Appendix A

Tables

CML	Power (GW)	Total (GW)	%
162.853	192.5	1350	14.3
163.156	218	1383	15.8
163.458	147.7	1467	10.1
163.760	292.3	1482	19.7
164.062	226.0	1327	17.0
164.365	166.7	1307	12.8
164.667	167.5	1322	12.7
164.968	245.4	1365	18.0
165.270	269.4	1440	18.7
165.574	243.8	1471	16.6
165.876	343.3	1505	22.8
166.178	90.6	1363	6.6
166.480	161.0	1383	11.6
166.782	166.5	1389	12.0
167.085	226.7	1433	15.87
167.387	132.2	1406	9.4
167.688	152.0	1403	10.8
167.991	155.0	1401	11.1
168.294	189.2	1400	13.5
168.596	237.9	1417	16.8
168.898	248.6	1426	17.4
169.200	224.6	1461	15.4
169.502	213.4	1504	14.2
169.805	375.3	1597	23.5
170.107	258.3	1476	17.5
170.408	226.0	1453	15.6
170.711	231.3	1459	15.9
171.014	157.6	1494	10.5
171.316	382.4	1631	23.4
171.617	186.3	1555	12.0
171.920	253.6	1609	15.8
172.223	273.9	1607	17.0
172.525	177.9	1518	11.7
172.827	217.7	1532	14.2
173.128	301.4	1604	18.8
173.431	314.6	1604	19.6
173.734	202.2	1527	13.2
174.036	122.5	1546	7.9
174.337	170.7	1580	10.8
174.640	211.3	1591	13.3

CML	Power (GW)	Total (GW)	%
174.943	196.4	1605	12.2
175.245	226.7	1631	13.9
175.547	272.2	1632	16.7
175.850	263.0	1723	15.3
176.151	341.5	1877	18.2
176.454	221.9	1666	13.3
176.756	248.3	1615	15.4
177.059	173.7	1656	10.5
177.360	302.6	1734	17.5
177.663	166.7	1657	10.1
177.965	164.1	1651	9.9
178.267	139.9	1663	8.4
178.570	149.4	1707	8.8
178.872	167.8	1738	9.7
179.174	265.1	1787	14.8
179.476	155.8	1758	8.9
179.779	204.6	1735	11.8
180.080	175.9	1770	10.0
180.383	201.9	1774	11.4
180.685	217.8	1779	12.3
180.988	242.7	1802	13.5
181.290	132.8	1788	7.4
181.592	233.6	1864	12.5
181.894	374.6	1973	19.0
182.197	239.2	1961	12.2
182.499	226.5	1867	12.1
182.800	132.3	1798	7.4
183.103	136.7	1807	7.6
183.405	223.7	1860	12.0
183.708	234.8	1925	12.2
184.010	199.7	1942	10.3
184.312	207.4	1945	10.7
184.614	222.3	2012	11.0
184.917	201.5	2089	9.6
185.218	218.7	2175	10.12
185.520	129.4	2058	6.3
185.823	128.5	2085	6.2
186.126	146.9	2133	6.9
186.428	230.2	2168	10.6
186.730	142.0	2190	6.5

Table A.1. Power emitted by the auroral bridge, total power emitted by the whole aurora, and contribution of the auroral bridge to the total emitted power (in %) for case k24.

CML	Power (GW)	Total (GW)	%
146.398	394.6	1615	24.4
146.701	685.2	1664	41.2
147.003	176.4	1330	13.3
147.305	197.4	1409	14.0
147.608	325.1	1478	22.0
147.910	234.1	1483	15.8
148.212	375.7	1549	24.3
148.515	813.8	1667	48.8
149.119	323.4	1651	19.6
149.421	292.0	1466	19.9
149.724	327.1	1438	22.7
150.026	400.3	1500	26.7
150.328	222.9	1455	15.3
150.631	338.7	1504	22.5
150.933	575.9	1928	29.9
151.235	365.0	1868	19.5
151.538	216.3	1549	14.0
151.840	347.4	1598	21.7
152.142	339.2	1526	22.2
152.445	341.3	1484	23.0
152.747	325.3	1425	22.8
153.049	387.4	1397	27.7
153.352	324.0	1428	22.7
153.654	355.4	1432	24.8
153.956	376.4	1454	25.9
154.259	374.1	1458	25.7
154.561	324.1	1552	20.9
154.863	883.6	1873	47.2
155.166	227.4	1525	14.9
155.468	345.7	1539	22.5
155.770	635.4	1907	33.3
156.072	1137.1	2239	50.8
156.375	560.9	2394	23.4
156.677	503.6	2440	20.6
156.979	877.5	2224	39.5
157.282	478.0	1771	27.0
157.584	635.3	1798	35.3
157.886	655.5	1738	37.7
158.189	497.5	1680	29.6
158.491	527.8	1746	30.2
158.793	532.2	1713	31.1
159.096	484.9	1605	30.2
159.398	494.8	1572	31.5
159.700	404.5	1565	25.8
160.003	476.0	1574	30.2
160.305	529.8	1603	33.1
160.607	485.5	1614	30.1
160.910	540.0	1726	31.3
161.212	780.8	1718	45.4
161.514	345.5	1521	22.7
161.817	400.0	1484	27.0
162.119	228.6	1501	15.2
162.421	441.1	1512	29.2
162.724	379.4	1495	25.4
163.026	343.2	1467	23.4
163.328	339.3	1460	23.2
163.630	399.6	1461	27.4
163.933	376.3	1455	25.9
164.235	447.4	1480	30.2
164.537	507.1	1750	29.0
164.840	242.7	1599	15.2
165.142	337.2	1573	21.4
165.444	492.6	1715	28.7
165.747	212.4	1630	13.0
166.049	204.2	1615	12.6
166.351	318.5	1930	16.5
166.654	2089.7	2773	75.4
166.956	217.4	1949	11.1
167.258	395.6	2079	19.0
167.561	1225.2	2533	48.4
167.863	174.6	1741	10.0
168.165	207.4	1694	12.2
168.468	266.9	1685	15.8
168.770	367.8	1651	22.3
169.072	417.5	1640	25.5
169.375	409.1	1746	23.4
169.677	444.7	2088	21.3
169.979	752.5	1968	38.2
170.281	376.8	1801	20.9

Table A.2. Power emitted by the auroral bridge, total power emitted by the whole aurora, and contribution of the auroral bridge to the total emitted power (in %) for case k57.

CML	Power (GW)	Total (GW)	%
163.792	177.4	1289	13.8
164.094	390.6	1380	28.3
164.397	136.7	1217	11.2
164.699	178.7	1506	11.9
165.001	149.8	1296	11.6
165.304	118.7	1270	9.3
165.606	282.8	1548	18.3
165.908	99.2	1217	8.2
166.211	37.6	1174	3.2
166.513	244.3	1540	15.9
166.815	145.5	1481	9.8
167.118	86.1	1452	5.9
167.420	612.1	1630	37.6
167.722	36.6	1265	2.9
168.025	77.5	1341	5.8
168.327	324.6	1464	22.2
168.629	53.3	1251	4.3
168.931	124.6	1461	8.5
169.234	64.8	1368	4.7
169.536	43.2	1135	3.8
169.838	120.8	1481	8.2
170.141	65.4	1443	4.5
170.443	48.6	1217	4.00
170.745	142.3	1597	8.9
171.048	44.9	1366	3.3
171.350	55	1144	4.8
171.652	81.7	1172	7.0
171.955	169.2	1323	12.8
172.257	263.7	1403	18.8
172.559	70.8	1275	5.6
172.862	100.6	1337	7.5
173.164	453.3	1437	31.5
173.466	78.3	1298	6.0
173.769	81.8	1577	5.2
174.071	392.9	1675	23.5
174.373	60.0	1338	4.5
174.676	77.8	1377	5.6
174.978	135.5	1583	8.6
175.280	298.9	1595	18.7
175.582	78.3	1288	6.1
175.885	116.2	1364	8.5
176.187	155.6	1500	10.4
176.489	149.0	1444	10.3
176.792	63.4	1261	5.0
177.094	96.3	1252	7.7
177.396	57.2	1206	4.7
177.699	40.4	1144	3.5
178.001	57.9	1149	5.0
178.303	32.4	1158	2.8
178.606	107	1264	8.5
178.908	217.6	1311	16.6
179.210	31.5	1222	2.6
179.513	47.7	1236	3.9
179.815	287.3	1551	18.5
180.117	27.3	1453	1.9
180.420	62.9	1545	4.1
180.772	298.2	1838	16.2
181.024	105.1	1540	6.8
181.327	82.6	1553	5.3
181.629	518.4	1667	31.1
181.931	78.0	1356	5.8
182.233	89.7	1378	6.5
182.536	142.7	1326	10.8
182.838	46.7	1203	3.9
183.140	142.5	1544	9.2
183.443	166.1	1567	10.6
183.745	73.9	1338	5.5
184.047	376.8	1462	25.8
184.350	241.6	1412	17.1
184.652	51.7	1236	4.2
184.954	42.5	1183	3.6
185.257	54.8	1198	4.6
185.559	91.8	1233	7.4
185.861	112.8	1270	8.9
186.164	79.0	1207	6.5
186.466	43.4	1192	3.6
186.768	253.5	1307	19.4
187.071	212.2	1332	15.9
187.373	36.3	1306	2.8
187.675	47.0	1348	3.5

Table A.3. Power emitted by the auroral bridge, total power emitted by the whole aurora, and contribution of the auroral bridge to the total emitted power (in %) for case k61.

