

High resolution (5-min)  
analysis of Sun-Earth  
coupling efficiency  
since 1981

Bart Olsthoorn

IRF Technical Report 055  
March 2016

ISSN 0284-1738

**INSTITUTET FÖR RYMDFYSIK**  

---

**Swedish Institute of Space Physics**

**Kiruna, Sweden**



# High resolution (5-min) analysis of Sun-Earth coupling efficiency since 1981

**Bart Olsthoorn**  
(supervised by M. Yamauchi)

**IRF technical report 055**  
August 2015

**INSTITUTET FÖR RYMDFYSIK**  
Swedish institute of space physics

Kiruna, Sweden

## **Abstract**

This report includes correlation results of the Sun-Earth coupling during cycle #22, #23 and the current cycle #24 using NASA's High Resolution OMNI (HRO) data.

# Contents

<b>1</b>	<b>Introduction</b>	<b>3</b>
<b>2</b>	<b>Coupling functions (<math>\epsilon'</math> and <math>d\Phi_{MP}/dt</math>)</b>	<b>4</b>
<b>3</b>	<b>High Resolution OMNI data set (HRO)</b>	<b>6</b>
<b>4</b>	<b>Coupling efficiency for varying solar wind input</b>	<b>7</b>
4.1	Correlation with AE, AL and AU (method A) . . . . .	7
4.2	Correlation with AE, AL and AU (method B) . . . . .	9
4.3	AE, AL and AU during pre- and post-solar maximum . . . . .	11
4.4	Dst-index . . . . .	15
<b>5</b>	<b>Coupling efficiency over long time (annual and decadal)</b>	<b>16</b>
<b>6</b>	<b>High input and big event data sets</b>	<b>17</b>
6.1	High Kp-index events . . . . .	17
6.2	High $\epsilon'$ events . . . . .	18
6.3	AE, AL and AU during high $\epsilon'$ and $d\Phi_{MP}/dt$ events . . . . .	19
<b>7</b>	<b>Conclusion</b>	<b>20</b>
<b>8</b>	<b>References</b>	<b>21</b>
<b>A</b>	<b>Largest geomagnetic storms since 1932 and 1910</b>	<b>22</b>
<b>B</b>	<b>Python code for key figures</b>	<b>25</b>
B.1	Ratio of correlation (cf. Figure 3) . . . . .	25
B.2	Scatter plot (cf. Figure 9) . . . . .	25
B.3	Time series (cf. Figure 11a) . . . . .	25
B.4	Big events (cf. Figure 13) . . . . .	25

# 1 Introduction

The work presented here is part of a summer project at IRF.

The geomagnetic field is affected by the solar wind energy and momentum transfer to the Earth's magnetosphere. This interaction finally drives electric currents in the ionosphere and the induction current at the ground, the last part of which are detected as the geomagnetic disturbances at the Earth's surface. These variations in the geomagnetic field due to such Sun-Earth coupling are measured by a global network of magnetometers. To visualize global level geomagnetic activities, a variety of geomagnetic indices has been introduced at low latitude (*Dst*), mid-latitude (*SYM* and *ASY*), subauroral region (*Kp*, *ap*, and *aa*), auroral region (*AE*) and the polar region (*PC*).

Since the geomagnetic disturbances are driven largely by the solar wind, the Sun-Earth coupling functions have been constructed [Perreault and Akasofu, 1978, Akasofu, 1981, Newell et al., 2007] to predict the global geomagnetic disturbance level (that are represented by the geomagnetic indices) using a scalar function  $\epsilon$ ,  $\epsilon'$ , etc. calculated from the solar wind parameters. However, a recent study using one-hour resolution data has found that the response of the geomagnetic activity to the same value of the coupling function (Sun-Earth coupling efficiency) during last 10 years since 2006 is significantly lower than those of the previous four cycles [Yamauchi, 2015]. This means that for the same solar wind conditions, less geomagnetic activity was observed during the current solar cycle #24 than the previous four cycles.

The decreased is valid only for low or moderate solar wind conditions ( $\epsilon' < 10^2 \text{ W/km}^2$ ), but not for active periods with  $\epsilon' > 5 \times 10^2 \text{ W/km}^2$  that is calculated from hourly values. Since the number of cases for such "strong" solar wind conditions are not very often, higher resolution data should be used to examine such "strong" cases. This report uses 5-minute resolution data to compare the current solar cycle #24 to the previous cycles #23 and #22 and compare results with the previous study. The result is the first step of improving nowcast of the geomagnetic activities. Furthermore, the result would be helpful in understanding the following questions that is relevant to the space weather:

1. Is the Sun-Earth coupling efficiency of cycle #24 at high-latitude during high solar wind input ( $\epsilon' > 5 \times 10^2 \text{ W/km}^2$ ) really higher than those of previous cycles, as the preliminary study suggests?
2. Is the declining phase singular year of the coupling efficiency seen in low-resolution data really singular in high-resolution data?

The analyses program made by Python code that reads the updated OMNI data. Some of the programs are attached in the appendix.

## 2 Coupling functions ( $\epsilon'$ and $d\Phi_{MP}/dt$ )

The scalar coupling function is calculated solely from the solar wind parameters to obtain a single parameter that correlates well with the magnetospheric conditions and particularly the geomagnetic indices. The simplest coupling function is just the northward component of interplanetary magnetic field  $B_z$ , and there are many other function forms that performs better [Newell et al., 2007] using  $B_z$ , solar wind velocity ( $v$ ), and density ( $n$ ).

As fore the representative of global geomagnetic and magnetospheric activities, Dst (low-latitude), Kp (mid-latitude), AE (high latitude), PC (high-latitude) indices and Cusp latitude are often used. The prediction accuracy are different between different sets of the coupling functions and geomagnetic indices [Newell et al., 2007].

In this report, two most commonly-used coupling functions are examined:

1. The Alasofu's epsilon [Akasofu, 1981],

$$\epsilon' = 4\pi \frac{v B_T^2}{\mu_0} \sin^4(\theta_c/2) \quad (1)$$

2. The Newell coupling function [Newell et al., 2007],

$$d\Phi_{MP}/dt = v^{4/3} B_T^{2/3} \sin^{8/3}(\theta_c/2) \quad (2)$$

where:  $v$  = flow speed [km/s]

$B_T$  = transverse component of IMF [nT] =  $\sqrt{B_y^2 + B_z^2}$

$\theta_c$  =  $\arctan(B_y/B_z)$

$B_y$  = y component of IMF (GSM) [nT]

$B_z$  = z component of IMF (GSM) [nT]

The parameters  $v$ ,  $B_y$  and  $B_z$  are directly available in the 5-minute resolution OMNI (HRO) data set at

[ftp://spdf.gsfc.nasa.gov/pub/data/omni/high\\_res\\_omni/omni\\_5min.asc](ftp://spdf.gsfc.nasa.gov/pub/data/omni/high_res_omni/omni_5min.asc)

The chosen description above matches with the column description given by NASA.

Here,  $\epsilon'$  is slightly different from original form of the Akasofu's epsilon [Perreault and Akasofu, 1978]:  $\epsilon'$  does not include the cross section but is multiplied by  $4\pi$  (i.e.,  $4\pi \times$  Poynting flux) to make a simple multiplication of  $1/\mu_0 = 10^7/4\pi$  [A<sup>2</sup>/N] to the raw values using km/s and nT. Therefore, the unit of  $\epsilon'$  is  $10^7$  [A<sup>2</sup>/N]  $\cdot 10^{-18}$  [T<sup>2</sup>]  $\cdot 10^3$  [m/s] =  $10^{-8}$  [W/m<sup>2</sup>] = [0.01 W/km<sup>2</sup>]; i.e., all numbers that are given by [nT<sup>2</sup>km/s] (which is use in this entire report) must be multiplied by 0.01 when converting to the [W/km<sup>2</sup>] unit. The unit of Newell's flux  $d\Phi_{MP}/dt$  is [nT<sup>2/3</sup>  $\cdot$  km<sup>4/3</sup>  $\cdot$  s<sup>-4/3</sup>] = [(mV/s)<sup>2/3</sup>] = [0.01 V<sup>2/3</sup>  $\cdot$  s<sup>-2/3</sup>].

Both coupling functions can also be written in alternative forms, eliminating the need of arctan and minimizing the number of operations:

$$\begin{aligned}
\epsilon' &= vB_T^2 (\sin^2(\theta_c/2))^2 \quad \text{substitute} \quad \sin^2(\theta_c/2) = \frac{1 - \cos(\theta_c)}{2} \\
&= vB_T^2 \left( \frac{1 - \cos(\theta_c)}{2} \right)^2 \\
&= vB_T^2 \frac{(1 - \cos(\theta_c))^2}{4} \quad \text{substitute} \quad \cos(\arctan(B_y/B_z)) = \frac{1}{\sqrt{\frac{B_y^2 + B_z^2}{B_z^2}}} = \frac{B_z}{B_T} \\
&= vB_T^2 \frac{\left(1 - \frac{B_z}{B_T}\right)^2}{4} \\
&= v \frac{(B_T - B_z)^2}{4} \\
d\Phi_{MP}/dt &= \left( v^2 B_T (\sin^2(\theta_c/2))^2 \right)^{\frac{2}{3}} \quad \text{substitute} \quad (\sin^2(\theta_c/2))^2 = \frac{\left(1 - \frac{B_z}{B_T}\right)^2}{4} \\
&= \left( v^2 B_T \frac{\left(1 - \frac{B_z}{B_T}\right)^2}{4} \right)^{\frac{2}{3}} \\
&= \left( v^2 \frac{(B_T - B_z)^2}{4B_T} \right)^{\frac{2}{3}}
\end{aligned}$$

For the data set used in this report, the statistics listed in Table 1 apply. More details about the data set will be discussed in Section 3. High values of  $\epsilon'$  and  $d\Phi_{MP}/dt$  are also referred to as *high solar wind input*.

	$\epsilon'$ [0.01 W/km <sup>2</sup> ]	$d\Phi_{MP}/dt$ [0.01 V <sup>2/3</sup> ·s <sup>-2/3</sup> ]
<b>mean</b>	$4.9 \times 10^3$	$3.9 \times 10^3$
<b>sd (<math>\sigma</math>)</b>	$2.0 \times 10^4$	$4.2 \times 10^3$
<b>min</b>	0	0
<b>25%</b>	$1.7 \times 10^2$	$8.6 \times 10^2$
<b>50%</b>	$1.2 \times 10^3$	$2.7 \times 10^3$
<b>75%</b>	$4.3 \times 10^3$	$5.5 \times 10^3$
<b>max</b>	$1.9 \times 10^6$	$1.1 \times 10^5$

**Table 1:** Statistics of the HRO data set, 1981–2015.



### 3 High Resolution OMNI data set (HRO)

OMNI data is spacecraft-interspersed, near-Earth solar wind data. The observations in the HRO dataset are time-shifted to the bow shock nose. The bow shock nose is about 14 Earth radii ( $R_E = 6371$  km) from the center of the Earth. The average flow speed in the HRO data set is  $434 \text{ km s}^{-1}$ . This means that on average, it would take the solar wind about 3 min to reach the Earth. However, the solar wind interacts with the magnetosphere and magnetotail with a complicated sequences before the energy reaches the Earth's surface. When using 5-minute resolution OMNI (HRO) data, the time lag would become unknown parameter. Two different time-lags will be used in this report (Section 4) but the integrated effect is not considered because that can largely be represented by 1-hour resolution data [Yamauchi , 2015]. The data coverage of the HRO data set used in this report is until 2015-06-18.

**Table 2:** The format of the High Resolution OMNI (HRO) data set. The bold entries are used in this report.

<b>Year</b>	I4	1995 ... 2006	
<b>Day</b>	I4	1 ... 365 or 366	
<b>Hour</b>	I3	0 ... 23	
<b>Minute</b>	I3	0 ... 59 at start of average	
ID for IMF spacecraft	I3		
ID for SW Plasma spacecraft	I3		
# of points in IMF averages	I4		
# of points in Plasma averages	I4		
Percent interp	I4		
Timeshift, sec	I7		
RMS, Timeshift	I7		
RMS, Phase front normal	F6.2		
Time btwn observations, sec	I7		
Field magnitude average, nT	F8.2		
Bx, nT (GSE, GSM)	F8.2		
By, nT (GSE)	F8.2		
Bz, nT (GSE)	F8.2		
<b>By, nT (GSM)</b>	F8.2	Determined from post-shift GSE components	
<b>Bz, nT (GSM)</b>	F8.2	Determined from post-shift GSE components	
RMS SD B scalar, nT	F8.2		
RMS SD field vector, nT	F8.2		
<b>Flow speed, km/s</b>	F8.1		
Vx Velocity, km/s, GSE	F8.1		
Vy Velocity, km/s, GSE	F8.1		
Vz Velocity, km/s, GSE	F8.1		
Proton Density, n/cc	F7.2		
Temperature, K	F9.0		
Flow pressure, nPa	F6.2		
Electric field, mV/m	F7.2		
Plasma beta	F7.2		
Alfven mach number	F6.1		
X(s/c), GSE, Re	F8.2		
Y(s/c), GSE, Re	F8.2		
Z(s/c), GSE, Re	F8.2		
BSN location, Xgse, Re	F8.2	BSN = bow shock nose	
BSN location, Ygse, Re	F8.2		
BSN location, Zgse, Re	F8.2		
<b>AE-index, nT</b>	I6		
<b>AL-index, nT</b>	I6		
<b>AU-index, nT</b>	I6		
SYM/D index, nT	I6		
SYM/H index, nT	I6		
ASY/D index, nT	I6		
ASY/H index, nT	I6		
PC(N) index,	F7.2		
Magnetosonic mach number	F5.1		

