  35
  ENDIF ELSE BEGIN
    IF DO_DAC4095_Only EQ 1 THEN BEGIN
      PRINT,'---> DAC was always below 4095, or all MAG_VECTOR data was fill values'
    ENDIF ELSE BEGIN
      PRINT,'---> ALL the MAG_VECTOR data was fill values'
    ENDELSE
  ENDELSE

  RETURN,Elevation_delta
END
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```