Appendix B

IDL routines

B.1 irene.script

```
; based on:
; /LPAP/HST/work2/grodent/juno/HST_PJ06_data_analysis/analyse_k0a.pro

; Modified by Irene Pardo Cantos

;-------------------------------------------------------------

; working directory
cd /home/LPAP/pardo/results/
idl
@/home/LPAP/pardo/add_lib     ; @ ./add_lib

device,cursor_standard=33

; restore the content of the file.dat
; catalogue41_14634.dat contains the metadata
restore, '/LPAP/HST/data/catalogue/catalogue41_14634.dat'

; Case study
;rootname = 'od8k57itq'
;rootname = 'od8k61lyq'
rootname = 'od8k24ioq'

;-------------------------------------------------

redu = '/LPAP/HST/redu/jup_stis_14634/'         ; reduced data
proj = '/LPAP/HST/redu/jup_stis_14634/proj/'    ; projections

ind = where(cat_ext.rootname eq rootname)
```

```
; Central Meridian Longitude (CML)
cml = cat_ext(ind).cml

; transforms in physical units
coef = bri_pow_conv_cr(cat_ext,rootname,2.5)

doy = date2doy(cat_ext(ind).date)               ; day of the year

cimage = clean_image(cat_ext,rootname)          ; 10 seconds frames

; coef(0): intensity in kR, coef(1): power in W
cimage_pow = cimage*coef(1)

incl = cat_ext(ind).pitch-cat_ext(ind).tele_data(10)     ; inclination

 ; sub Earth latitude in degrees
latst = cat_ext(ind).latst

req = 71492.            ; equatorial radius of Jupiter in km
rpol = 66854.           ; polar radius of Jupier in km
alt = 400.                 ; altitude in km
alpha = incl/180.0*!PI        ; inclination in radians

lamda0 = latst*!pi/180.           ; sub Earth latitude in radians

; angle holded by a pixel (arcsec) *1024 = 25 arcsec (image)
pixangle = cat_ext(ind).tele_data(2)
d = cat_ext(ind).d                     ; HST-Jupiter distance in km

;virtual center of Jupiter
cx = cat_ext(ind).tele_data(7)        ; x-coord
cy = cat_ext(ind).tele_data(8)        ; y-coord

pixdist = d*pixangle/3600.0/180.0*!PI    ; arcsec to degrees to radians
reqpix = req/pixdist             ; conversion to pixels
rpolpix = rpol/pixdist
altpix = alt/pixdist

;-----------------------------------------------------

; Projection
;------------

filen = proj + rootname + '_nobkg_400_cps_map.idl'      ; map name
print,'restoring ' + filen
restore,filen
```

```
; in '_ephem_data.idl' there is more information than in the catalogue
file_ephem = redu + rootname + '_ephem_data.idl'    ; ephem: ephemeride

restore,file_ephem

;-----------------------------------------------------

cml_list = cml

; window: creates a graphics window. Size in pixels (xs,ys)
window, xs=1024, ys=1024

; loadct: loads one of the predefined IDL color tables
; 1: blue-white table, 3: red-white table, etc
loadct,3

; applies gamma correction to a color table.
; Values less than 1.0 yield lower contrast
gamma_ct,0.25

;-----------------------------------------------------

j = 248                        ; n image ;first image (j=0=t)
map = implan(*,*,j)

; shift the map, turns it, etc
mop = shift(map,(180.-cml_list[j])*10,0)

; map projection
bpglobe_mod,90,0,180,(mop*255>0),/grid,/horizon,londel=10,latdel=10
          ; 90: North pole.
          ; 0: inclination (over the pole).
          ; 180: 180degrees placed in the bottom of the image.
          ; Horizon: plot a borderline.
          ; londel=10,latdel=10: lon and lat grid every 10deg.

;-----------------------------------------------------

; select points in the image to obtain their coordinates.
; Click on the right bottom to finish.
traceoval,longitude,latitude

; longitudeb = 360.-longitude   ; to correct the negative sign in lon
longitudeb = -longitude

; position of the center of Jupiter
```

```
cx0 = cx
cy0 = cy

;---------------------------------------------------

; the .RUN command compiles procedures, functions and/or
; main programs in memory
.run

nn = n_elements(longitudeb)

; fltarr creates a floating-point vector of dimension(nn)
x1 = fltarr(nn)
y1 = x1

for i=0,nn-1 do begin
pos_xy = planet2xy(reqpix+altpix,rpolpix+altpix,cx0,cy0,alpha,lamda0,$
         latitude[i]*!dtor,(cml_list[j]-longitudeb[i])*!dtor)
x1[i] = pos_xy[0,0,0]
y1[i] = pos_xy[0,0,1]
endfor

end   ; end run

imi = cimage_pow[*,*,j]         ; image. pow: power in W
imi2 = imi

; overlap points on the image (from lat-lon to X-Y)
imi2(x1,y1) = max(imi)*1.5

; tvscl scales the intensity values of the image (imi) into the range
; of the Direct Graphics image display device and outputs the data
; to the image display at the specified location
tvscl,imi2

; Cut the image selecting the region of interest
;-------------------------------------------------------
 ; defroi: defines an irregular region of interest of an image
z = defroi(1024,1024)

; total: adds intensities of the selected region
;print, total(imi[z])

; rdpix: interactively displays the X position, Y position and pixel
;value at the cursor
rdpix, imi2
```

```
;-------------------------------------------------

dummy = imi*0.
dummy[z] = 1.
zdummy = dummy[578:650,758:830]  ; specify coordinates of the new region

; new image (ama) which is the zoom of the original image (imi)
ama = imi[578:650,758:830]
s = size(ama)
print, s

; rebin: resizes a vector or array to dimensions given by the parameters
amma = rebin(ama, s[1]*4.,s[2]*4.)
;amma = congrid(ama, s[1]*2,s[2]*2)    ; can use congrid instead of rebin
azdummy = rebin(zdummy, s[1]*4.,s[2]*4.)
wz=where(azdummy eq 1.)

tvdg, amma                       ; changes the size of the window
;print, total(amma[wz])/16.      ; total(amma) with pixel correction

intensity_imi = total(imi[z])
intensity_amma = total(amma[wz])/16.

; show the solutions (total power of a region in W)
print, intensity_imi, intensity_amma

;-------------------------------------------------

; Save files
;------------

$pwd      ; $ acts out of IDL

jj = strcompress((string(j)),/remove_all)

lon = longitude
lat = latitude
int_imi = intensity_imi
int_amma = intensity_amma

filename = rootname + '_' + jj + '_CML'

; save files in IDL format
save, filename = filename+'.idl', lon, lat, int_imi, int_amma, cml

restore, filename + '.idl'
```

```
; save files in a .txt file
output = '/home/LPAP/pardo/results/'+filename+'.txt'

s = size(lon)
length = s[1]

openw,1,output

printf,1, 'intensity_imi =', int_imi
printf,1, 'intensity_amma =', int_amma
printf,1, '   '
printf,1,'Longitude ,' Latitude ,' CML '

for ii=0,length-1 do begin
   printf,1,format='(f9.4,x,f8.4,x,f8.4)',lon[ii],lat[ii],cml[j]
endfor

close,1

;-----------------------------------------------------

; Put all the retrieved coordinates together
;-----------------------------------------------------
newfilename = rootname + '_all_CML'

; .TXT file
newoutput = '/home/LPAP/pardo/results/'+newfilename+'.txt'

openw,1,newoutput
;printf,1,'Longitude ,' Latitude ,' CML '

.run

for x=0,9 do begin
    xx = strcompress((string(x)),/remove_all)
    filen = rootname+'_00'+xx+'_CML.idl'
    restore, filen
    s = size(lon)
    length = s[1]
    for ii=0,length-1 do printf,1,format='(f9.4,x,f8.4,x,f8.4)',$
                                  lon[ii],lat[ii],cml[x]
endfor ;x

for y=10,99 do begin
    yy = strcompress((string(y)),/remove_all)
    filen = rootname+'_0'+yy+'_CML.idl'
    restore, filen
```

```
    s = size(lon)
    length = s[1]
    for ii=0,length-1 do printf,1,format='(f9.4,x,f8.4,x,f8.4)',$
                                  lon[ii],lat[ii],cml[y]
endfor ;y

for z=100,248 do begin
    zz = strcompress((string(z)),/remove_all)
    filen = rootname+'_'+zz+'_CML.idl'
    restore, filen
    s = size(lon)
    length = s[1]
    for ii=0,length-1 do printf,1,format='(f9.4,x,f8.4,x,f8.4)',$
                                  lon[ii],lat[ii],cml[z]
endfor ;z

end

close,1

;--------

; .IDL file

openr,2,newfilename+'.txt'

.run

lon_c = 0.
lat_c = 0.
cml_c = 0.

lines = file_lines(newfilename+'.txt')    ; number of lines of a file

; FLTARR: creates a floating-point vector or array of the specified
; dimensions (lines) (it's like FINDGEN function)
lon = fltarr(lines)

lat = fltarr(lines)
cml = fltarr(lines)

ic=0

while ~ eof(2) do begin
      readf,2,format='(f9.4,x,f8.4,x,f8.4)',lon_c,lat_c,cml_c
      lon[ic]=lon_c
      lat[ic]=lat_c
```

```
      cml[ic]=cml_c
      ic+=1
      print,format='(f9.4,x,f8.4,x,f8.4)',lon_c,lat_c,cml_c
endwhile

end

close,2

save,filename='/home/LPAP/pardo/results/'+newfilename+'.idl',lon,lat,cml
```