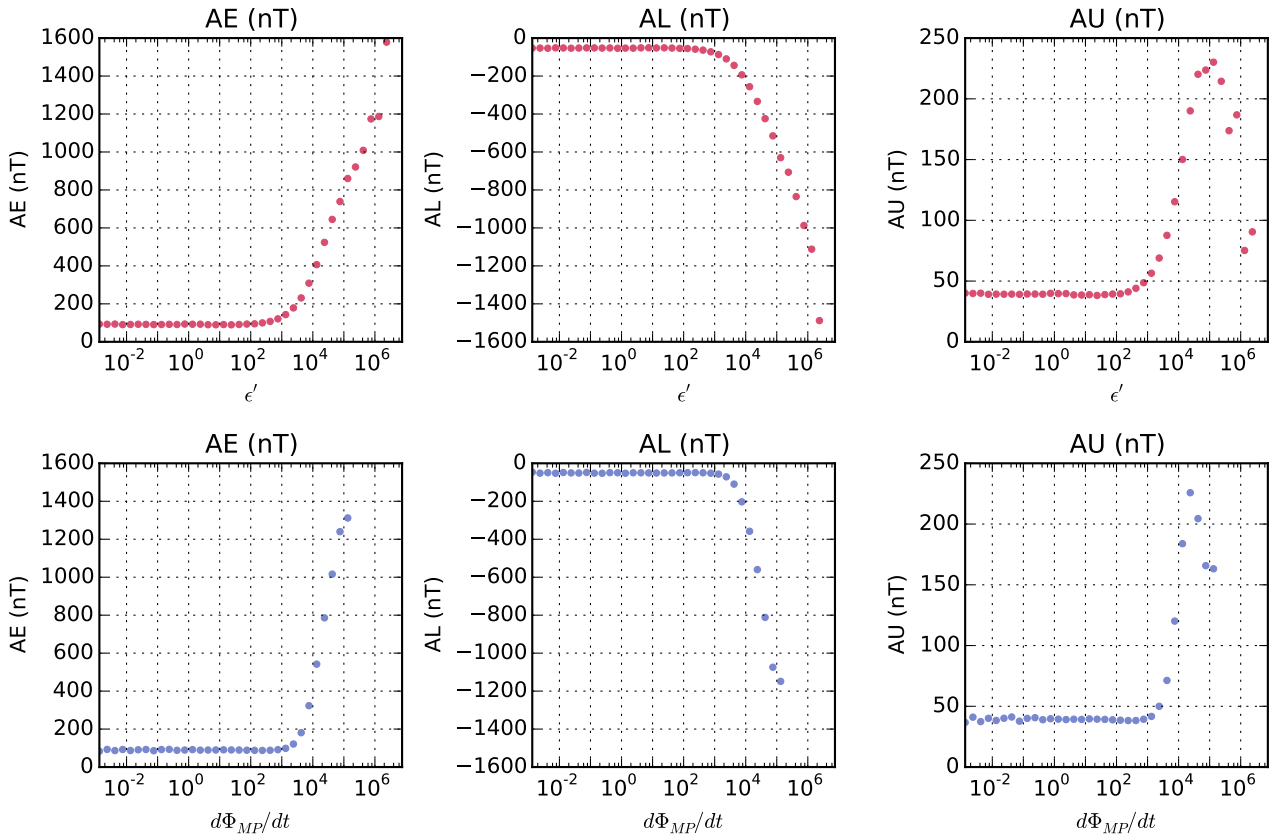
## 4 Coupling efficiency for varying solar wind input

The coupling functions ( $\epsilon'$  and  $d\Phi_{MP}/dt$ ) are calculated for the 5-min OMNI (HRO) dataset, and the correlation between these coupling functions and the geomagnetic indices will be obtained for different individual solar cycles. Two methods will be used:

- (A) The 5-minute resolution coupling parameters ( $\epsilon'$  and  $d\Phi_{MP}/dt$ ) are directly correlated with the AE, AL and AU indices of the next 5-minute interval (introducing a 5-minute delay).
- (B) For each 5-minute resolution coupling parameter, the highest AE, AL and AU is selected from the following hour (the highest AE, AL and AU do not have to occur at the same interval).

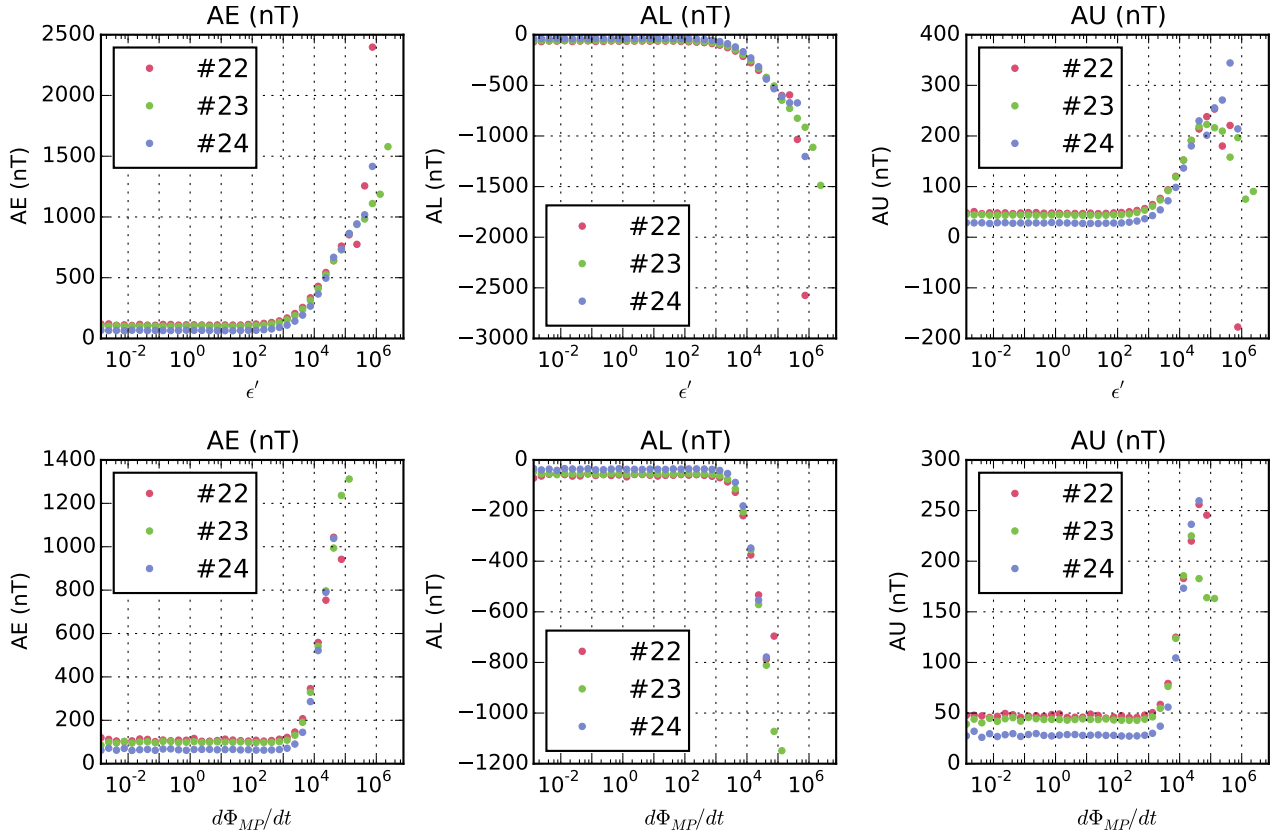
### 4.1 Correlation with AE, AL and AU (method A)

Correlations between the solar wind input (5-minute resolution) and the geomagnetic AE indices (AE, AL, and AU) are plotted over  $\epsilon' = 10^{-3} - 10^7 [\times 0.01 \text{ W/km}^2]$  in Figures 1 and 2. Figure 1 shows the correlation using entire dataset from 1981, whereas different solar cycles (#22, #23, and #24) are subdivided in Figure 2.

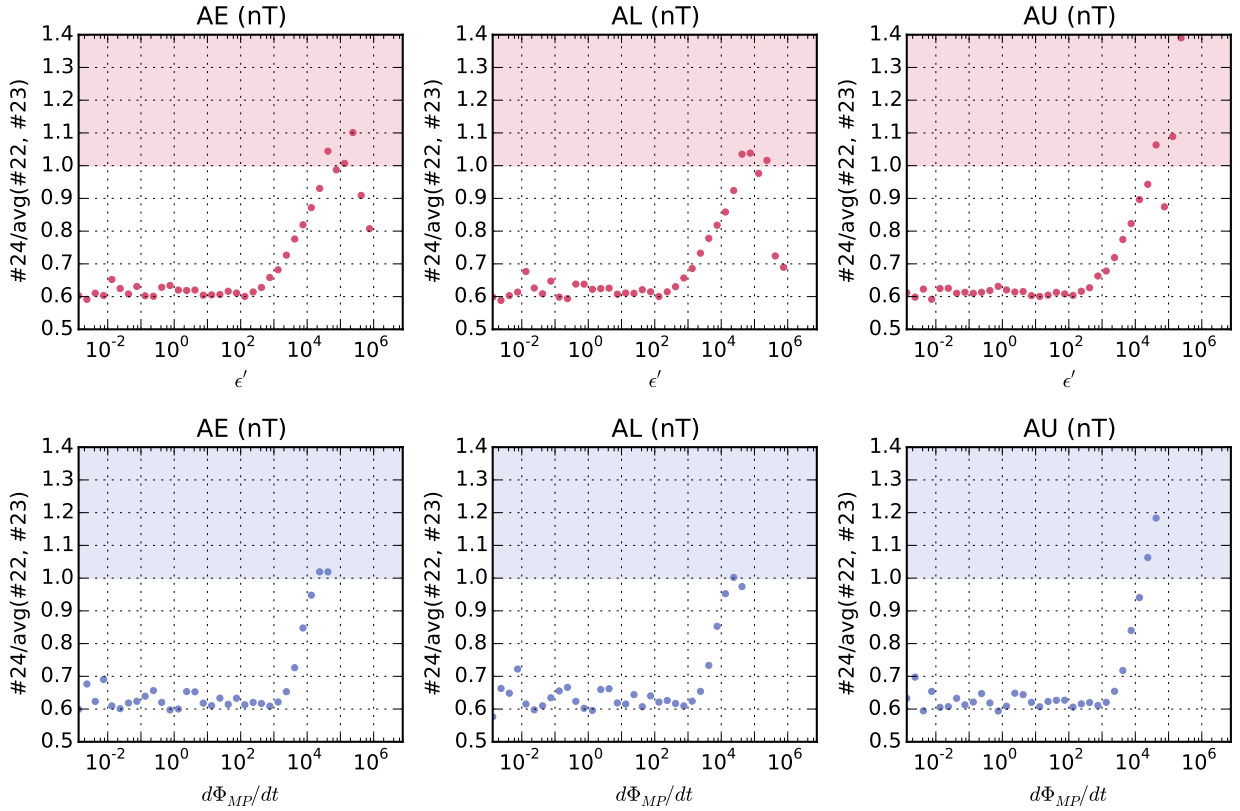


**Figure 1:** Correlations between the 5-minute resolution coupling functions and the 5-minute averaged geomagnetic AE indices (AE, AL, and AU) with 5-minute time lag (method A). The input values (x-axis) are divided into 40 evenly distributed bins.

Figure 2 indicates that the same low input ( $\epsilon' < 10^4 [\times 0.01 \text{ W/km}^2]$ ) leads to less AE activity for solar cycle #24 than cycle #23 or #22. However, the difference becomes smaller for  $\epsilon' > 10^3 [\times 0.01 \text{ W/km}^2]$  (Figure 3). Due to the small number of big events, it is uncertain what happens to this ratio around  $\epsilon' = 10^5 [\times 0.01 \text{ W/km}^2]$ , and this will be studied in more detail in a later section.



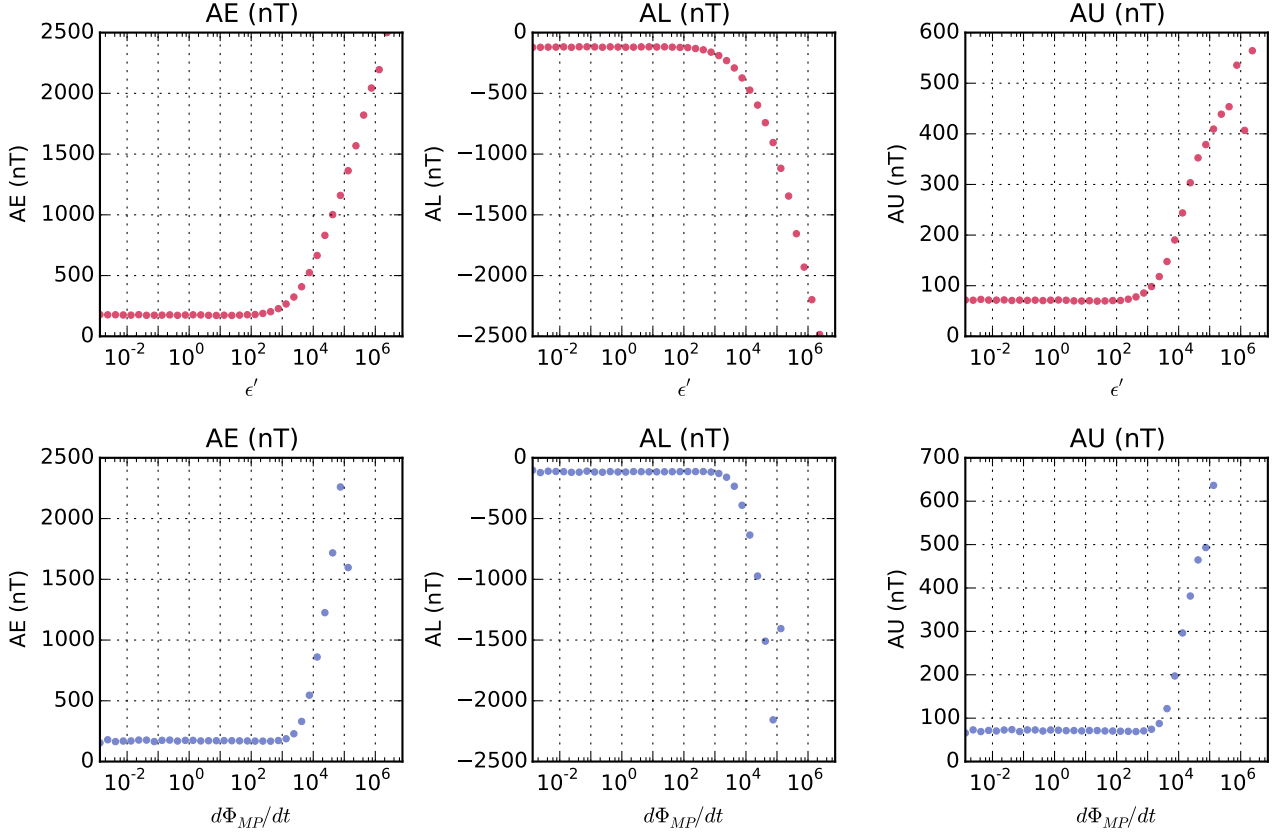
**Figure 2:** The same as Figure 1 but subdivided into different solar cycles.



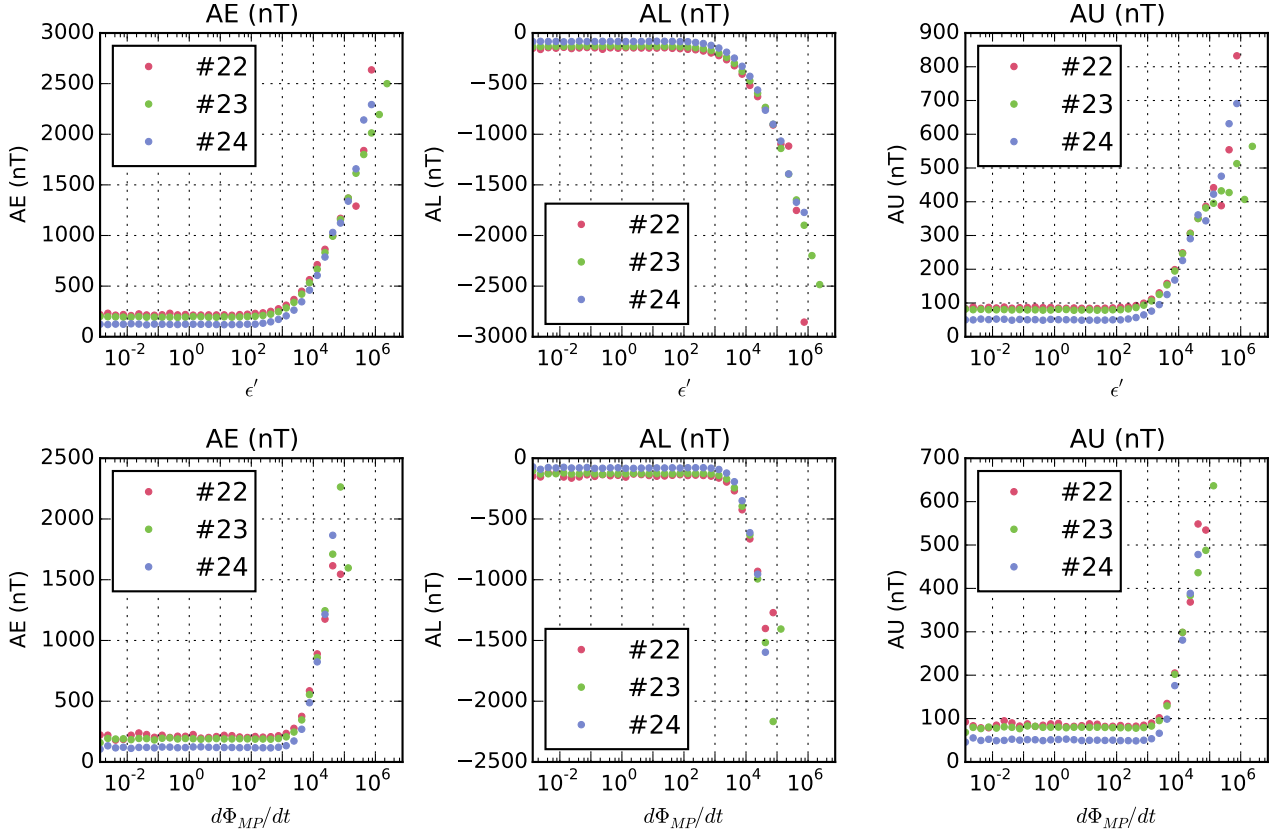
**Figure 3:** Cycle #24 divided by the average of #22 and #23 (method A).

## 4.2 Correlation with AE, AL and AU (method B)

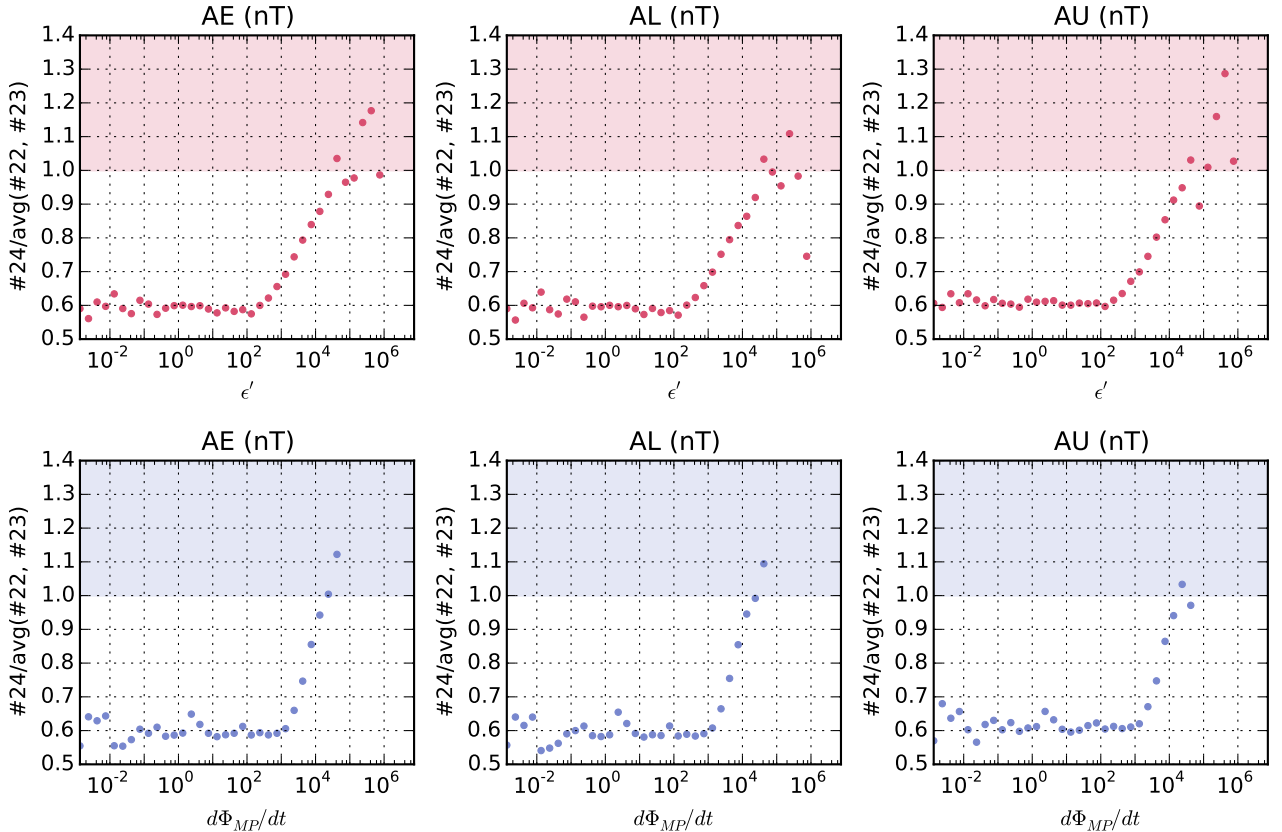
The previous analyses are repeated with method B instead of method A; i.e., for each 5-minute solar wind values, the highest 5-minute averaged AE, AL and AU during the following hour are correlated with  $\epsilon'$  and  $d\Phi_{MP}/dt$ . Except AU during high values of coupling functions, the results (Figures 4, 5, and 6) are similar to the previous section (method A).



**Figure 4:** Correlations between the 5-minute resolution coupling functions and maximum 5-minute averaged values of geomagnetic AE indices (AE, AL, and AU) within following 1 hour (method B). The input values (x-axis) are divided into 40 evenly distributed bins. The entire 5-minute resolution OMNI data are used (from 1981)



**Figure 5:** Correlations between the 5-minute resolution coupling functions and maximum 5-minute averaged values of geomagnetic AE indices (AE, AL, and AU) within following 1 hour (method B). The input values (x-axis) are divided into 40 evenly distributed bins, and data are subdivided into different solar cycles.

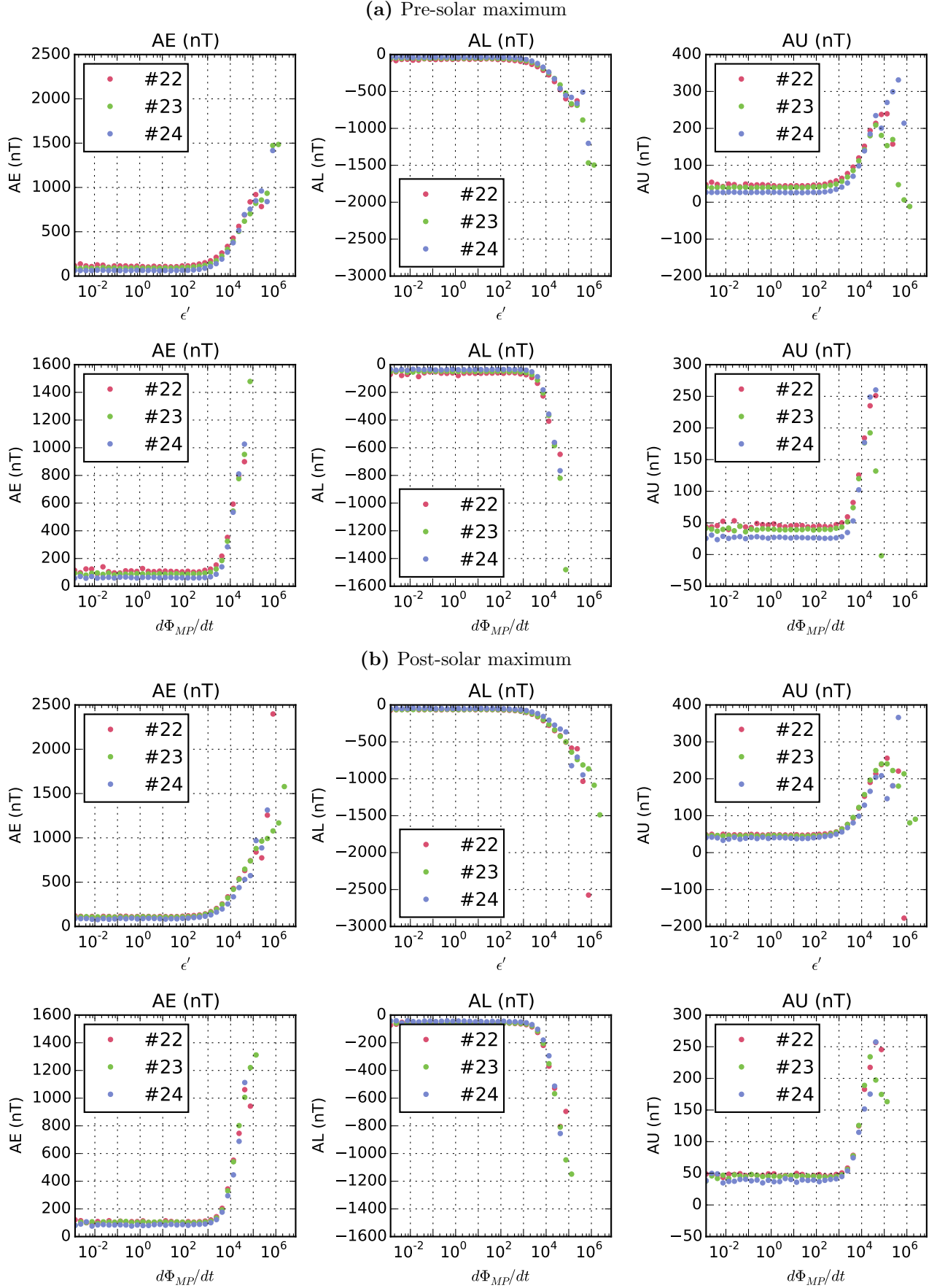


**Figure 6:** Cycle #24 divided by the average of #22 and #23 (method B).

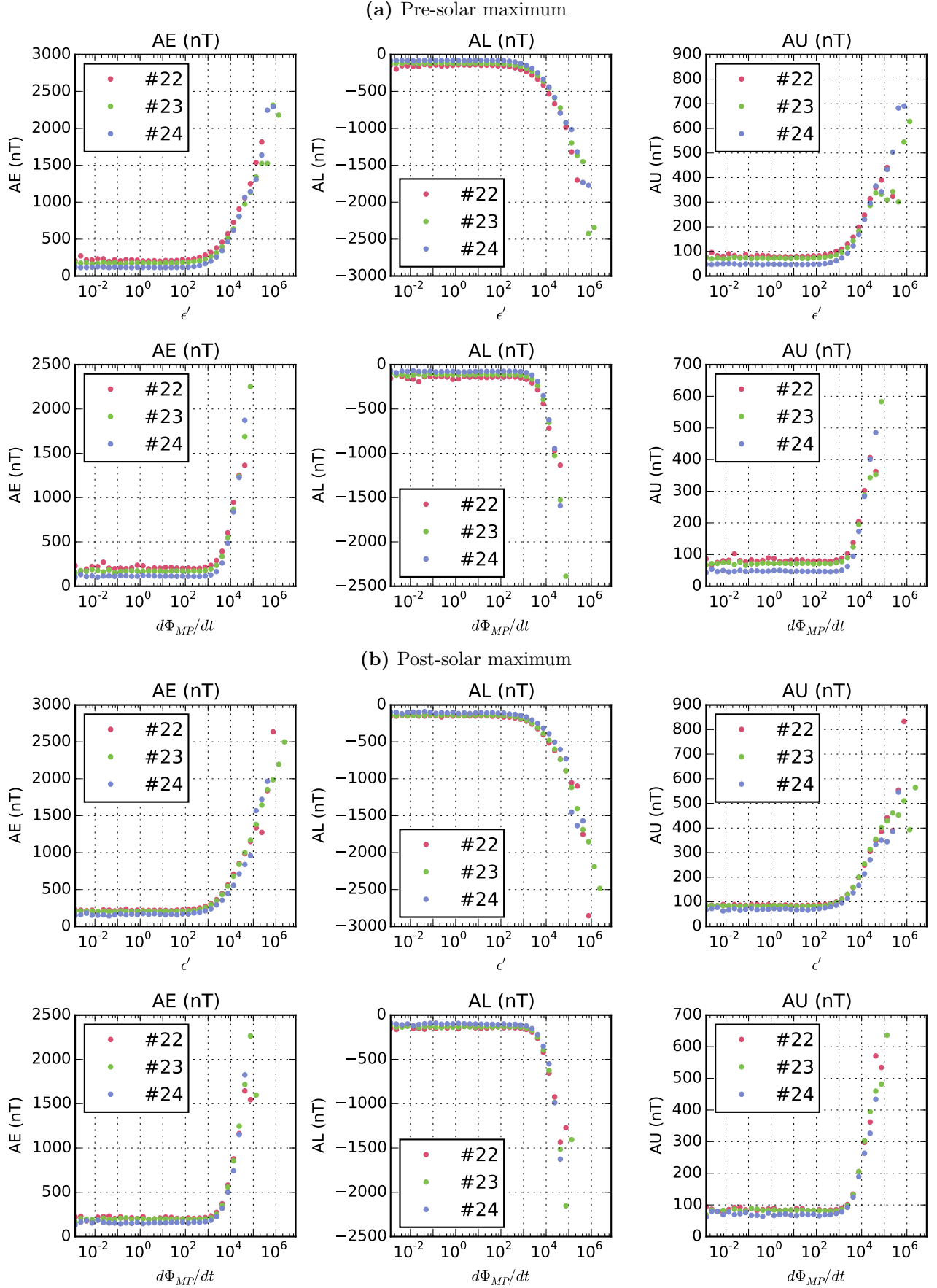
### 4.3 AE, AL and AU during pre- and post-solar maximum

Next, we further subdivide the dataset between inclining and declining phases of the solar cycle, i.e., before and after each solar maximum. Both methods A (Figure 7) and B (Figure 8) are again examined.