B.2 ionosph_to_msph_ipc

```
; Irene Pardo Cantos

; MAPPING FROM THE IONOSPHERE TO THE MAGNETOSPHERE
; USING THE MAPPING_FUNCTION (Vogt et al., 2011, 2015)

;-------------------------------------------------------

.r /LPAP/HST/work/commun/idl_library/mapping_program_june_2019/$
mapping_function2019.pro

; Choose the case to study
;--------------------------
;rootname = 'od8k57itq'
;rootname = 'od8k61lyq'
rootname = 'od8k24ioq'

filename = '/home/pardo/results/' + rootname + '_all_CML.idl'

restore, filename

lon = -lon

cd, '/home/LPAP/pardo/'
set_dev        ; call set_dev.pro
loadct,39      ; Color Table "Rainbow + White"
device,/color

; Plot the results to check locations
plot, lon, lat, psym=1, xtitle='Longitude', ytitle='Latitude',$
background=255, color=0, charsize=1.5
```

```
nn = n_elements(lon)
nns = strcompress(string(nn-1),/remove_all)

; Create a vector to keep the results
rrr = fltarr(nn)  ; Radial Distance (in Rj)
lot = fltarr(nn)  ; Local Time LT (in hours 0-24)

;---------------------------------------

.run

for i = 0,nn-1 do begin
    print,i,' / ',nns

    ; FUNCTION (choose one model)
;    msph = mapping_function2019('ion_to_mag',cml[i],lat[i],lon[i],$
     'jrm09','fieldline_tracing')
;    msph = mapping_function2019('ion_to_mag',cml[i],lat[i],lon[i],'jrm09')
;    msph = mapping_function2019('ion_to_mag',cml[i],lat[i],lon[i],$
     'gam','fieldline_tracing')
;    msph = mapping_function2019('ion_to_mag',cml[i],lat[i],lon[i],'gam')
    msph = mapping_function2019('ion_to_mag',cml[i],lat[i],lon[i],$
    'khurana_jrm09','fieldline_tracing')

    ; OUTPUTS
    rrr[i] = msph[0]  ; Radial Distance (in Rj)
    lot[i] = msph[1]  ; Local Time LT (in hours 0-24)
endfor

end

; The program returns an array of flag values (-999, -998, -996, etc.)
; if the point cannot be mapped for some reason.

;-----------------------------------

; SAVE RESULTS
;---------------

; Choose the model name
;newfilename = rootname + '_RD_LT_JRM09_fieldline'
;newfilename = rootname + '_RD_LT_JRM09_flux'
;newfilename = rootname + '_RD_LT_GAM_fieldline'
;newfilename = rootname + '_RD_LT_GAM_flux'
newfilename = rootname + '_RD_LT_khurana_fieldline'

;---------------
```

```
; .TXT FILE

newoutput = '/home/LPAP/pardo/results/'+newfilename+'.txt'
openw,1,newoutput

.run

for x=0,nn-1 do begin
    printf,1,format='(f8.4,x,f8.4)',rrr[x],lot[x]
endfor

end

close,1

;---------------

; .IDL FILE

openr,2,newoutput

.run

rrr_c = 0.
lot_c = 0.

; Number of lines of a file
lines = file_lines(newoutput)

rrr = fltarr(lines)
lot = fltarr(lines)

ic=0

while ~ eof(2) do begin
      readf,2,format='(f8.4,x,f8.4)',rrr_c,lot_c
      rrr[ic]=rrr_c
      lot[ic]=lot_c
      ic+=1
      print,format='(f8.4,x,f8.4)',rrr_c,lot_c
endwhile

end

close,2
```

```
print, 'rrr : ', rrr
print, 'lot : ', lot

save, filename = '/home/LPAP/pardo/results/'+newfilename+'.idl', rrr, lot

;-------------------------------------

; Check where there are errors (-99999)
;--------------------------------------
restore, '/home/LPAP/pardo/results/'+newfilename+'.idl'

i_rrr_nan = where(rrr eq -99999.0, count, complement=rrr_nan_c,$
ncomplement=count_c)
print, 'Number of rrr = -99999.0 :', count
print, 'Subscripts of rrr = -99999.0 :', i_rrr_nan
print, 'Number of rrr /= -99999.0 :', count_c
print, 'Subscripts of rrr /= -99999.0 :', rrr_nan_c

i_lot_nan = where(lot eq 9999.00, countt, complement=lot_nan_c,$
ncomplement=count_cc)
print, 'Number of lot = 9999.00 :', countt
print, 'Subscripts of lot = 9999.00 :', i_lot_nan
print, 'Number of lot /= 9999.00 :', count_cc
print, 'Subscripts of lot /= 9999.00 :', lot_nan_c

;-------------------------------------

; Put all the models together
;----------------------------
name_model_1 = '/home/pardo/results/'+ rootname + '_RD_LT_JRM09_fieldline'
name_model_2 = '/home/pardo/results/'+ rootname + '_RD_LT_JRM09_flux'
name_model_3 = '/home/pardo/results/'+ rootname + '_RD_LT_GAM_fieldline'
name_model_4 = '/home/pardo/results/'+ rootname + '_RD_LT_GAM_flux'
name_model_5 = '/home/pardo/results/'+ rootname + '_RD_LT_khurana_fieldline'

restore, name_model_1+'.idl'
rrr_1 = rrr
lot_1 = lot

restore, name_model_2+'.idl'
rrr_2 = rrr
lot_2 = lot

restore, name_model_3+'.idl'
rrr_3 = rrr
lot_3 = lot
```

```
restore, name_model_4+'.idl'
rrr_4 = rrr
lot_4 = lot

restore, name_model_5+'.idl'
rrr_5 = rrr
lot_5 = lot

nn = n_elements(rrr_1)

filename_models = rootname + '_5MODELS'

:---------------

; .TXT file

output_models = '/home/LPAP/pardo/results/'+filename_models+'.txt'

openw,3,output_models

.run

for i=0,nn-1 do begin
    printf,3,format='(f8.4,x,f8.4,x,f8.4,x,f8.4,x,f8.4,x,f8.4,x,f8.4,x,$
    f8.4,x,f8.4,x,f8.4)',rrr_1[i],lot_1[i],rrr_2[i],lot_2[i],rrr_3[i],$
    lot_3[i],rrr_4[i],lot_4[i],rrr_5[i],lot_5[i]
endfor

end

close,3

:---------------

; .IDL FILE

openr,4,output_models

.run

rrr_1_c = 0.
lot_1_c = 0.
rrr_2_c = 0.
lot_2_c = 0.
rrr_3_c = 0.
lot_3_c = 0.
rrr_4_c = 0.
```

```
lot_4_c = 0.
rrr_5_c = 0.
lot_5_c = 0.

lines = file_lines(output_models)

rrr_1 = fltarr(lines)
lot_1 = fltarr(lines)
rrr_2 = fltarr(lines)
lot_2 = fltarr(lines)
rrr_3 = fltarr(lines)
lot_3 = fltarr(lines)
rrr_4 = fltarr(lines)
lot_4 = fltarr(lines)
rrr_5 = fltarr(lines)
lot_5 = fltarr(lines)

ic=0

while ~ eof(4) do begin
      readf,4,format='(f8.4,x,f8.4,x,f8.4,x,f8.4,x,f8.4,x,f8.4,x,f8.4,x,$
      f8.4,x,f8.4,x,f8.4)',rrr_1_c,lot_1_c,rrr_2_c,lot_2_c,rrr_3_c,lot_3_c,$
      rrr_4_c,lot_4_c,rrr_5_c,lot_5_c

      rrr_1[ic]=rrr_1_c
      lot_1[ic]=lot_1_c
      rrr_2[ic]=rrr_2_c
      lot_2[ic]=lot_2_c
      rrr_3[ic]=rrr_3_c
      lot_3[ic]=lot_3_c
      rrr_4[ic]=rrr_4_c
      lot_4[ic]=lot_4_c
      rrr_5[ic]=rrr_5_c
      lot_5[ic]=lot_5_c

      ic+=1
endwhile

end

close,4

save, filename = '/home/LPAP/pardo/results/'+filename_models+'.idl',$
rrr_1, lot_1, rrr_2, lot_2, rrr_3, lot_3, rrr_4, lot_4, rrr_5, lot_5
```