The results are quite reliable for relatively low solar wind inputs ( $\epsilon' < 10^4$  [ $\times 0.01$  W/km<sup>2</sup>]) with many data points, but scattering becomes overwhelming or high wind inputs ( $\epsilon' < 10^5$  [ $\times 0.01$  W/km<sup>2</sup>]) as shown in Figure 9.

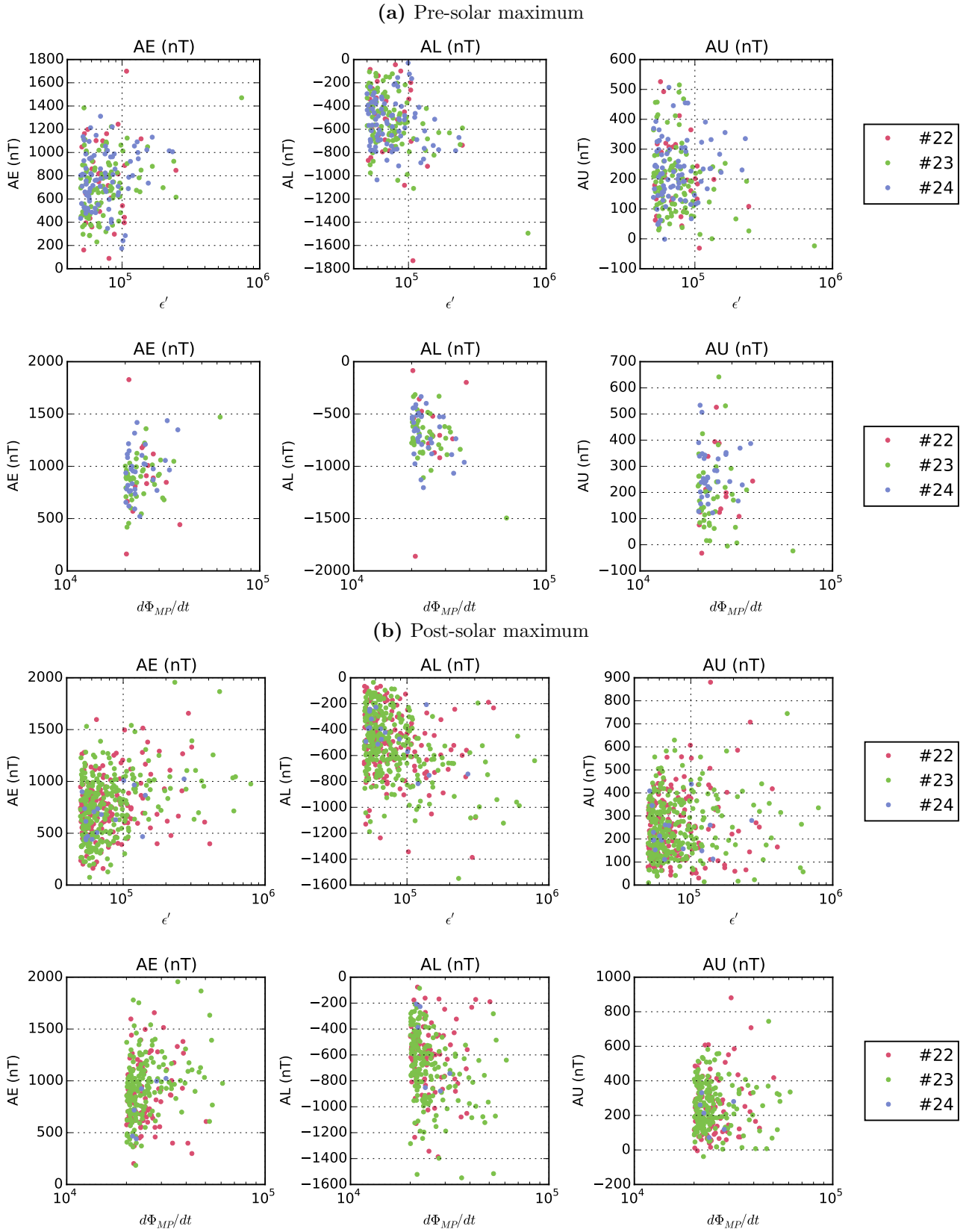


**Figure 7:** The same Figure 2 but further subdivided into (a) inclining phase and (b) declining phase for each solar cycle. The 5-minute delay is applied (method A).



**Figure 8:** The same Figure 5 but further subdivided into (a) inclining phase and (b) declining phase for each solar cycle. Maximum 5-minute values of AE, AL, and AU within 1-hour is used (method B).



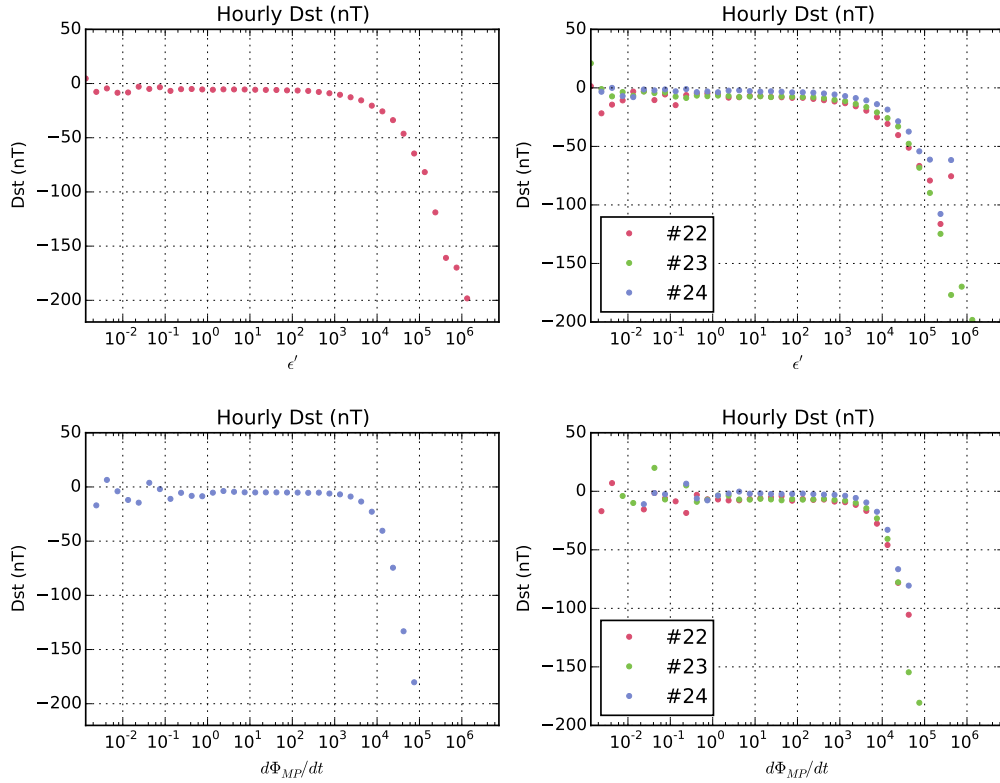


**Figure 9:** Scatter plot of AE, AL, AU against two different coupling functions limiting data to pre or post maximum.

#### 4.4 Dst-index

The Dst-index is only available in hourly resolution, but it is correlated with the coupling function parameters with a time delay of 5 minutes as shown in Figure 10.

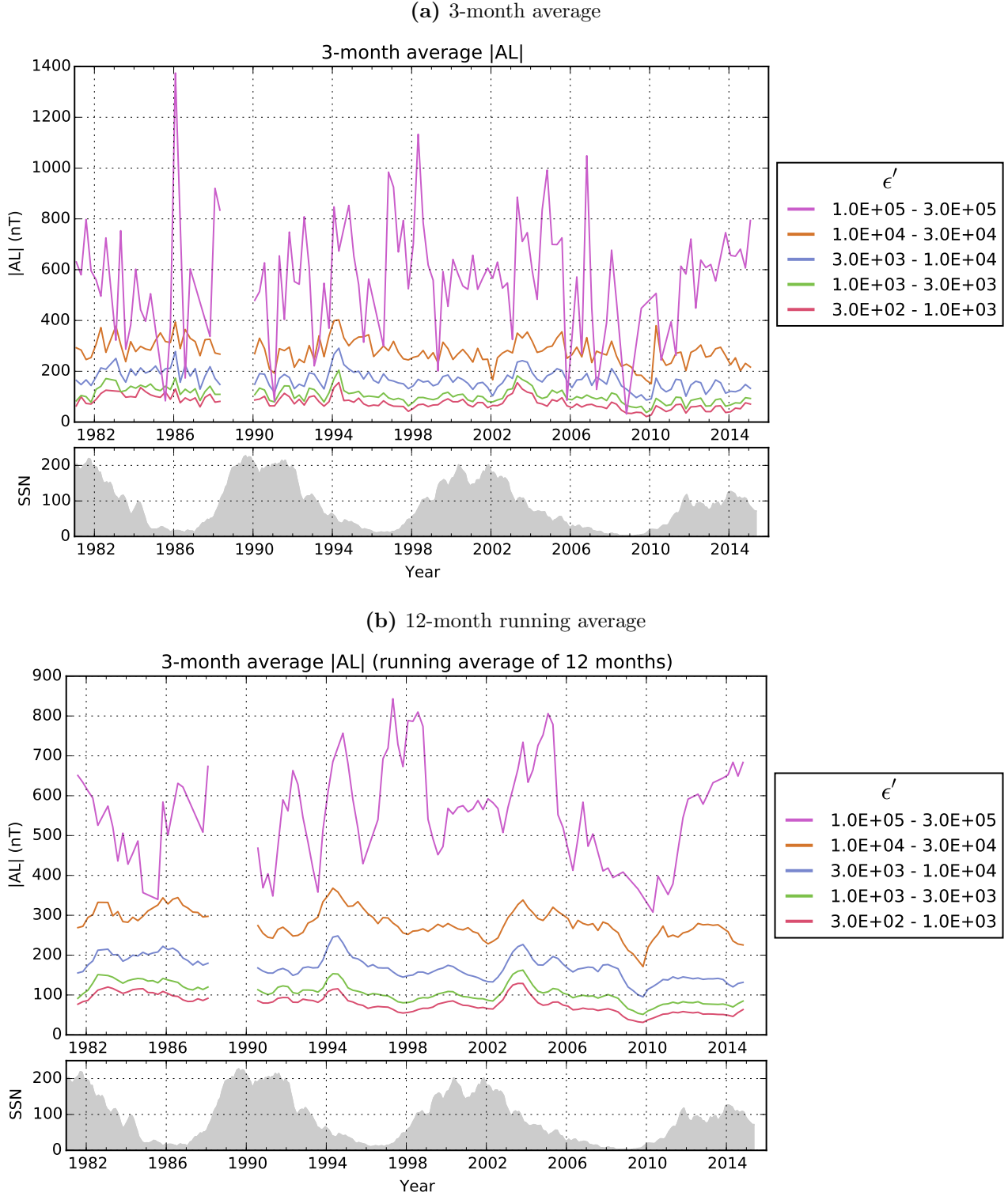
The cycle-to-cycle difference repeats what was obtained from 1-hour resolution Dst values ([Yamauchi, 2015]) repeats the that the solar cycle #24, even for high solar wind input ( $\epsilon'$  and  $d\Phi_{MP}/dt$ ), has a lower coupling efficiency than cycle #23 and #22. However, it should be noted that there are relatively few high-input events during cycle #24.



**Figure 10:** Hourly Dst index correlated with hourly average of  $\epsilon'$  and  $d\Phi_{MP}/dt$  calculated using 5-min resolution OMNI data. (similar to Method A)

## 5 Coupling efficiency over long time (annual and decadal)

Figure 11a shows the 3-month average of the absolute AL index for fixed solar wind input ( $\epsilon'$ ). The seasonal variation is removed with a running average of 12 months, as shown in Figure 11b. During the declining phase of each solar cycle, there is a brief ( $\sim 1$  year) increase in coupling efficiency.



**Figure 11:** Variation of average  $|AL|$  for fixed ranges of  $\epsilon'$  (in  $[0.01 \text{ W/km}^2]$ )

## 6 High input and big event data sets

In the space weather, big events are mainly discussed. This section focuses on these big events with high values of coupling functions or geomagnetic indices.

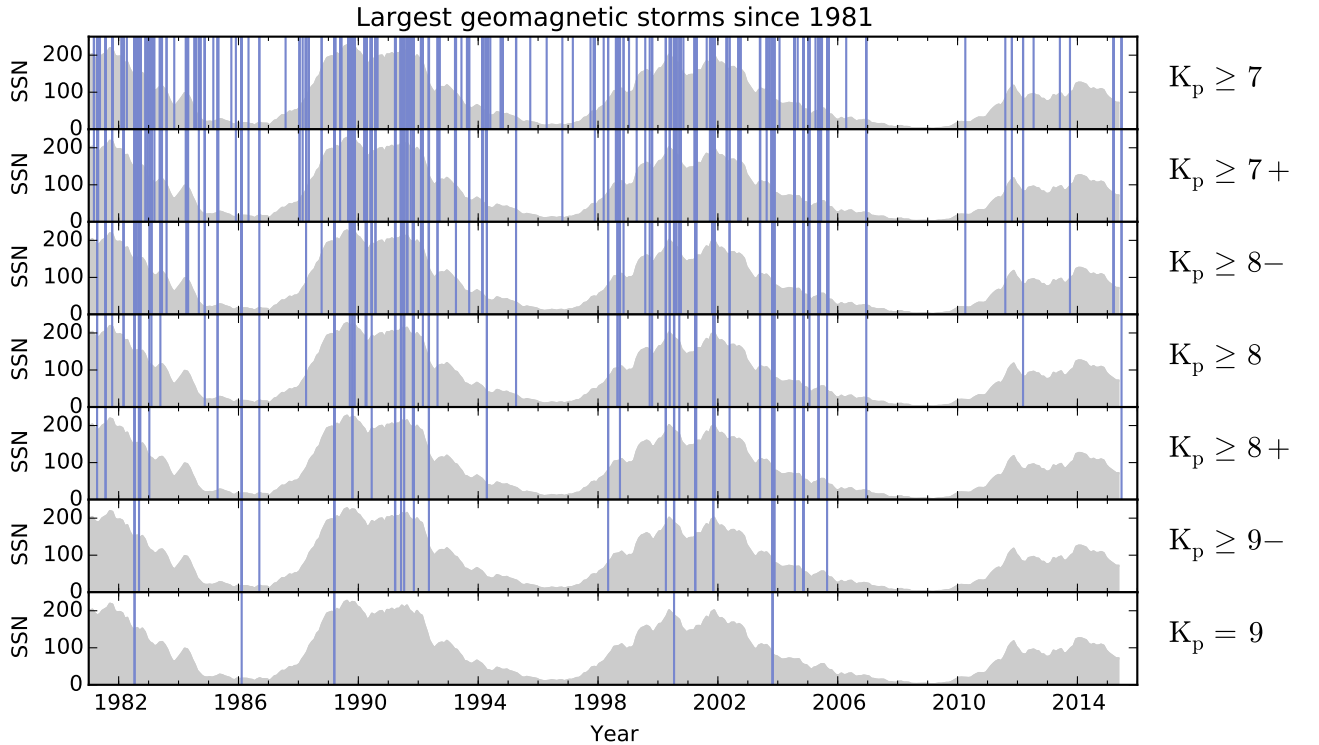
### 6.1 High Kp-index events

Table 3 shows the distribution of the Kp-index for the period between 1 January, 1981 and 15 July, 2015. Each 3-hour interval is counted. The current solar cycle #24 (starting 4 January, 2008) is extracted in the last row. In order to compare different solar cycles, it is important to consider the distribution of these events over time.

**Table 3:** K<sub>p</sub>-index distribution for 1981-2015.

K <sub>p</sub> -index =	6	6+	7-	7	7+	8-	8	8+	9-	9
<b>1981-2015</b>	502	315	248	160	131	100	52	45	33	9
<b>Cycle #24</b>	20	27	10	4	3	7	1	1	0	0

For the highest 5 levels of Kp (e.g.  $K_p \geq 8-$ ,  $8+$ ,  $9-$ ,  $9$ ), OMNI data sets are produced. The data sets contain the original OMNI data, but are filtered on the Kp condition. This means that these new datasets only contain the OMNI values during the high-Kp events. The data format of these datasets is the same as the original OMNI dataset, except for an added column indicating the Kp index. Figure 12 shows the cumulative distribution of high-Kp events.



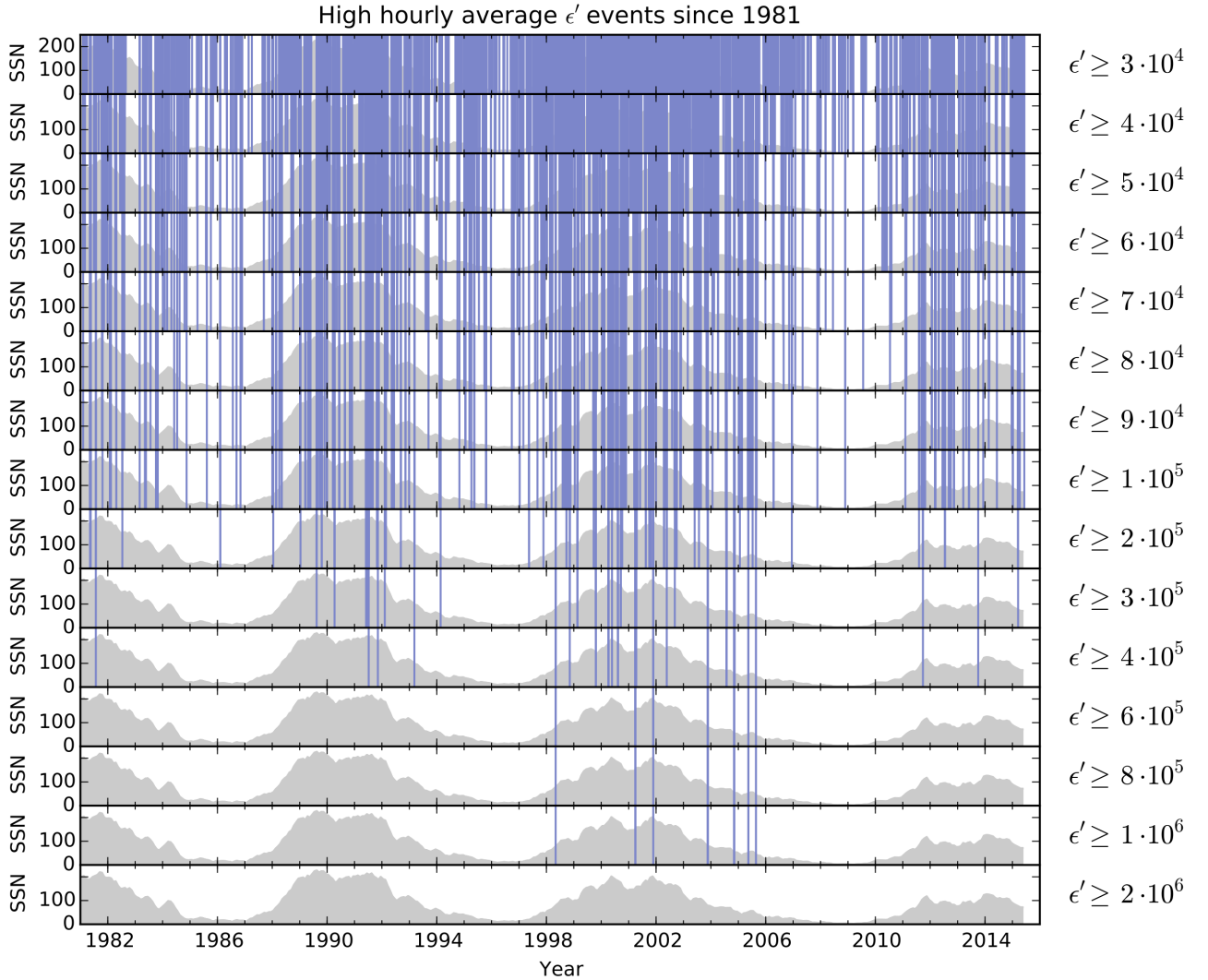
**Figure 12:** The occurrences of high Kp are plotted as vertical lines over the monthly sunspot number (running average of 5 months). Sunspot data is obtained from the World Data Center SILSO, Royal Observatory of Belgium, Brussels.

## 6.2 High $\epsilon'$ events

Table 4 shows the distribution of  $\epsilon'$  for the period between 1 January, 1981 and 15 July, 2015 (the  $\epsilon'$  counterpart of Table 3). Each 3-hour interval is counted. Figure 13 shows the events where the hourly average of  $\epsilon'$  is greater or equal than the  $\epsilon'$  threshold indicated (on the right).

**Table 4:**  $\epsilon'$  cumulative distribution for 1981-2015 and cycle #24. Each hourly interval is counted.

$\epsilon' \geq$	$3 \times 10^4$	$4 \times 10^4$	$5 \times 10^4$	$6 \times 10^4$	$7 \times 10^4$	$8 \times 10^4$	$9 \times 10^4$	$1 \times 10^5$	$2 \times 10^5$
<b>1981-2015</b>	4921	3118	2191	1643	1303	1041	859	714	202
<b>Cycle #24</b>	810	469	306	220	174	142	112	87	12
$\epsilon' \geq$	$3 \times 10^5$	$4 \times 10^5$	$6 \times 10^5$	$8 \times 10^5$	$1 \times 10^6$	$2 \times 10^6$			
<b>1981-2015</b>	97	61	26	21	18	0			
<b>Cycle #24</b>	3	2	0	0	0	0			



**Figure 13:** The same format as Figure 12 but for high  $\epsilon'$  events.

### 6.3 AE, AL and AU during high $\epsilon'$ and $d\Phi_{MP}/dt$ events

Figure 9 shows scatter plots of the coupling functions ( $\epsilon'$  and  $d\Phi_{MP}/dt$ ) versus AE index (AE, AL, and AU) for high input only. In such cases, the data is sparse and therefore averaging is less meaningful than for low input. Therefore, individual case studies are recommended.

## 7 Conclusion

This report features an extensive correlation analysis between the different coupling functions ( $\epsilon' = (4\pi/\mu_0)vB_T^2 \sin^4(\theta_c/2)$  and  $d\Phi_{MP}/dt = v^{4/3}B_T^{2/3} \sin^{8/3}(\theta_c/2)$ ) obtained from 5-minute resolution solar wind OMNI dataset and the geomagnetic activities (mainly AE, AL, and AU) to give hint of the solar cycle to cycle difference in these correlations (or Sun-Earth coupling efficiency). The obtained plots supported previous results using 1-hour resolution data [Yamauchi , 2015]. Furthermore, the Sun-Earth coupling efficiency shows a one-year peak during the declining phase of each solar cycle.

### Acknowledgements

Dst, Kp, AL, AU, AE and sunspot numbers (RI) are official IAGA- and IAA-endorsed indices that are provided by the World Data Center for Geomagnetism, Kyoto University, Japan (Dst and AL, AU, AE), GFZ, Adolf-Schmidt-Observatory Niemegek, Germany (Kp), and the Royal Observatory of Belgium, Brussels (RI). High resolution OMNI data, including the previously mentioned indices, are obtained through NASA/GSFC's Space Physics Data Facility's ftp service.

I'd like to thank Dr. Masatoshi Yamauchi for providing the opportunity to work for the Swedish Institute of Space Physics, and guiding me during this internship.

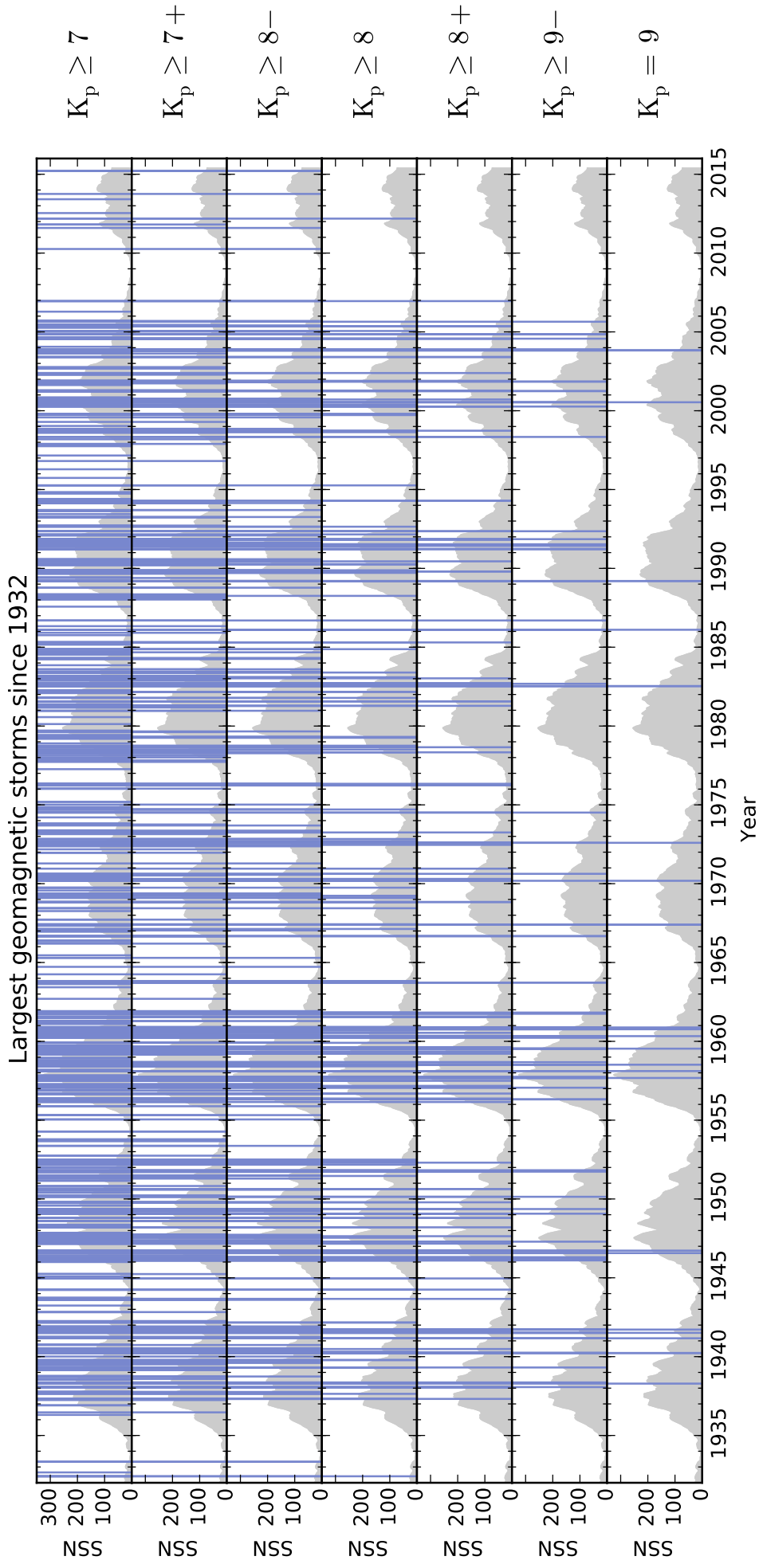
## 8 References

- Akasofu S.-I. (1981) Energy coupling between the solar wind and the magnetosphere. **Space Sci. Rev.**, **28**(2):121–190. doi:10.1007/BF00218810
- Perreault W.K., Akasofu S.-I. (1978) A study of geomagnetic storms. **Geophys. J. Int.**, **54**(3):547–573. doi:10.1111/j.1365-246X.1978.tb05494.x
- Newell P.T., Sotirelis T., Liou K., Meng C-I., Rich F.J. (2007) A nearly universal solar wind-magnetosphere coupling function inferred from 10 magnetospheric state variables. **J. Geophys. Res.**, **112**:A01206. doi:10.1029/2006JA012015
- Yamauchi M. (2015): Decreased Sun-Earth energy coupling efficiency starting from 2006, **Earth Planets Space**, **67**:44. doi:10.1186/s40623-015-0211-5