B.3 mean_stdev_flag_models_ipc

```
; Irene Pardo Cantos

; PROGRAM TO COMPARE THE 5 MAGNETIC FIELD MODELS USED TO CALCULATE THE
; LOCATION OF THE BRIDGE (radial distance & local time).
; IT CALCULATES THE MEAN POSITION, THE STANDARD DEVIATION OF EACH MODEL,
; AND THE FLAG THAT INDICATES HOW MANY MODELS WHERE USED TO MAKE
; CALCULATIONS (0-5).

;-------------------------------------------------------------

; Choose the case to study
;-------------------------
;rootname = 'od8k57itq'
;rootname = 'od8k61lyq'
rootname = 'od8k24ioq'

filename = '/home/pardo/results/' + rootname + '_5MODELS.idl'

restore, filename

rrr = [[rrr_1],[rrr_2],[rrr_3],[rrr_4],[rrr_5]]
rrrt = transpose(rrr)         ; matrix with rrr (radial distance) data

lot = [[lot_1],[lot_2],[lot_3],[lot_4],[lot_5]]
lott = transpose(lot)         ; matrix with lot (local time) data

print, rrrt
print, lott

nn = n_elements(rrr_1)

; data
;------

.run

rrr_matrix = make_array(5,nn,/float, value=0)
lot_matrix = make_array(5,nn,/float, value=0)

for i=0,nn-1 do begin   ;rows

    rrr_array = [rrrt[0,i],rrrt[1,i],rrrt[2,i],rrrt[3,i],rrrt[4,i]]
    rrr_matrix[*,i] = rrr_array
    lot_array = [lott[0,i],lott[1,i],lott[2,i],lott[3,i],lott[4,i]]
```

```
    lot_matrix[*,i] = lot_array

endfor

end

;---------------------------------

; MEAN, STDDEV, FLAG
;--------------------
;;;; 5 MODELS ;;;;

newfilename10 = '/home/pardo/results/'+rootname+'_mean_stdev_flag_all5'
openw,10,newfilename10+'.txt'
printf,10, 'mean_r', 'stdev_r', 'mean_lot', 'stdv_lot', 'flag'
printf,10,' '

.run

for i=0,nn-1 do begin   ;rows

r = make_array(5,nn,/float, value=-9999)
lot = make_array(5,nn,/float, value=-9999)
mean_r = fltarr(nn)
mean_lot = fltarr(nn)
std_r = fltarr(nn)
std_lot = fltarr(nn)
cov = fltarr(nn)
flag = fltarr(nn)
;mean_lot_rad = fltarr(nn)
;lot_rad = fltarr(nn)

; FLAG = 0 (no model maps the points)
;---------

        if (rrr_matrix[0,i] eq 0.) && (rrr_matrix[1,i] eq 0.) && $
           (rrr_matrix[2,i] eq 0.) && (rrr_matrix[3,i] eq 0.) && $
           (rrr_matrix[4,i] eq 0.) then begin
             printf,10, format='(f9.4,x,f9.4,x,f9.4,x,f9.4,x,f11.4,$
                           x,f6.0)', 0, 0, 0, 0, 0, 0
        endif

    for j=0,4 do begin  ;columns

; FLAG = 4 (4 models map the points)
;---------
```

```
if (rrr_matrix[0,i] gt 0.) && (rrr_matrix[1,i] gt 0.) && $
   (rrr_matrix[2,i] gt 0.) && (rrr_matrix[3,i] gt 0.) && $
   (rrr_matrix[4,i] eq 0.) then begin

   r[j,i] = rrr_matrix[j,i]
   lot[j,i] = lot_matrix[j,i]

   mean_r[i] = mean(r[0:3,i])
   mean_lot[i] = mean(lot[0:3,i])

   std_r[i] = stddev(r[0:3,i])
   std_lot[i] = stddev(lot[0:3,i])

endif else begin

if (rrr_matrix[0,i] gt 0.) && (rrr_matrix[1,i] gt 0.) && $
   (rrr_matrix[2,i] gt 0.) && (rrr_matrix[3,i] eq 0.) && $
   (rrr_matrix[4,i] gt 0.) then begin

   r[j,i] = rrr_matrix[j,i]
   lot[j,i] = lot_matrix[j,i]

   mean_r[i] = mean([r[0,i],r[1,i],r[2,i],r[4,i]])
   mean_lot[i] = mean([lot[0,i],lot[1,i],lot[2,i],lot[4,i]])

   std_r[i] = stddev([r[0,i],r[1,i],r[2,i],r[4,i]])
   std_lot[i] = stddev([lot[0,i],lot[1,i],lot[2,i],lot[4,i]])

endif else begin

if (rrr_matrix[0,i] gt 0.) && (rrr_matrix[1,i] gt 0.) && $
   (rrr_matrix[2,i] eq 0.) && (rrr_matrix[3,i] gt 0.) && $
   (rrr_matrix[4,i] gt 0.) then begin

   r[j,i] = rrr_matrix[j,i]
   lot[j,i] = lot_matrix[j,i]

   mean_r[i] = mean([r[0,i],r[1,i],r[3,i],r[4,i]])
   mean_lot[i] = mean([lot[0,i],lot[1,i],lot[3,i],lot[4,i]])

   std_r[i] = stddev([r[0,i],r[1,i],r[3,i],r[4,i]])
   std_lot[i] = stddev([lot[0,i],lot[1,i],lot[3,i],lot[4,i]])

endif else begin

if (rrr_matrix[0,i] gt 0.) && (rrr_matrix[1,i] eq 0.) && $
   (rrr_matrix[2,i] gt 0.) && (rrr_matrix[3,i] gt 0.) && $
```

```
            (rrr_matrix[4,i] gt 0.) then begin

            r[j,i] = rrr_matrix[j,i]
            lot[j,i] = lot_matrix[j,i]

            mean_r[i] = mean([r[0,i],r[2,i],r[3,i],r[4,i]])
            mean_lot[i] = mean([lot[0,i],lot[2,i],lot[3,i],lot[4,i]])

            std_r[i] = stddev([r[0,i],r[2,i],r[3,i],r[4,i]])
            std_lot[i] = stddev([lot[0,i],lot[2,i],lot[3,i],lot[4,i]])

         endif else begin

         if (rrr_matrix[0,i] eq 0.) && (rrr_matrix[1,i] gt 0.) && $
            (rrr_matrix[2,i] gt 0.) && (rrr_matrix[3,i] gt 0.) && $
            (rrr_matrix[4,i] gt 0.) then begin

            r[j,i] = rrr_matrix[j,i]
            lot[j,i] = lot_matrix[j,i]

            mean_r[i] = mean(r[1:4,i])
            mean_lot[i] = mean(lot[1:4,i])

            std_r[i] = stddev(r[1:4,i])
            std_lot[i] = stddev(lot[1:4,i])

         endif else begin


; FLAG = 3 (3 models map the points)
;---------

         if (rrr_matrix[0,i] gt 0.) && (rrr_matrix[1,i] gt 0.) && $
            (rrr_matrix[2,i] gt 0.) && (rrr_matrix[3,i] eq 0.) && $
            (rrr_matrix[4,i] eq 0.) then begin

            r[j,i] = rrr_matrix[j,i]
            lot[j,i] = lot_matrix[j,i]

            mean_r[i] = mean(r[0:2,i])
            mean_lot[i] = mean(lot[0:2,i])

            std_r[i] = stddev(r[0:2,i])
            std_lot[i] = stddev(lot[0:2,i])

         endif else begin
```

```
if (rrr_matrix[0,i] gt 0.) && (rrr_matrix[1,i] gt 0.) && $
   (rrr_matrix[2,i] eq 0.) && (rrr_matrix[3,i] gt 0.) && $
   (rrr_matrix[4,i] eq 0.) then begin

   r[j,i] = rrr_matrix[j,i]
   lot[j,i] = lot_matrix[j,i]

   mean_r[i] = mean([r[0,i],r[1,i],r[3,i]])
   mean_lot[i] = mean([lot[0,i],lot[1,i],lot[3,i]])

   std_r[i] = stddev([r[0,i],r[1,i],r[3,i]])
   std_lot[i] = stddev([lot[0,i],lot[1,i],lot[3,i]])

endif else begin

if (rrr_matrix[0,i] gt 0.) && (rrr_matrix[1,i] gt 0.) && $
   (rrr_matrix[2,i] eq 0.) && (rrr_matrix[3,i] eq 0.) && $
   (rrr_matrix[4,i] gt 0.) then begin

   r[j,i] = rrr_matrix[j,i]
   lot[j,i] = lot_matrix[j,i]

   mean_r[i] = mean([r[0,i],r[1,i],r[4,i]])
   mean_lot[i] = mean([lot[0,i],lot[1,i],lot[4,i]])

   std_r[i] = stddev([r[0,i],r[1,i],r[4,i]])
   std_lot[i] = stddev([lot[0,i],lot[1,i],lot[4,i]])

endif else begin

if (rrr_matrix[0,i] gt 0.) && (rrr_matrix[1,i] eq 0.) && $
   (rrr_matrix[2,i] gt 0.) && (rrr_matrix[3,i] gt 0.) && $
   (rrr_matrix[4,i] eq 0.) then begin

   r[j,i] = rrr_matrix[j,i]
   lot[j,i] = lot_matrix[j,i]

   mean_r[i] = mean([r[0,i],r[2,i],r[3,i]])
   mean_lot[i] = mean([lot[0,i],lot[2,i],lot[3,i]])

   std_r[i] = stddev([r[0,i],r[2,i],r[3,i]])
   std_lot[i] = stddev([lot[0,i],lot[2,i],lot[3,i]])

endif else begin

if (rrr_matrix[0,i] gt 0.) && (rrr_matrix[1,i] eq 0.) && $
   (rrr_matrix[2,i] gt 0.) && (rrr_matrix[3,i] eq 0.) && $
```

```
   (rrr_matrix[4,i] gt 0.) then begin

   r[j,i] = rrr_matrix[j,i]
   lot[j,i] = lot_matrix[j,i]

   mean_r[i] = mean([r[0,i],r[2,i],r[4,i]])
   mean_lot[i] = mean([lot[0,i],lot[2,i],lot[4,i]])

   std_r[i] = stddev([r[0,i],r[2,i],r[4,i]])
   std_lot[i] = stddev([lot[0,i],lot[2,i],lot[4,i]])

endif else begin

if (rrr_matrix[0,i] gt 0.) && (rrr_matrix[1,i] eq 0.) && $
   (rrr_matrix[2,i] eq 0.) && (rrr_matrix[3,i] gt 0.) && $
   (rrr_matrix[4,i] gt 0.) then begin

   r[j,i] = rrr_matrix[j,i]
   lot[j,i] = lot_matrix[j,i]

   mean_r[i] = mean([r[0,i],r[3,i],r[4,i]])
   mean_lot[i] = mean([lot[0,i],lot[3,i],lot[4,i]])

   std_r[i] = stddev([r[0,i],r[3,i],r[4,i]])
   std_lot[i] = stddev([lot[0,i],lot[3,i],lot[4,i]])

endif else begin

if (rrr_matrix[0,i] eq 0.) && (rrr_matrix[1,i] gt 0.) && $
   (rrr_matrix[2,i] gt 0.) && (rrr_matrix[3,i] gt 0.) && $
   (rrr_matrix[4,i] eq 0.) then begin

   r[j,i] = rrr_matrix[j,i]
   lot[j,i] = lot_matrix[j,i]

   mean_r[i] = mean(r[1:3,i])
   mean_lot[i] = mean(lot[1:3,i])

   std_r[i] = stddev(r[1:3,i])
   std_lot[i] = stddev(lot[1:3,i])

endif else begin

if (rrr_matrix[0,i] eq 0.) && (rrr_matrix[1,i] gt 0.) && $
   (rrr_matrix[2,i] gt 0.) && (rrr_matrix[3,i] eq 0.) && $
   (rrr_matrix[4,i] gt 0.) then begin
```

```
            r[j,i] = rrr_matrix[j,i]
            lot[j,i] = lot_matrix[j,i]

            mean_r[i] = mean([r[1,i],r[2,i],r[4,i]])
            mean_lot[i] = mean([lot[1,i],lot[2,i],lot[4,i]])

            std_r[i] = stddev([r[1,i],r[2,i],r[4,i]])
            std_lot[i] = stddev([lot[1,i],lot[2,i],lot[4,i]])

         endif else begin

         if (rrr_matrix[0,i] eq 0.) && (rrr_matrix[1,i] gt 0.) && $