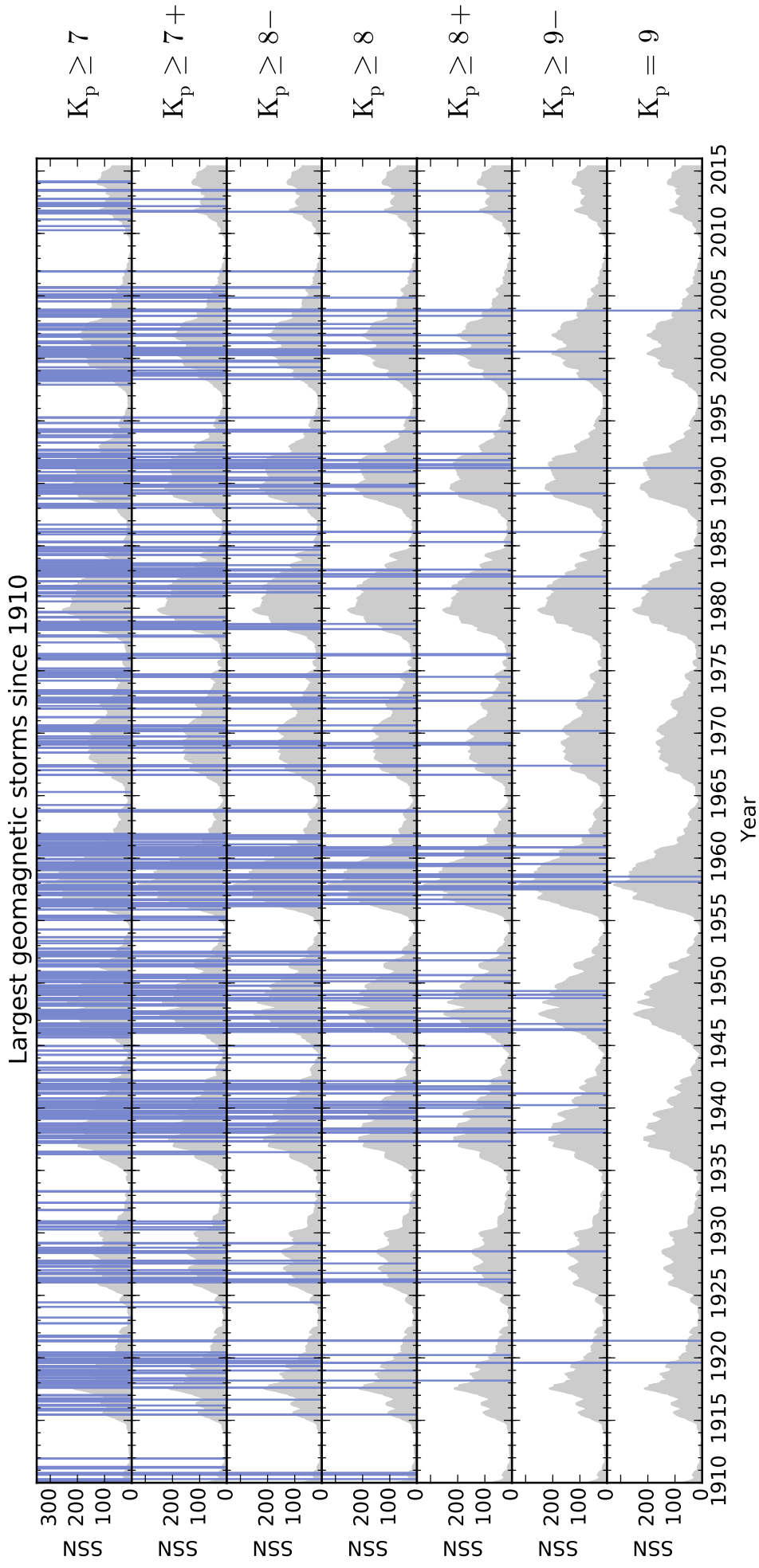


## **A Largest geomagnetic storms since 1932 and 1910**

The next two landscape-oriented pages will show the largest geomagnetic storms on a longer time scale.



**Figure 14:** Largest geomagnetic storms since 1932 as marked by the  $K_p$ -index.



**Figure 15:** Largest geomagnetic storms since 1910 as marked by the  $K_p$ -equivalent index.

## **B Python code for key figures**

### **B.1 Ratio of correlation (cf. Figure 3)**

figure3.py (=Appendix-B1)

### **B.2 Scatter plot (cf. Figure 9)**

figure4.py (=Appendix-B2)

### **B.3 Time series (cf. Figure 11a)**

figure2.py (=Appendix-B3)

### **B.4 Big events (cf. Figure 13)**

figure5.py (=Appendix-B4)

```

1 #-----1-----2-----3-----4-----5-----6-----7
2 # Appendix-B1
3 # epsilon vs AE (ratio of solar cycle 24 to the previous cycles)
4 # = sunspot-kp/angeo/figure3.py (v2015)
5 #-----1-----2-----3-----4-----5-----6-----7
6 import pandas as pd
7 import numpy as np
8 colspecs = [(0, 4), # Year 1995 ... 2006
9             (5, 8), # Day 1 ... 365 or 366
10            (9, 11), # Hour 0 ... 23
11            (12, 14), # Minute 0 ... 59 at start of average
12            (92, 99), # BY, nT (GSM)
13            (100, 107), # Bz, nT (GSM)
14            (124, 131), # Flow speed km/s
15            (246, 251), # AE-index nT
16            (252, 257), # AL-index nT
17            (258, 263) # AU-index NT
18            ]
19 names = ['Year', 'Day', 'Hour', 'Minute', 'By', 'Bz', 'v', 'AE', 'AL', 'AU']
20 frame = pd.read_fwf('spdf.gsfc.nasa.gov/pub/data/omni/high_res_omni_5min.asc', colspecs=colspecs, names=names)
21
22 frame['Bz'].replace(to_replace=9999.99, value=np.nan, inplace=True)
23 frame['By'].replace(to_replace=9999.99, value=np.nan, inplace=True)
24 frame['v'].replace(to_replace=9999.9, value=np.nan, inplace=True)
25 frame['AE'].replace(to_replace=99999, value=np.nan, inplace=True)
26 frame['AL'].replace(to_replace=99999, value=np.nan, inplace=True)
27 frame['AU'].replace(to_replace=99999, value=np.nan, inplace=True)
28
29 frame['Datetime'] = frame['Year'] * 10000000 + frame['Day'] * 10000 + frame['Hour'] * 100 + frame['Minute']
30 frame['Datetime'] = pd.to_datetime(frame['Datetime'], format='%Y%H%M')
31 frame.set_index('Datetime', inplace=True)
32 #frame.head()
33
34 d = frame.reset_index()
35 for index in range(len(d)):
36     indices = d.loc[index:index+29][['AL', 'AE', 'AU']]
37     d.set_value(index, 'AL', indices['AL'].min())
38     d.set_value(index, 'AE', indices['AE'].max())
39     d.set_value(index, 'AU', indices['AU'].max())
40
41 d.to_csv('data/omni_5min_highest_hourly_AEALAU.csv')
42
43 # You can skip the slow part above and load the cache/CSV file if available
44 import pandas as pd
45 frame = pd.read_csv('data/omni_5min_highest_hourly_AEALAU.csv')
46
47 import numpy as np
48 original_form = False
49
50 B_T = np.sqrt(frame['By'] ** 2 + frame['Bz'] ** 2)
51
52 if original_form:
53     theta = np.arctan2(frame['By'], frame['Bz'])
54     frame['akasofu'] = frame['v'] * B_T ** 2 * np.sin(theta/2) ** 4

```

```

55 frame['newell'] = frame['v'] ** (4/3) * B_T ** (2/3) * np.sin(theta/2) ** (8/3)
56 else:
57     frame['akasofu'] = frame['v'] * ((B_T - frame['Bz']) ** 2) / 4
58     frame['newell'] = ( frame['v'] ** 2 * ((B_T - frame['Bz']) ** 2 / (4*B_T)) ) ** (2/3)
59
60 frame['Datetime'] = pd.to_datetime(frame['Datetime'])
61 frame = frame.set_index('Datetime')
62
63 import numpy as np
64 import pandas as pd
65 %matplotlib inline
66
67 import matplotlib.pyplot as plt
68
69 n_bands = 40
70 b = np.logspace(-3, 7, n_bands*2+1)
71
72 d = frame[['akasofu', 'newell', 'AE', 'AL', 'AU']]
73
74 def logarithmic_bands(data, coupling_function):
75     bands = []
76     for i in range(0, n_bands*2, 2):
77         band_left = b[i]
78         band_center = b[i+1]
79         band_right = b[i+2]
80         condition1 = data[coupling_function] >= band_left
81         condition2 = data[coupling_function] < band_right
82         conditions = condition1 & condition2
83         AE_mean = data['AE'][conditions].mean()
84         AL_mean = data['AL'][conditions].mean()
85         AU_mean = data['AU'][conditions].mean()
86         bands.append([band_center, AE_mean, AL_mean, AU_mean])
87     return bands
88
89 akasofu_bands = logarithmic_bands(d, 'akasofu')
90 newell_bands = logarithmic_bands(d, 'newell')
91
92 # Plotting
93 # Paper plot
94 bands_frame = {}
95 for cycle_number in ['22', '23', '24']:
96     # Before maximum
97     condition1 = d.index > cycle[cycle_number][0]
98     condition2 = d.index < cycle_maxima[cycle_number]
99
100     cycle_frame = d[condition1 & condition2]
101     akasofu_bands = logarithmic_bands(cycle_frame, 'akasofu')
102     bands_frame[cycle_number] = pd.DataFrame(akasofu_bands).set_index(0)
103     bands_frame[cycle_number].columns = ['AE', 'AL', 'AU']
104 #bands_22_23 = (bands_frame['22'] + bands_frame['23']) / 2
105
106
107 fig = plt.figure(figsize=(8.5, 5.8))
108 ax1 = fig.add_subplot(231)

```

```

109 ax2 = fig.add_subplot(232)
110 ax3 = fig.add_subplot(233)
111 ax4 = fig.add_subplot(234)
112 ax5 = fig.add_subplot(235)
113 ax6 = fig.add_subplot(236)
114
115 bands_fraction = bands_frame['24'] / bands_frame['22']
116 bands_fraction.plot(style='.', y='AE', grid=True, ax=ax1, logx=True, color=colors[0], legend=None)
117 bands_fraction.plot(style='.', y='AL', grid=True, ax=ax2, logx=True, color=colors[0], legend=None)
118 bands_fraction.plot(style='.', y='AU', grid=True, ax=ax3, logx=True, color=colors[0], legend=None)
119
120 bands_fraction = bands_frame['23'] / bands_frame['22']
121 bands_fraction.plot(style='.', y='AE', grid=True, ax=ax1, logx=True, color=colors[2], legend=None)
122 bands_fraction.plot(style='.', y='AL', grid=True, ax=ax2, logx=True, color=colors[2], legend=None)
123 bands_fraction.plot(style='.', y='AU', grid=True, ax=ax3, logx=True, color=colors[2], legend=None)
124
125 bands_frame = {}
126 for cycle_number in ['22', '23', '24']:
127     # After maximum
128     condition1 = d.index > cycle_maxima[cycle_number]
129     condition2 = d.index < cycle[cycle_number][1]
130
131     cycle_frame = d[condition1 & condition2]
132     akasofu_bands = logarithmic_bands(cycle_frame, 'akasofu')
133     bands_frame[cycle_number] = pd.DataFrame(akasofu_bands).set_index(0)
134     bands_frame[cycle_number].columns = ['AE', 'AL', 'AU']
135
136
137 bands_fraction = bands_frame['24'] / bands_frame['22']
138 bands_fraction.plot(style='.', y='AE', grid=True, ax=ax4, logx=True, color=colors[0], legend=None)
139 bands_fraction.plot(style='.', y='AL', grid=True, ax=ax5, logx=True, color=colors[0], legend=None)
140 bands_fraction.plot(style='.', y='AU', grid=True, ax=ax6, logx=True, color=colors[0], legend=None)
141
142 bands_fraction = bands_frame['23'] / bands_frame['22']
143 bands_fraction.plot(style='.', y='AE', grid=True, ax=ax4, logx=True, color=colors[2], legend=None)
144 bands_fraction.plot(style='.', y='AL', grid=True, ax=ax5, logx=True, color=colors[2], legend=None)
145 bands_fraction.plot(style='.', y='AU', grid=True, ax=ax6, logx=True, color=colors[2], legend=None)
146
147
148 for ax in [ax1, ax2, ax3, ax4, ax5, ax6]:
149     ax.set_xlabel('\$epsilon\$.')
150     ax.set_ylim([.4, 1.6])
151     ax.axhspan(1.0, 1.6, color=colors[0], alpha=0.2)
152
153 for ax in [ax1, ax4]:
154     ax.set_title('AE (nT)')
155 for ax in [ax2, ax5]:
156     ax.set_title('AL (nT)')
157 for ax in [ax3, ax6]:
158     ax.set_title('AU (nT)')
159 for ax in [ax1, ax2, ax3, ax4, ax5, ax6]:
160     ax.set_ylabel('Ratio')
161
162 for ax in [ax1, ax2, ax3, ax4, ax5, ax6]:

```

```

163 lines, labels = ax.get_legend_handles_labels()
164 ax.legend(lines[:-1], ['#24', '#23'][:-1], loc='upper left', numpoints=1)
165
166 plt.tight_layout()
167
168 for ax in [ax1,ax2,ax3,ax4,ax5,ax6]:
169     xticks = ax.xaxis.get_major_ticks()
170     for i in range(1, len(xticks), 2):
171         xticks[i].label1.set_visible(False)
172     #ax.set_xlim([10**1, 10**6])
173
174 fig.subplots_adjust(hspace=0.4)
175
176 fig.savefig('Pre_Post_Maximum_OMNI_akasofu_newell_AE_AL_AU_cycle_paper.eps')
177 #-----1-----2-----3-----4-----5-----6-----7
178 # Appendix-B2
179 # scatter plot of epsilon vs AE (only for large epsilon)
180 # = sunspot-kp/angeo/figure4.py (v2015)
181 #-----1-----2-----3-----4-----5-----6-----7
182 import pandas as pd
183 method = 'hourly' # 'hourly'
184
185 # This CSV file has already been produced in figure3.py
186 frame = pd.read_csv('data/omni_5min_highest_'+method+'_AEALAU.csv')
187
188
189 # Calculate akasofu
190 import numpy as np
191 original_form = False
192
193 B_T = np.sqrt(frame['By'] ** 2 + frame['Bz'] ** 2)
194
195 if original_form:
196     theta = np.arctan2(frame['By'], frame['Bz'])
197     frame['akasofu'] = frame['v'] * B_T ** 2 * np.sin(theta/2) ** 4
198     frame['newell'] = frame['v'] ** (4/3) * B_T ** (2/3) * np.sin(theta/2) ** (8/3)
199 else:
200     frame['akasofu'] = frame['v'] * ((B_T - frame['Bz']) ** 2) / 4
201     frame['newell'] = ( frame['v'] ** 2 * ((B_T - frame['Bz']) ** 2 / (4*B_T)) ) ** (2/3)
202
203 frame['akasofu'] = frame['akasofu'] / 100 # to W/km^2
204
205
206 # Convert datetime index to datetime objects
207 frame['Datetime'] = pd.to_datetime(frame['Datetime'])
208 frame = frame.set_index('Datetime')
209
210
211 # Define logarithmic bands function
212 import numpy as np
213 import pandas as pd
214 %matplotlib inline
215
216 import matplotlib.pyplot as plt

```



```

217 n_bands = 40
218 b = np.logspace(-3-2, 7-2, n_bands*2+1)
219
220 d = frame[['akasoфу', 'newell', 'AE', 'AL', 'AU']]
221
222 def logarithmic_bands(data, coupling_function):
223     bands = []
224     for i in range(0, n_bands*2, 2):
225         band_left = b[i]
226         band_center = b[i+1]
227         band_right = b[i+2]
228         condition1 = data[coupling_function] >= band_left
229         condition2 = data[coupling_function] < band_right
230         conditions = condition1 & condition2
231         AE_mean = data['AE'][conditions].mean()
232         AL_mean = data['AL'][conditions].mean()
233         AU_mean = data['AU'][conditions].mean()
234         bands.append([band_center,
235                     AE_mean, AL_mean, AU_mean,
236                     np.count_nonzero(conditions)]#/np.count_nonzero(~np.isnan(data[coupling_function]))])
237     return bands
238
239
240
241 # Prepare dataframe (delay is optional and set to 0
242 # because we are using previously calculated peak hourly values directly)
243 delay = 0
244 akasoфу = frame['akasoфу'].values#[::-delay]
245 newell = frame['newell'].values#[::-delay]
246 AE = frame['AE'].values#[delay:]
247 AL = frame['AL'].values#[delay:]
248 AU = frame['AU'].values#[delay:]
249
250 d = pd.concat([pd.Series(frame.index.values), pd.Series(akasoфу), pd.Series(newell), pd.Series(AE), pd.Series(AL),
251                  pd.Series(AU)), axis=1]
252 d.columns = ['Datetime', 'akasoфу', 'newell', 'AE', 'AL', 'AU']
253
254
255 # Remove minute/second information (and average per hour)
256 d['Datetime'] = d['Datetime'].apply(lambda x: x.replace(minute=0, second=0))
257 d = d.groupby('Datetime').mean()
258
259
260 # Plot pre-maximum
261 %matplotlib inline
262 import matplotlib.pyplot as plt
263 from datetime import datetime, timedelta
264 colors = ["#D94E70",
265          "#7DC14B",
266          "#7887CE",
267          "#D4752C",
268          "#CA5DCA"]
269
270 fig = plt.figure(figsize=(8.5, 5.8*1.5))

```

```

271 ax1 = fig.add_subplot(231)
272 ax2 = fig.add_subplot(232)
273 ax3 = fig.add_subplot(233)
274 ax4 = fig.add_subplot(234)
275 ax5 = fig.add_subplot(235)
276 ax6 = fig.add_subplot(236)
277 axes = [ax1, ax2, ax3, ax4, ax5, ax6]
278
279
280 cycle = {}
281 cycle['21post'] = pd.to_datetime(['01-1800', '09-1986'], format='%m-%Y')
282 cycle['22pre'] = pd.to_datetime(['09-1986', '03-1991'], format='%m-%Y')
283 cycle['22post'] = pd.to_datetime(['03-1991', '08-1996'], format='%m-%Y')
284 cycle['23pre'] = pd.to_datetime(['08-1996', '12-2001'], format='%m-%Y')
285 cycle['23post'] = pd.to_datetime(['12-2001', '12-2008'], format='%m-%Y')
286 cycle['24pre'] = pd.to_datetime(['12-2008', '05-2014'], format='%m-%Y')
287 cycle['24post'] = pd.to_datetime(['05-2014', '01-2020'], format='%m-%Y')
288
289 cycle_colors = {}
290 cycle_colors['21'] = '#363636'
291 cycle_colors['22'] = colors[2]
292 cycle_colors['23'] = colors[1]
293 cycle_colors['24'] = colors[0]
294
295 for cycle_number in ['22', '23', '24']:
296     condition1 = d.index >= cycle[cycle_number+'pre'][0]
297     #condition1 = d.index > cycle_maxima[cycle_number]
298     #condition2 = d.index < cycle[cycle_number][1]
299     condition2 = d.index < cycle[cycle_number+'pre'][1]
300     cycle_frame = d[condition1 & condition2]
301     high_condition = cycle_frame['akasofu'] >= 5 * 10 ** 2
302     print('Cycle number', cycle_number, 'has', np.count_nonzero(high_condition), 'events')
303     #print(cycle_frame[high_condition])
304     last_dt = None
305     consecutive_events = []
306     for dt, row in cycle_frame[high_condition].iterrows():
307         if last_dt != (dt - timedelta(hours=1)):
308             # Add new group of consecutive values
309             consecutive_events.append([])
310
311         consecutive_events[len(consecutive_events)-1].append(row)
312         last_dt = dt
313     print('Consecutive events:', len(consecutive_events))
314
315     for i, index in enumerate(['AE', 'AL', 'AU']):
316         datapoints = []
317         for consecutive_event in consecutive_events:
318             ak = np.mean([row['akasofu'] for row in consecutive_event])
319             y = np.mean([row[index] for row in consecutive_event])
320             datapoints.append([ak, y])
321         events_scatter = pd.DataFrame(datapoints)
322         events_scatter.columns = ['akasofu', index+cycle_number]
323         #print(events_scatter.head())
324         events_scatter.plot(style='.', x='akasofu', y=index+cycle_number, grid=True, logx=True, color=cycle_colors[cycle_number])

```

```

325 high_condition = cycle_frame['newell'] >= 2 * 10 ** 4
326 print('cycle number', cycle_number, 'has', np.count_nonzero(high_condition), 'events')
327 #print(cycle_frame[high_condition])
328 last_dt = None
329 consecutive_events = []
330 for dt, row in cycle_frame[high_condition].iterrows():
331     if last_dt != (dt - timedelta(hours=1)):
332         # Add new group of consecutive values
333         consecutive_events.append([])
334
335 consecutive_events[len(consecutive_events)-1].append(row)
336 last_dt = dt
337 print('Consecutive events:', len(consecutive_events))
338
339 for i, index in enumerate(['AE', 'AL', 'AU']):
340     datapoints = []
341     for consecutive_event in consecutive_events:
342         ne = np.mean([row['newell'] for row in consecutive_event])
343         y = np.mean([row[index] for row in consecutive_event])
344         datapoints.append([ne, y])
345
346 events_scatter = pd.DataFrame(datapoints)
347 events_scatter.columns = ['newell', index+cycle_number]
348 #print(events_scatter.head())
349 events_scatter.plot(style='.', x='newell', y=index+cycle_number, grid=True, logx=True, color=cycle_colors[cycle_number])
350
351
352 for ax in [ax1, ax2, ax4, ax5]:
353     #lines, labels = ax.get_legend_handles_labels()
354     #ax.legend(lines, ['#22', '#23', '#24'], loc='best', numpoints=1)
355     ax.legend().remove()
356
357 legend = []
358 for ax in [ax3, ax6]:
359     lines, labels = ax.get_legend_handles_labels()
360     legend.append(ax.legend(lines, ['#22', '#23', '#24'], numpoints=1, loc='center left', bbox_to_anchor=(1.12, 0.5)))
361
362
363 for ax in [ax1, ax2, ax3]:
364     ax.set_xlabel('\$epsilon\`\\mathrm{[W/km^2]}')
365     for ax in [ax4, ax5, ax6]:
366         ax.set_xlabel('\$d\\Phi_{MP}/dt$')
367
368 for ax in [ax1, ax4]:
369     ax.set_ylabel('AE (nT)')
370     ax.set_title('AE (nT)')
371     for ax in [ax2, ax5]:
372         ax.set_ylabel('AL (nT)')
373         ax.set_title('AL (nT)')
374     for ax in [ax3, ax6]:
375         ax.set_ylabel('AU (nT)')
376         ax.set_title('AU (nT)')
377
378 plt.tight_layout()

```

```

379 for ax in [ax1,ax2,ax3,ax4,ax5,ax6]:
380     #xticks = ax.xaxis.get_major_ticks()
381     #for i in range(1, len(xticks), 2):
382         # xticks[i].label1.set_visible(False)
383     ax.set_xlim([4*10**2, 10**4])
384 for ax in [ax4,ax5,ax6]:
385     ax.set_xlim([1*10**4, 10**5])
386
387 fig.savefig('Pre-maximum-high-epsilon-scatter.eps', bbox_extra_artists=(legend[0], legend[1]), bbox_inches='tight')
388
389
390 # Plot post-maximum
391 %matplotlib inline
392 import matplotlib.pyplot as plt
393 from datetime import datetime, timedelta
394 colors = ["#D94E70",
395           "#7DC14B",
396           "#7887CE",
397           "#D4752C",
398           "#CA5DCA"]
399
400 fig = plt.figure(figsize=(8.5, 5.8*1.5))
401 ax1 = fig.add_subplot(231)
402 ax2 = fig.add_subplot(232)
403 ax3 = fig.add_subplot(233)
404 ax4 = fig.add_subplot(234)
405 ax5 = fig.add_subplot(235)
406 ax6 = fig.add_subplot(236)
407 axes = [ax1,ax2,ax3,ax4,ax5,ax6]
408
409
410 cycle = {}
411 cycle['21post'] = pd.to_datetime(['01-1800', '09-1986'], format='%m-%Y')
412 cycle['22pre'] = pd.to_datetime(['09-1986', '03-1991'], format='%m-%Y')
413 cycle['22post'] = pd.to_datetime(['03-1991', '08-1996'], format='%m-%Y')
414 cycle['23pre'] = pd.to_datetime(['08-1996', '12-2001'], format='%m-%Y')
415 cycle['23post'] = pd.to_datetime(['12-2001', '12-2008'], format='%m-%Y')
416 cycle['24pre'] = pd.to_datetime(['12-2008', '05-2014'], format='%m-%Y')
417 cycle['24post'] = pd.to_datetime(['05-2014', '01-2020'], format='%m-%Y')
418
419 cycle_colors = {}
420 cycle_colors['21'] = '#363636'
421 cycle_colors['22'] = colors[2]
422 cycle_colors['23'] = colors[1]
423 cycle_colors['24'] = colors[0]
424
425 for cycle_number in ['21', '22', '23', '24']:
426     condition1 = d.index >= cycle[cycle_number+'post'][0]
427     #condition1 = d.index > cycle_maxima[cycle_number]
428     #condition2 = d.index < cycle[cycle_number][1]
429     condition2 = d.index < cycle[cycle_number+'post'][1]
430     cycle_frame = d[condition1 & condition2]
431     high_condition = cycle_frame['akasofu'] >= 5 * 10 ** 2
432     print('Cycle number', cycle_number, 'has', np.count_nonzero(high_condition), 'events')

```

```

433 #print(cycle_frame[high_condition])
434 last_dt = None
435 consecutive_events = []
436 for dt, row in cycle_frame[high_condition].iterrows():
437     if last_dt != (dt - timedelta(hours=1)):
438         # Add new group of consecutive values
439         consecutive_events.append([])
440
441     consecutive_events[len(consecutive_events)-1].append(row)
442     last_dt = dt
443     print('Consecutive events:', len(consecutive_events))
444
445     for i, index in enumerate(['AE', 'AL', 'AU']):
446         datapoints = []
447         for consecutive_event in consecutive_events:
448             ak = np.mean([row['akasofu'] for row in consecutive_event])
449             y = np.mean([row[index] for row in consecutive_event])
450             datapoints.append([ak, y])
451         events_scatter = pd.DataFrame(datapoints)
452         #print(events_scatter.head())
453         events_scatter.plot(style='.', x='akasofu', y=index+cycle_number, grid=True, logx=True, color=cycle_colors[cycle_number])
454
455     high_condition = cycle_frame['newell'] >= 2 * 10 ** 4
456     print('Cycle number', cycle_number, 'has', np.count_nonzero(high_condition), 'events')
457     #print(cycle_frame[high_condition])
458     last_dt = None
459     consecutive_events = []
460     for dt, row in cycle_frame[high_condition].iterrows():
461         if last_dt != (dt - timedelta(hours=1)):
462             # Add new group of consecutive values
463             consecutive_events.append([])
464
465         consecutive_events[len(consecutive_events)-1].append(row)
466         last_dt = dt
467         print('Consecutive events:', len(consecutive_events))
468
469         for i, index in enumerate(['AE', 'AL', 'AU']):
470             datapoints = []
471             for consecutive_event in consecutive_events:
472                 ne = np.mean([row['newell'] for row in consecutive_event])
473                 y = np.mean([row[index] for row in consecutive_event])
474                 datapoints.append([ne, y])
475             events_scatter = pd.DataFrame(datapoints)
476             #print(events_scatter.head())
477             events_scatter.plot(style='.', x='newell', y=index+cycle_number, grid=True, ax=axes[i+3], logx=True, color=cycle_colors[cycle_number])
478
479
480
481
482 for ax in [ax1, ax2, ax4, ax5]:
483     #lines, labels = ax.get_legend_handles_labels()
484     #ax.legend(lines, ['#22', '#23', '#24'], loc='best', numpoints=1)
485     ax.legend().remove()
486