            (rrr_matrix[2,i] eq 0.) && (rrr_matrix[3,i] gt 0.) && $
            (rrr_matrix[4,i] gt 0.) then begin

            r[j,i] = rrr_matrix[j,i]
            lot[j,i] = lot_matrix[j,i]

            mean_r[i] = mean([r[1,i],r[3,i],r[4,i]])
            mean_lot[i] = mean([lot[1,i],lot[3,i],lot[4,i]])

            std_r[i] = stddev([r[1,i],r[3,i],r[4,i]])
            std_lot[i] = stddev([lot[1,i],lot[3,i],lot[4,i]])

         endif else begin

         if (rrr_matrix[0,i] eq 0.) && (rrr_matrix[1,i] eq 0.) && $
            (rrr_matrix[2,i] gt 0.) && (rrr_matrix[3,i] gt 0.) && $
            (rrr_matrix[4,i] gt 0.) then begin

            r[j,i] = rrr_matrix[j,i]
            lot[j,i] = lot_matrix[j,i]

            mean_r[i] = mean(r[2:4,i])
            mean_lot[i] = mean(lot[2:4,i])

            std_r[i] = stddev(r[2:4,i])
            std_lot[i] = stddev(lot[2:4,i])

         endif else begin


; FLAG = 2 (2 models map the points)
;----------

         if (rrr_matrix[0,i] gt 0.) && (rrr_matrix[1,i] gt 0.) && $
            (rrr_matrix[2,i] eq 0.) && (rrr_matrix[3,i] eq 0.) && $
```

```
   (rrr_matrix[4,i] eq 0.) then begin

   r[j,i] = rrr_matrix[j,i]
   lot[j,i] = lot_matrix[j,i]

   mean_r[i] = mean(r[0:1,i])
   mean_lot[i] = mean(lot[0:1,i])

   std_r[i] = stddev(r[0:1,i])
   std_lot[i] = stddev(lot[0:1,i])

endif else begin

if (rrr_matrix[0,i] gt 0.) && (rrr_matrix[1,i] eq 0.) && $
   (rrr_matrix[2,i] gt 0.) && (rrr_matrix[3,i] eq 0.) && $
   (rrr_matrix[4,i] eq 0.) then begin

   r[j,i] = rrr_matrix[j,i]
   lot[j,i] = lot_matrix[j,i]

   mean_r[i] = mean([r[0,i],r[2,i]])
   mean_lot[i] = mean([lot[0,i],lot[2,i]])

   std_r[i] = stddev([r[0,i],r[2,i]])
   std_lot[i] = stddev([lot[0,i],lot[2,i]])

endif else begin

if (rrr_matrix[0,i] gt 0.) && (rrr_matrix[1,i] eq 0.) && $
   (rrr_matrix[2,i] eq 0.) && (rrr_matrix[3,i] gt 0.) && $
   (rrr_matrix[4,i] eq 0.) then begin

   r[j,i] = rrr_matrix[j,i]
   lot[j,i] = lot_matrix[j,i]

   mean_r[i] = mean([r[0,i],r[3,i]])
   mean_lot[i] = mean([lot[0,i],lot[3,i]])

   std_r[i] = stddev([r[0,i],r[3,i]])
   std_lot[i] = stddev([lot[0,i],lot[3,i]])

endif else begin

if (rrr_matrix[0,i] gt 0.) && (rrr_matrix[1,i] eq 0.) && $
   (rrr_matrix[2,i] eq 0.) && (rrr_matrix[3,i] eq 0.) && $
   (rrr_matrix[4,i] gt 0.) then begin
```

```
   r[j,i] = rrr_matrix[j,i]
   lot[j,i] = lot_matrix[j,i]

   mean_r[i] = mean([r[0,i],r[4,i]])
   mean_lot[i] = mean([lot[0,i],lot[4,i]])

   std_r[i] = stddev([r[0,i],r[4,i]])
   std_lot[i] = stddev([lot[0,i],lot[4,i]])

endif else begin


if (rrr_matrix[0,i] eq 0.) && (rrr_matrix[1,i] gt 0.) && $
   (rrr_matrix[2,i] gt 0.) && (rrr_matrix[3,i] eq 0.) && $
   (rrr_matrix[4,i] eq 0.) then begin

   r[j,i] = rrr_matrix[j,i]
   lot[j,i] = lot_matrix[j,i]

   mean_r[i] = mean(r[1:2,i])
   mean_lot[i] = mean(lot[1:2,i])

   std_r[i] = stddev(r[1:2,i])
   std_lot[i] = stddev(lot[1:2,i])

endif else begin

if (rrr_matrix[0,i] eq 0.) && (rrr_matrix[1,i] gt 0.) && $
   (rrr_matrix[2,i] eq 0.) && (rrr_matrix[3,i] gt 0.) && $
   (rrr_matrix[4,i] eq 0.) then begin

   r[j,i] = rrr_matrix[j,i]
   lot[j,i] = lot_matrix[j,i]

   mean_r[i] = mean([r[1,i],r[3,i]])
   mean_lot[i] = mean([lot[1,i],lot[3,i]])

   std_r[i] = stddev([r[1,i],r[3,i]])
   std_lot[i] = stddev([lot[1,i],lot[3,i]])

endif else begin

if (rrr_matrix[0,i] eq 0.) && (rrr_matrix[1,i] gt 0.) && $
   (rrr_matrix[2,i] eq 0.) && (rrr_matrix[3,i] eq 0.) && $
   (rrr_matrix[4,i] gt 0.) then begin

   r[j,i] = rrr_matrix[j,i]
```

```
      lot[j,i] = lot_matrix[j,i]

      mean_r[i] = mean([r[1,i],r[4,i]])
      mean_lot[i] = mean([lot[1,i],lot[4,i]])

      std_r[i] = stddev([r[1,i],r[4,i]])
      std_lot[i] = stddev([lot[1,i],lot[4,i]])

   endif else begin

   if (rrr_matrix[0,i] eq 0.) && (rrr_matrix[1,i] eq 0.) && $
      (rrr_matrix[2,i] gt 0.) && (rrr_matrix[3,i] gt 0.) && $
      (rrr_matrix[4,i] eq 0.) then begin

      r[j,i] = rrr_matrix[j,i]
      lot[j,i] = lot_matrix[j,i]

      mean_r[i] = mean(r[2:3,i])
      mean_lot[i] = mean(lot[2:3,i])

      std_r[i] = stddev(r[2:3,i])
      std_lot[i] = stddev(lot[2:3,i])

   endif else begin

   if (rrr_matrix[0,i] eq 0.) && (rrr_matrix[1,i] eq 0.) && $
      (rrr_matrix[2,i] gt 0.) && (rrr_matrix[3,i] eq 0.) && $
      (rrr_matrix[4,i] gt 0.) then begin

      r[j,i] = rrr_matrix[j,i]
      lot[j,i] = lot_matrix[j,i]

      mean_r[i] = mean([r[2,i],r[4,i]])
      mean_lot[i] = mean([lot[2,i],lot[4,i]])

      std_r[i] = stddev([r[2,i],r[4,i]])
      std_lot[i] = stddev([lot[2,i],lot[4,i]])

   endif else begin

   if (rrr_matrix[0,i] eq 0.) && (rrr_matrix[1,i] eq 0.) && $
      (rrr_matrix[2,i] eq 0.) && (rrr_matrix[3,i] gt 0.) && $
      (rrr_matrix[4,i] gt 0.) then begin

      r[j,i] = rrr_matrix[j,i]
      lot[j,i] = lot_matrix[j,i]
```

```
            mean_r[i] = mean(r[3:4,i])
            mean_lot[i] = mean(lot[3:4,i])

            std_r[i] = stddev(r[3:4,i])
            std_lot[i] = stddev(lot[3:4,i])

         endif else begin


; FLAG = 1 (only 1 model maps the points)
;---------

         if (rrr_matrix[0,i] gt 0.) && (rrr_matrix[1,i] eq 0.) && $
            (rrr_matrix[2,i] eq 0.) && (rrr_matrix[3,i] eq 0.) && $
            (rrr_matrix[4,i] eq 0.) then begin

            r[j,i] = rrr_matrix[j,i]
            lot[j,i] = lot_matrix[j,i]

            mean_r[i] = r[0,i]
            mean_lot[i] = lot[0,i]

            std_r[i] = stddev(r[0,i])
            std_lot[i] = stddev(lot[0,i])

         endif else begin

         if (rrr_matrix[0,i] eq 0.) && (rrr_matrix[1,i] gt 0.) && $
            (rrr_matrix[2,i] eq 0.) && (rrr_matrix[3,i] eq 0.) && $
            (rrr_matrix[4,i] eq 0.) then begin

            r[j,i] = rrr_matrix[j,i]
            lot[j,i] = lot_matrix[j,i]

            mean_r[i] = r[1,i]
            mean_lot[i] = lot[1,i]

            std_r[i] = stddev(r[1,i])
            std_lot[i] = stddev(lot[1,i])

         endif else begin

         if (rrr_matrix[0,i] eq 0.) && (rrr_matrix[1,i] eq 0.) && $
            (rrr_matrix[2,i] gt 0.) && (rrr_matrix[3,i] eq 0.) && $
            (rrr_matrix[4,i] eq 0.) then begin

            r[j,i] = rrr_matrix[j,i]
```

```
            lot[j,i] = lot_matrix[j,i]

            mean_r[i] = r[2,i]
            mean_lot[i] = lot[2,i]

            std_r[i] = stddev(r[2,i])
            std_lot[i] = stddev(lot[2,i])

         endif else begin

         if (rrr_matrix[0,i] eq 0.) && (rrr_matrix[1,i] eq 0.) && $
            (rrr_matrix[2,i] eq 0.) && (rrr_matrix[3,i] gt 0.) && $
            (rrr_matrix[4,i] eq 0.) then begin

            r[j,i] = rrr_matrix[j,i]
            lot[j,i] = lot_matrix[j,i]

            mean_r[i] = r[3,i]
            mean_lot[i] = lot[3,i]

            std_r[i] = stddev(r[3,i])
            std_lot[i] = stddev(lot[3,i])

            endif else begin

         if (rrr_matrix[0,i] eq 0.) && (rrr_matrix[1,i] eq 0.) && $
            (rrr_matrix[2,i] eq 0.) && (rrr_matrix[3,i] eq 0.) && $
            (rrr_matrix[4,i] gt 0.) then begin

            r[j,i] = rrr_matrix[j,i]
            lot[j,i] = lot_matrix[j,i]

            mean_r[i] = r[4,i]
            mean_lot[i] = lot[4,i]
            ;print,'£££',i,mean_r(i)

            std_r[i] = stddev(r[4,i])
            std_lot[i] = stddev(lot[4,i])

         endif else begin

; FLAG = 5 (all the models (5) map the points)
;---------

         if (rrr_matrix[j,i] ne 0.) then begin

            r[j,i] = rrr_matrix[j,i]
```

```
        lot[j,i] = lot_matrix[j,i]

        mean_r[i] = mean(r[*,i])
        mean_lot[i] = mean(lot[*,i])

        std_r[i] = stddev(r[*,i])
        std_lot[i] = stddev(lot[*,i])

    endif

endelse
endelse
endelse
endelse
endelse
endelse
endelse
endelse
endelse
endelse
endelse
endelse
endelse
endelse
endelse
endelse
endelse
endelse
endelse
endelse
endelse
endelse
endelse
endelse
endelse
endelse
endelse
endelse
endelse
endelse
endelse

endfor

if (mean_r[i] gt 0.) then begin
    flag[i] = total(r[*,i])/mean_r[i]
    printf,10, format='(f9.4,x,f9.4,x,f9.4,x,f9.4,x,f11.4,x,f6.0)',$
      mean_r[i], std_r[i], mean_lot[i], std_lot[i], cov[i], flag[i]
```

```
    endif

endfor

end

close,10
```

B.4 plot_data_mgnp_ipc

```
; Irene Pardo Cantos

; PROGRAM TO PLOT DATA AND OPLOT MAGNETOPAUSE

;-----------------------------------------------

; Joy's parameters assume that all distances are given in Rj/120.

Pd=0.044 ;nPa ; Joy expanded magnetopause(90 Rj)

Pd=0.37 ;nPa ; Joy compressed magnetopause(60 Rj)

Pd_4=Pd^(-0.25)

A=-0.134+0.488*Pd_4
B=-0.581-0.225*Pd_4
C=-0.186-0.016*Pd_4
D=-0.014+0.096*Pd
E=-0.814-0.811*Pd
F=-0.05+0.168*Pd

x=(findgen(400)-200.0)/120.

alpha=D+F*x
beta=sqrt(alpha*alpha-4.*E*(A+B*x+C*x*x))
gamma=2.*E

y1=(-alpha+beta)/gamma*120.
y2=(-alpha-beta)/gamma*120.

; PLOT MAGNETOPAUSE
;-------------------
cd, '/home/LPAP/pardo/'
set_dev   ; call set_dev.pro
```

```
loadct,39  ; Color Table "Rainbow + White"
device,/color

aa = findgen(360)*!dtor       ; ANGLE
rr = replicate(100,360)         ; RADIUS

x_rect = rr*cos(aa)     ; Rectangular (cartesian) coordinates
y_rect = rr*sin(aa)

window, 1, xs=1024, ys=1024

plot,y1,-x*120.,xrange=[-130,130],yrange=[-130,130], xtitle='12LT',$
     ytitle='6LT', xstyle=8, ystyle=8,linestyle=2,/isotropic,$
     background=255,color=0, xmargin=[10,10], ymargin=[6,6],$
     charsize=1.5, thick=3
oplot,y2,-x*120.,linestyle=2,color=0, thick=3

axis, xaxis=1, xtitle='0LT', color=0, xrange=[-130,130], charsize=1.5
axis, yaxis=1, ytitle='18LT', color=0, yrange=[-130,130], charsize=1.5

; READ DATA
;-----------
;rootname = 'od8k57itq'
;rootname = 'od8k61lyq'
rootname = 'od8k24ioq'

filename = '/home/pardo/results/' + rootname + '_mean_stdev_flag_all5.txt'
openr,20,filename

.run

mean_r = 0.
mean_lot = 0.

std_r = 0.
std_lot = 0.
cov = 0.
flag = 0.

lines = file_lines(filename)

mean_rr = fltarr(lines-2)
mean_lot_rad = fltarr(lines-2)
lot_rad = fltarr(lines-2)
std_rr = fltarr(lines-2)
std_lott = fltarr(lines-2)
std_lot_rad = fltarr(lines-2)
```

```
std_lot_radd = fltarr(lines-2)
covar = fltarr(lines-2)
flagg = fltarr(lines-2)

i=0

skip_lun,20,2,/lines

while ~ eof(20) do begin

      readf,20, format='(f9.4,x,f9.4,x,f9.4,x,f9.4,x,f11.4,x,f6.0)',$
                        mean_r, std_r, mean_lot, std_lot, cov, flag

      mean_rr[i] = mean_r

      mean_lot_rad[i] = mean_lot/24.*360.*!dtor
      lot_rad[i] = mean_lot_rad[i] + !pi/2.

      std_rr[i] = std_r
      std_lott[i] = std_lot
      ;std_lot_rad[i] = std_lot/24.*360.*!dtor
      ;std_lot_radd[i] = std_lot_rad[i] + !pi/2.

      covar[i] = cov

      flagg[i] = flag
      i+=1

endwhile

end

close,20

; PLOT DATA
;-----------
s = size(flagg)
length = s[1]

.run

mean_rr_1 = fltarr(length)
lot_rad_1 = fltarr(length)

mean_rr_2 = fltarr(length)
lot_rad_2 = fltarr(length)
```

```
mean_rr_3 = fltarr(length)
lot_rad_3 = fltarr(length)

mean_rr_4 = fltarr(length)
lot_rad_4 = fltarr(length)

mean_rr_5 = fltarr(length)
lot_rad_5 = fltarr(length)

for i=0,length-1 do begin
    if (flagg[i] eq 1.) then begin
       mean_rr_1[i] = mean_rr[i]
       lot_rad_1[i] = lot_rad[i]
    endif else begin

    if (flagg[i] eq 2.) then begin
       mean_rr_2[i] = mean_rr[i]
       lot_rad_2[i] = lot_rad[i]
    endif else begin

    if (flagg[i] eq 3.) then begin
       mean_rr_3[i] = mean_rr[i]
       lot_rad_3[i] = lot_rad[i]
    endif else begin

    if (flagg[i] eq 4.) then begin
       mean_rr_4[i] = mean_rr[i]
       lot_rad_4[i] = lot_rad[i]
    endif else begin

    if (flagg[i] eq 5.) then begin
       mean_rr_5[i] = mean_rr[i]
       lot_rad_5[i] = lot_rad[i]
    endif

    endelse
    endelse
    endelse
    endelse

endfor

end


close,20
```

```
;----------------------------

x_1 = mean_rr_1*cos(lot_rad_1)
x_2 = mean_rr_2*cos(lot_rad_2)
x_3 = mean_rr_3*cos(lot_rad_3)
x_4 = mean_rr_4*cos(lot_rad_4)
x_5 = mean_rr_5*cos(lot_rad_5)

y_1 = mean_rr_1*sin(lot_rad_1)
y_2 = mean_rr_2*sin(lot_rad_2)
y_3 = mean_rr_3*sin(lot_rad_3)
y_4 = mean_rr_4*sin(lot_rad_4)
y_5 = mean_rr_5*sin(lot_rad_5)

oplot, x_1, y_1, psym=1, color=250   ; FLAG=1 ; red
oplot, x_2, y_2, psym=1, color=70    ; FLAG=2 ; blue
oplot, x_3, y_3, psym=1, color=140   ; FLAG=3 ; green
oplot, x_4, y_4, psym=1, color=90    ; FLAG=4 ; cyan
oplot, x_5, y_5, psym=1, color=205   ; FLAG=5 ; orange

; PLOT CIRCLES (distances Rj)
;--------------
oplot,x_rect/100.,y_rect/100.,thick=3, color=0  ; Jupiter at the centre
oplot,0.2*x_rect,0.2*y_rect, color=0
oplot,0.4*x_rect,0.4*y_rect, color=0
oplot,0.6*x_rect,0.6*y_rect, color=0
oplot,0.8*x_rect,0.8*y_rect, color=0
oplot,1.2*x_rect,1.2*y_rect, color=0
oplot,x_rect,y_rect, color=0
```