```

```

487 legend = []
488 for ax in [ax3, ax6]:
489     lines, labels = ax.get_legend_handles_labels()
490     legend.append(ax.legend(lines, ['#21', '#22', '#23', '#24'], numpoints=1, loc='center left', bbox_to_anchor=(1.12, 0.5)))
491
492
493 for ax in [ax1, ax2, ax3]:
494     ax.set_xlabel('\$epsilon\$\epsilon\backslash\mathrm{[W/km^2]}$')
495     for ax in [ax4, ax5, ax6]:
496         ax.set_xlabel('\$d\Phi_{MP}/dt$')
497
498 for ax in [ax1, ax4]:
499     ax.set_ylabel('AE (nT)')
500     ax.set_title('AE (nT)')
501 for ax in [ax2, ax5]:
502     ax.set_ylabel('AL (nT)')
503     ax.set_title('AL (nT)')
504 for ax in [ax3, ax6]:
505     ax.set_ylabel('AU (nT)')
506     ax.set_title('AU (nT)')
507
508 plt.tight_layout()
509 for ax in [ax1, ax2, ax3, ax4, ax5, ax6]:
510     #xticks = ax.xaxis.get_major_ticks()
511     #for i in range(1, len(xticks), 2):
512         # xticks[i].label1.set_visible(False)
513     ax.set_xlim([4*10**2, 10**4])
514 for ax in [ax4, ax5, ax6]:
515     ax.set_xlim([1*10**4, 10**5])
516
517 ax4.set_ylim([0, 4000])
518 plt.tight_layout()
519
520 fig.savefig('Large_Post-maximum-high-epsilon-scatter.eps', bbox_extra_artists=(legend[0], legend[1]), bbox_inches='tight')
521 #-----1-----2-----3-----4-----5-----6-----7
522 # Appendix-B3
523 # time series plot of AE for fixed epsilon
524 # = sunspot-kp/angeo/figure2.py (v2015)
525 #-----1-----2-----3-----4-----5-----6-----7
526 import pandas as pd
527 import numpy as np
528 colspecs = [(0, 4), # Year 1995 ... 2006
529             (5, 8), # Day 1 ... 365 or 366
530             (9, 11), # Hour 0 ... 23
531             (12, 14), # Minute 0 ... 59 at start of average
532             (92, 99), # BY, nT (GSM)
533             (100, 107), # Bz, nT (GSM)
534             (124, 131), # Flow speed km/s
535             (246, 251), # AE-index nT
536             (252, 257), # AL-index nT
537             (258, 263) # AU-index NT
538 ]
539 names = ['Year', 'Day', 'Hour', 'Minute', 'By', 'Bz', 'v', 'AE', 'AL', 'AU']
540 frame = pd.read_fwf('spdf.gsfc.nasa.gov/pub/data/omni/high_res_omni_5min.asc', colspecs=colspecs, names=names)

```

```

541 frame['Bz'].replace(to_replace=9999.99, value=np.nan, inplace=True)
542 frame['By'].replace(to_replace=9999.99, value=np.nan, inplace=True)
543 frame['v'].replace(to_replace=9999.9, value=np.nan, inplace=True)
544 frame['AE'].replace(to_replace=99999, value=np.nan, inplace=True)
545 frame['AL'].replace(to_replace=99999, value=np.nan, inplace=True)
546 frame['AU'].replace(to_replace=99999, value=np.nan, inplace=True)
547 frame['AU'].replace(to_replace=99999, value=np.nan, inplace=True)
548
549 frame['Datetime'] = frame['Year'] * 10000000 + frame['Day'] * 10000 + frame['Hour'] * 100 + frame['Minute']
550 frame['Datetime'] = pd.to_datetime(frame['Datetime'], format='%Y%H%M')
551 frame.set_index('Datetime', inplace=True)
552
553 # Calculate epsilon and newell using alternate form
554 original_form = False
555
556 B_T = np.sqrt(frame['By'] ** 2 + frame['Bz'] ** 2)
557
558 if original_form:
559     theta = np.arctan2(frame['By'], frame['Bz'])
560     frame['akasofu'] = frame['v'] * B_T ** 2 * np.sin(theta/2) ** 4
561     frame['newell'] = frame['v'] ** (4/3) * B_T ** (2/3) * np.sin(theta/2) ** (8/3)
562 else:
563     frame['akasofu'] = frame['v'] * ((B_T - frame['Bz']) ** 2) / 4
564     frame['newell'] = ( frame['v'] ** 2 * ((B_T - frame['Bz']) ** 2 / (4*B_T)) ) ** (2/3)
565
566 # frame[['akasofu', 'newell']].describe().applymap(lambda x: '{:.2G}'.format(x))
567 # akasofu          newell
568 # count 2.3E+06    2.3E+06
569 # mean  4.9E+03    3.9E+03
570 # std   2E+04      4.2E+03
571 # min   0          0
572 # 25%  1.7E+02    8.6E+02
573 # 50%  1.2E+03    2.7E+03
574 # 75%  4.3E+03    5.5E+03
575 # max   1.9E+06    1.1E+05
576
577 # Check data coverage date
578 # frame[['akasofu'][-np.isnan(frame['akasofu'])].tail(1)
579 # 2015-07-30 23:50:00 16890.425449
580
581 # Group |AL| per 3 months and use 5-min delay
582 delay = 1
583 akasofu = frame['akasofu'].values[:(-delay)]
584 newell = frame['newell'].values[:(-delay)]
585 AE = np.abs(frame['AE']).values[:(-delay)]
586 AL = np.abs(frame['AL']).values[:(-delay)]
587 AU = np.abs(frame['AU']).values[:(-delay)]
588
589 d = pd.concat([pd.Series(frame.index.values[:(-delay)],
590                        pd.Series(akasofu), pd.Series(newell),
591                        pd.Series(AE), pd.Series(AL), pd.Series(AU)], axis=1)
592
593 d.columns = ['Datetime', 'akasofu', 'newell', 'AE', 'AL', 'AU']
594 months = ['02', '05', '08', '11'] # Feb, May, Aug, Nov

```

```

595 d['Datetime group'] = d['Datetime'].apply(lambda x: str(x.year) + '-' + months[int(np.floor((x.month - 1) / 3))]
596 d.set_index('Datetime', inplace=True)
597
598 d['Datetime group'] = pd.to_datetime(d['Datetime group'], format='%Y-%m')
599
600 # Turn data into bands (note that units are not yet converted to W/km2)
601 bands = [[300, 1000],
602           [1000, 3000],
603           [3000, 10000],
604           [10000, 30000],
605           [100000, 300000]]
606 AL_bands = []
607 for band in bands:
608     condition1 = d['akasofu'] >= band[0]
609     condition2 = d['akasofu'] < band[1]
610     AL_bands.append(d[condition1 & condition2][['Datetime group', 'AL']])
611
612
613 # Plotting code (and add Sunspot number plot)
614 %matplotlib inline
615 import matplotlib.pyplot as plt
616 import matplotlib.dates as dates
617 import matplotlib.gridspec as gridspec
618
619 colors = ["#D94E70",
620           "#7DC14B",
621           "#7887CE",
622           "#D4752C",
623           "#CA5DCA"]
624
625
626 fig = plt.figure(figsize=(8, 6))
627 gs = gridspec.GridSpec(2, 1, height_ratios=[4,1])
628
629 ax1 = plt.subplot(gs[0])
630 ax2 = plt.subplot(gs[1], sharex=ax1)
631
632 for i, AL_band in enumerate(AL_bands):
633     result = AL_band.groupby('Datetime group').mean()
634     #result = pd.rolling_mean(result, 4, center=True)
635     result.plot(ax=ax1, color=colors[i], grid=True, rot=0)
636
637 ax1.set_ylabel('|AL| (nT)')
638
639 lines, labels = ax1.get_legend_handles_labels()
640 legend = ax1.legend(lines[1:-1],
641                    [str('{:1E}'.format(band[0]/100)) + '-' + str('{:1E}'.format(band[1]/100)) for band in bands[1:-1]],
642                    loc='center left', bbox_to_anchor=(1, 0.5), title='${\epsilon}\mathrm{[W/km^2]}$', fontsize=11)
643 plt.setp(legend.get_title(), fontsize=16)
644 for legobj in legend.legendHandles:
645     legobj.set_linewidth(2.0)
646
647 box = ax1.get_position()
648 ax1.set_position([box.x0, box.y0, box.width * 0.8, box.height])

```



```

649 ax1.set_title('3-month average |AL|')
650
651 from datetime import datetime, timedelta
652 def convert_partial_year(number):
653     year = int(number)
654     d = timedelta(days=(number - year)*365)
655     day_one = datetime(year,1,1)
656     date = d + day_one
657     return date
658
659 SN = pd.read_csv('SN_m_tot_V2.0.csv', sep=';', header=None)
660 SN['Datetime'] = SN[2].apply(convert_partial_year)
661
662 SN.set_index('Datetime', inplace=True)
663 SN[3] = pd.rolling_mean(SN[3], 5, center=True)
664 ax = SN.plot(ax=ax2, y=3, legend=None, color='0.8', zorder=1, rot=0, grid=True)
665 ax.fill_between(SN.index, 0, SN[3], color='0.8')
666 ax.set_xlim(pd.to_datetime([1981, 2016]), format='%Y')
667 ax.set_ylim([0, 250])
668
669 for ax in [ax1, ax2]:
670     ax.set_xlabel('Year')
671     ax.xaxis.set_minor_locator(dates.YearLocator())
672
673 for label in ax.get_xticklabels():
674     label.set_horizontalalignment('center')
675
676 ax1.set_xlabel('')
677 fig.subplots_adjust(hspace=0.115)
678 ax2.set_ylabel('SSN')
679 plt.yticks(np.arange(0, 250, 100))
680
681 fig.savefig('HRO-Time-Series-AL-paper.eps', bbox_extra_artists=(legend,), bbox_inches='tight')
682 #-----1-----2-----3-----4-----5-----6-----7
683 # Appendix-B4
684 # time series plot of Sun-Spot Number(RI) with large events marked
685 # = sunspot-kp/angeo/figure5.py (v2015)
686 #-----1-----2-----3-----4-----5-----6-----7
687 import pandas as pd
688 import numpy as np
689 colspecs = [(0, 4), # Year 1995 ... 2006
690             (5, 8), # Day 1 ... 365 or 366
691             (9, 11), # Hour 0 ... 23
692             (12, 14), # Minute 0 ... 59 at start of average
693             (60, 67), # Field magnitude average, nT
694             (76, 83), # BY, nT (GSE)
695             (84, 91), # Bz, nT (GSE)
696             (124, 131), # Flow speed km/s
697             (246, 251), # AE-index nT
698             (252, 257), # AL-index nT
699             (258, 263) # AU-index NT
700 ]
701 names = ['Year', 'Day', 'Hour', 'Minute', 'B', 'By', 'Bz', 'v', 'AE', 'AL', 'AU']
702 frame = pd.read_fwf('spdf.gsfc.nasa.gov/pub/data/omni/high_res_omni/omni_5min.asc', colspecs=colspecs, names=names)

```

```

703 frame['Bz'].replace(to_replace=9999.99, value=np.nan, inplace=True)
704 frame['By'].replace(to_replace=9999.99, value=np.nan, inplace=True)
705 frame['B'].replace(to_replace=9999.99, value=np.nan, inplace=True)
706 frame['AE'].replace(to_replace=99999, value=np.nan, inplace=True)
707 frame['AL'].replace(to_replace=99999, value=np.nan, inplace=True)
708 frame['AU'].replace(to_replace=99999, value=np.nan, inplace=True)
709 frame['v'].replace(to_replace=99999.9, value=np.nan, inplace=True)
710
711 # Don't use minutes, we are interested in hourly averages only
712 frame['Datetime'] = frame['Year'] * 10000000 + frame['Day'] * 10000 + frame['Hour'] * 100
713
714 theta = np.arctan2(frame['By'], frame['Bz'])
715 B_T = np.sqrt(frame['By'] ** 2 + frame['Bz'] ** 2)
716 frame['akasofu'] = frame['v'] * B_T ** 2 * np.sin(theta/2) ** 4
717 frame['newell'] = frame['v'] ** (4/3) * B_T ** (2/3) * np.sin(theta/2) ** (8/3)
718
719 # Calculate hourly averages
720 frame = frame.groupby('Datetime').mean()
721
722 # Turn integer index of hourly averages into datetimes
723 frame = frame.reset_index()
724 frame['Datetime'] = pd.to_datetime(frame['Datetime'], format='%Y%H%M')
725 frame.set_index('Datetime', inplace=True)
726
727 print(frame['akasofu'].describe())
728 # count      211359.000000
729 # mean        1103.194195
730 # std         3069.888314
731 # min          0.000000
732 # 25%         110.016861
733 # 50%         403.112641
734 # 75%         1113.352047
735 # max         226225.913370
736 # Name: akasofu, dtype: float64
737
738 # Plotting logic:
739 #for i, lower_limit in enumerate([
740 #    1*10 ** 4,
741 #    2*10 ** 4,
742 #    3*10 ** 4,
743 #    4*10 ** 4,
744 #    5*10 ** 4,
745 #    6*10 ** 4,
746 #    7*10 ** 4,
747 #    8*10 ** 4,
748 #    9*10 ** 4,
749 #    1*10 ** 5,
750 #    2*10 ** 5,
751 #    3*10 ** 5]):
752 #    condition = frame['akasofu'] >= lower_limit
753 #    big_events = frame[condition].index

```





**Institutet för rymdfysik**

**Swedish Institute of Space Physics**

Swedish Institute of Space Physics  
Box 812, SE- 981 28 Kiruna, SWEDEN  
tel. +46-980-790 00, fax +46-980-790 50, e-post: [irf@irf.se](mailto:irf@irf.se)

**[www.irf.se](http://www.irf.se)**