B.5 plot_error_bars_ipc

```
; Irene Pardo Cantos

; PROGRAM TO SELECT THE BEST POINTS TO PLOT ERROR BARS ON
; PLOT THESE POINTS AND THEIR ERROR BARS
; These points are where:
;      - mean(RD) is maximum
;      - mean(RD) is minimum
;      - mean(LT) is maximum
;      - mean(LT) is minimum
;      - stdev(RD) is maximum
;      - stdev(RD) is minimum
```

```
;     - stdev(LT) is maximum
;     - stdev(LT) is minimum
;     - diff_RD is maximum; diff_RD = mean(RD) - stdev(RD)
;     - diff_RD is minimum
;     - diff_LT is maximum; diff_LT = mean(LT) - stdev(LT)
;     - diff_LT is minimum

;-------------------------------------------------------

; PLOT CIRCLES
;--------------

cd, '/home/LPAP/pardo/'
set_dev        ; call set_dev.pro
loadct,39      ; Color Table "Rainbow + White"
device,/color

aa = findgen(360)*!dtor            ; ANGLE
rr = replicate(100,360)            ; RADIUS

window, 1, xs=1024, ys=1024

plot, rr, aa, xrange=[-130,130],yrange=[-130,130], xtitle='12LT',$
     ytitle='6LT', xstyle=8, ystyle=8, xmargin=[13,13], ymargin=[8,4],$
     background=255,color=0,/polar,/isotropic,xcharsize=1.5,ycharsize=1.5
axis, xaxis=1, xtitle='0LT', color=0, xrange=[-130,130], xcharsize=1.5
axis, yaxis=1, ytitle='18LT', color=0, yrange=[-130,130], ycharsize=1.5

oplot, rr/100, aa, /polar, thick=3, color=0    ; Jupiter at the centre
oplot, 0.2*rr, aa, /polar, color=0
oplot, 0.4*rr, aa, /polar, color=0
oplot, 0.6*rr, aa, /polar, color=0
oplot, 0.8*rr, aa, /polar, color=0
oplot, 1.2*rr, aa, /polar, color=0

; PLOT MAGNETOPAUSE
;-------------------

Pd = 0.044 ;nPa ;Joy,2002 expanded magnetopause (90 Rj)
Pd = 0.37 ;nPa ;Joy,2002 compressed magnetopause (60 Rj)
Pd_4 = Pd^(-0.25)

A = -0.134 + 0.488*Pd_4
B = -0.581 - 0.225*Pd_4
C = -0.186 - 0.016*Pd_4
D = -0.014 + 0.096*Pd
E = -0.814 - 0.811*Pd
```

```
F = -0.05 + 0.168*Pd

x = (findgen(400)-200.0)/120.

alpha = D+F*x
beta = sqrt(alpha*alpha-4.*E*(A+B*x+C*x*x))
gamma = 2.*E

y1 = (-alpha+beta)/gamma*120.
y2 = (-alpha-beta)/gamma*120.

oplot,y1,-x*120.,linestyle=2,color=0, thick=3
oplot,y2,-x*120.,linestyle=2,color=0, thick=3

; READ DATA
;-----------

;rootname = 'od8k57itq'
;rootname = 'od8k61lyq'
rootname = 'od8k24ioq'

filename = '/home/pardo/results/' + rootname + '_mean_stdev_flag_all5.txt'

lines = file_lines(filename)

openr,1,filename

.run

mean_r = 0.
mean_lot = 0.

std_r = 0.
std_lot = 0.
cov = 0.
flag = 0.

mean_rr = fltarr(lines-2)
mean_lot_rad = fltarr(lines-2)
lot_rad = fltarr(lines-2)
std_rr = fltarr(lines-2)
std_lott = fltarr(lines-2)
std_lot_rad = fltarr(lines-2)
std_lot_radd = fltarr(lines-2)
covar = fltarr(lines-2)
flagg = fltarr(lines-2)
```

```
i=0

skip_lun,1,2,/lines

while ~ eof(1) do begin

      readf,1, format='(f9.4,x,f9.4,x,f9.4,x,f9.4,x,f11.4,x,f6.0)',$
               mean_r, std_r, mean_lot, std_lot, cov, flag

      mean_rr[i] = mean_r

      mean_lot_rad[i] = mean_lot/24.*360.*!dtor       ; in radians
      lot_rad[i] = mean_lot_rad[i] + !pi/2.

      std_rr[i] = std_r

      std_lott[i] = std_lot
      std_lot_rad[i] = std_lot/24.*360.*!dtor         ; in radians
      std_lot_radd[i] = std_lot_rad[i] + !pi/2.


      covar[i] = cov

      flagg[i] = flag
      i+=1

endwhile

end

close,1

; WHERE THERE ARE ZEROS
;----------------------

w_positives = where(mean_rr ne 0., complement=w_zeros)

mean_rad = mean_rr[w_positives]  ; keep only mean values greater than 0.
std_rad = std_rr[w_positives]

; SELECT THE POINTS OF INTEREST
;------------------------------

max_r = max(mean_rad)
min_r = min(mean_rad)
pos_max_r = where(mean_rr eq max_r)
pos_min_r = where(mean_rr eq min_r)
```

```
print, 'max_r = ', max_r, '  at i = ', pos_max_r
print, 'min_r = ', min_r, '  at i = ', pos_min_r

diff_r = mean_rad - std_rad
diff_rr = mean_rr - std_rr
max_diff_r = max(diff_r)
min_diff_r = min(diff_r)
pos_max_diff_r = where(diff_rr eq max_diff_r)
pos_min_diff_r = where(diff_rr eq min_diff_r)

print, 'max_diff_r = ', max_diff_r, '  at i = ', pos_max_diff_r
print, 'min_diff_r = ', min_diff_r, '  at i = ', pos_min_diff_r

;------------

mean_time = lot_rad[w_positives]
std_time = std_lott[w_positives]

max_lot = max(mean_time)
min_lot = min(mean_time)
pos_max_lot = where(lot_rad eq max_lot)
pos_min_lot = where(lot_rad eq min_lot)

print, 'max_lot = ', max_lot, '  at i = ', pos_max_lot
print, 'min_lot = ', min_lot, '  at i = ', pos_min_lot

diff_lot = mean_time - std_time
diff_lott = lot_rad - std_lott
max_diff_lot = max(diff_lot)
min_diff_lot = min(diff_lot)
pos_max_diff_lot = where(diff_lott eq max_diff_lot)
pos_min_diff_lot = where(diff_lott eq min_diff_lot)

print, 'max_diff_lot = ', max_diff_lot, '  at i = ', pos_max_diff_lot
print, 'min_diff_lot = ', min_diff_lot, '  at i = ', pos_min_diff_lot

;------------

std_rad = std_rr[w_positives]
std_time = std_lott[w_positives]

max_std_r = max(std_rad)
min_std_r = min(std_rad)
max_std_lt = max(std_time)
min_std_lt = min(std_time)
```

```
pos_max_std_r = where(std_rr eq max_std_r)
pos_min_std_r = where(std_rr eq min_std_r)
pos_max_std_lt = where(std_lott eq max_std_lt)
pos_min_std_lt = where(std_lott eq min_std_lt)

print, 'max_std_r = ', max_std_r, '  at i = ', pos_max_std_r
print, 'min_std_r = ', min_std_r, '  at i = ', pos_min_std_r
print, 'max_std_lt = ', max_std_lt, '  at i = ', pos_max_std_lt
print, 'min_std_lt = ', min_std_lt, '  at i = ', pos_min_std_lt

; PLOT SELECTED DATA + ERROR BARS
;-------------------------------

r1 = mean_rr[pos_max_r] + std_rr[pos_max_r]
r2 = mean_rr[pos_max_r] - std_rr[pos_max_r]

lt0 = lot_rad[pos_max_r]
lt1 = lot_rad[pos_max_r] + std_lot_rad[pos_max_r]
lt2 = lot_rad[pos_max_r] - std_lot_rad[pos_max_r]

oplot, [r1,r2], [lot_rad[pos_max_r],lot_rad[pos_max_r]], /polar,$
        color=250, thick=3           ; red
oplot, [mean_rr[pos_max_r],mean_rr[pos_max_r],mean_rr[pos_max_r]],$
        [lt1,lt0,lt2], /polar, color=250, thick=3       ; red

;--------

r1 = mean_rr[pos_min_r] + std_rr[pos_min_r]
r2 = mean_rr[pos_min_r] - std_rr[pos_min_r]

lt0 = lot_rad[pos_min_r]
lt1 = lot_rad[pos_min_r] + std_lot_rad[pos_min_r]
lt2 = lot_rad[pos_min_r] - std_lot_rad[pos_min_r]

oplot, [r1,r2], [lot_rad[pos_min_r],lot_rad[pos_min_r]], /polar,$
        color=250, thick=3             ; red
oplot, [mean_rr[pos_min_r],mean_rr[pos_min_r],mean_rr[pos_min_r]],$
        [lt1,lt0,lt2], /polar, color=250, thick=3       ; red

;--------

r1 = mean_rr[pos_max_lot] + std_rr[pos_max_lot]
r2 = mean_rr[pos_max_lot] - std_rr[pos_max_lot]

lt0 = lot_rad[pos_max_lot]
lt1 = lot_rad[pos_max_lot] + std_lot_rad[pos_max_lot]
lt2 = lot_rad[pos_max_lot] - std_lot_rad[pos_max_lot]
```

```
lt11 = lot_rad[pos_max_lot] + std_lot_rad[pos_max_lot] - 0.1
lt22 = lot_rad[pos_max_lot] - std_lot_rad[pos_max_lot] + 0.1

r_arr = [mean_rr[pos_max_lot],mean_rr[pos_max_lot],mean_rr[pos_max_lot],$
          mean_rr[pos_max_lot],mean_rr[pos_max_lot]]
lt_arr = [lt1,lt11,lt0,lt22,lt2]

oplot, [r1,r2], [lot_rad[pos_max_lot],lot_rad[pos_max_lot]], /polar,$
        color=140, thick=3        ; green
oplot, r_arr, lt_arr, /polar, color=140, thick=3         ; green

;--------

r1 = mean_rr[pos_min_lot] + std_rr[pos_min_lot]
r2 = mean_rr[pos_min_lot] - std_rr[pos_min_lot]

lt0 = lot_rad[pos_min_lot]
lt1 = lot_rad[pos_min_lot] + std_lot_rad[pos_min_lot]
lt2 = lot_rad[pos_min_lot] - std_lot_rad[pos_min_lot]

oplot, [r1,r2], [lot_rad[pos_min_lot],lot_rad[pos_min_lot]], /polar,$
        color=140, thick=3          ; green
oplot, [mean_rr[pos_min_lot],mean_rr[pos_min_lot],mean_rr[pos_min_lot]],$
        [lt1,lt0,lt2], /polar, color=140, thick=3        ; green

;--------

r1 = mean_rr[pos_max_diff_r] + std_rr[pos_max_diff_r]
r2 = mean_rr[pos_max_diff_r] - std_rr[pos_max_diff_r]

lt0 = lot_rad[pos_max_diff_r]
lt1 = lot_rad[pos_max_diff_r] + std_lot_rad[pos_max_diff_r]
lt2 = lot_rad[pos_max_diff_r] - std_lot_rad[pos_max_diff_r]

oplot, [r1,r2], [lot_rad[pos_max_diff_r],lot_rad[pos_max_diff_r]],/polar,$
        color=205, thick=3       ; orange
oplot, [mean_rr[pos_max_diff_r],mean_rr[pos_max_diff_r],$
        mean_rr[pos_max_diff_r]],[lt1,lt0,lt2],/polar,color=205,thick=3

;--------

r1 = mean_rr[pos_min_diff_r] + std_rr[pos_min_diff_r]
r2 = mean_rr[pos_min_diff_r] - std_rr[pos_min_diff_r]

lt0 = lot_rad[pos_min_diff_r]
lt1 = lot_rad[pos_min_diff_r] + std_lot_rad[pos_min_diff_r]
lt2 = lot_rad[pos_min_diff_r] - std_lot_rad[pos_min_diff_r]
```

```
oplot, [r1,r2], [lot_rad[pos_min_diff_r],lot_rad[pos_min_diff_r]],$
        /polar, color=205, thick=3      ; orange
oplot, [mean_rr[pos_min_diff_r],mean_rr[pos_min_diff_r],$
        mean_rr[pos_min_diff_r]], [lt1,lt0,lt2],/polar,color=205,thick=3

;--------

r1 = mean_rr[pos_max_diff_lot] + std_rr[pos_max_diff_lot]
r2 = mean_rr[pos_max_diff_lot] - std_rr[pos_max_diff_lot]

lt0 = lot_rad[pos_max_diff_lot]
lt1 = lot_rad[pos_max_diff_lot] + std_lot_rad[pos_max_diff_lot]
lt2 = lot_rad[pos_max_diff_lot] - std_lot_rad[pos_max_diff_lot]

oplot, [r1,r2], [lot_rad[pos_max_diff_lot],lot_rad[pos_max_diff_lot]],$
         /polar, color=70, thick=3         ; blue
oplot, [mean_rr[pos_max_diff_lot],mean_rr[pos_max_diff_lot],$
        mean_rr[pos_max_diff_lot]], [lt1,lt0,lt2],/polar,color=70,thick=3

;--------

r1 = mean_rr[pos_min_diff_lot] + std_rr[pos_min_diff_lot]
r2 = mean_rr[pos_min_diff_lot] - std_rr[pos_min_diff_lot]

lt0 = lot_rad[pos_min_diff_lot]
lt1 = lot_rad[pos_min_diff_lot] + std_lot_rad[pos_min_diff_lot]
lt2 = lot_rad[pos_min_diff_lot] - std_lot_rad[pos_min_diff_lot]
lt11 = lot_rad[pos_min_diff_lot] + std_lot_rad[pos_min_diff_lot] - 0.1
lt22 = lot_rad[pos_min_diff_lot] - std_lot_rad[pos_min_diff_lot] + 0.1

r_arr = [mean_rr[pos_min_diff_lot],mean_rr[pos_min_diff_lot],$
          mean_rr[pos_min_diff_lot],mean_rr[pos_min_diff_lot],$
          mean_rr[pos_min_diff_lot]]
lt_arr = [lt1,lt11,lt0,lt22,lt2]

oplot, [r1,r2], [lot_rad[pos_min_diff_lot],lot_rad[pos_min_diff_lot]],$
        /polar, color=70, thick=3       ; blue
oplot, r_arr, lt_arr, /polar, color=70, thick=3         ; blue

;--------

r1 = mean_rr[pos_max_std_r] + std_rr[pos_max_std_r]
r2 = mean_rr[pos_max_std_r] - std_rr[pos_max_std_r]

lt0 = lot_rad[pos_max_std_r]
lt1 = lot_rad[pos_max_std_r] + std_lot_rad[pos_max_std_r]
```

```
lt2 = lot_rad[pos_max_std_r] - std_lot_rad[pos_max_std_r]

oplot, [r1,r2], [lot_rad[pos_max_std_r],lot_rad[pos_max_std_r]], /polar,$
        color=95, thick=3       ; cyan
oplot, [mean_rr[pos_max_std_r],mean_rr[pos_max_std_r],$
        mean_rr[pos_max_std_r]], [lt1,lt0,lt2], /polar, color=95, thick=3

;--------

r1 = mean_rr[pos_min_std_r] + std_rr[pos_min_std_r]
r2 = mean_rr[pos_min_std_r] - std_rr[pos_min_std_r]

lt0 = lot_rad[pos_min_std_r]
lt1 = lot_rad[pos_min_std_r] + std_lot_rad[pos_min_std_r]
lt2 = lot_rad[pos_min_std_r] - std_lot_rad[pos_min_std_r]

oplot, [r1,r2], [lot_rad[pos_min_std_r],lot_rad[pos_min_std_r]], /polar,$
        color=95, thick=3       ; cyan
oplot, [mean_rr[pos_min_std_r],mean_rr[pos_min_std_r],$
        mean_rr[pos_min_std_r]], [lt1,lt0,lt2], /polar, color=95, thick=3

;--------

r1 = mean_rr[pos_max_std_lt] + std_rr[pos_max_std_lt]
r2 = mean_rr[pos_max_std_lt] - std_rr[pos_max_std_lt]

lt0 = lot_rad[pos_max_std_lt]
lt1 = lot_rad[pos_max_std_lt] + std_lot_rad[pos_max_std_lt]
lt2 = lot_rad[pos_max_std_lt] - std_lot_rad[pos_max_std_lt]
lt11 = lot_rad[pos_max_std_lt] + std_lot_rad[pos_max_std_lt] - 0.1
lt22 = lot_rad[pos_max_std_lt] - std_lot_rad[pos_max_std_lt] + 0.1

r_arr = [mean_rr[pos_max_std_lt],mean_rr[pos_max_std_lt],$
          mean_rr[pos_max_std_lt],mean_rr[pos_max_std_lt],$
          mean_rr[pos_max_std_lt]]
lt_arr = [lt1,lt11,lt0,lt22,lt2]

oplot, [r1,r2], [lot_rad[pos_max_std_lt],lot_rad[pos_max_std_lt]],$
        /polar, color=0, thick=3      ; black
oplot, r_arr, lt_arr, /polar, color=20, thick=3       ; black

;--------

r1 = mean_rr[pos_min_std_lt] + std_rr[pos_min_std_lt]
r2 = mean_rr[pos_min_std_lt] - std_rr[pos_min_std_lt]

lt0 = lot_rad[pos_min_std_lt]
```

```
lt1 = lot_rad[pos_min_std_lt] + std_lot_rad[pos_min_std_lt]
lt2 = lot_rad[pos_min_std_lt] - std_lot_rad[pos_min_std_lt]

oplot, [r1,r2], [lot_rad[pos_min_std_lt],lot_rad[pos_min_std_lt]],$
        /polar, color=0, thick=3    ; black
oplot, [mean_rr[pos_min_std_lt],mean_rr[pos_min_std_lt],$
        mean_rr[pos_min_std_lt]], [lt1,lt0,lt2], /polar, color=0, thick=3
```

B.6 Lomb_Periodogram_ipc

This last script was developed in RStudio.

```
library(sp)

# Read data
data = read.table(file.choose())   # Choose the file

# Define data
cml = data[,1]    # Central Meridian Longitude (CML)
power = data[,2]  # Power/Intensity (in W)

Pj = 9.92496  # Period of rotation in hours

time = cml*Pj*3600/360 - cml[1]*Pj*3600/360  # Time in seconds

plot(time, power, type="l", xlab="Time (s)", ylab="Emitted power (W)",
cex.lab=1.3, col=2)

datos <- data.frame(time,power)

# Lomb-Scargle analysis
library(lomb)

lomb <- lsp(datos,xlab='Period (s)',ylab='Lomb Normalized Periodogram',
                cex.lab=1.3,main=' ',times=NULL,type="period",from=40,
                to=1000,ofac=20,alpha=0.01,plot=TRUE)

summary(lomb)  # To obtain all the parameters
```