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
Swedis 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

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 ϵ, ϵ', 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 ($\epsilon' < 10^2\,\mathrm{W/km^2}$), but not for active periods with $\epsilon' > 5 \times 10^2\,\mathrm{W/km^2}$ 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 ($\epsilon' > 5 \times 10^2\,\mathrm{W/km^2}$) 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 (ϵ' and dΦ_{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],

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

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

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

where: v = flow speed [km/s]
B_T = transverse component of IMF [nT] = $\sqrt{B_y^2 + B_z^2}$
θ_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

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

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

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

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

$$
\begin{aligned}
d\Phi_{MP}/dt &= \left(v^2B_T\left(\sin^2(\theta_c/2)\right)^2\right)^{\frac{2}{3}} \quad \text{substitute} \quad \left(\sin^2(\theta_c/2)\right)^2 = \frac{\left(1-\frac{B_z}{B_T}\right)^2}{4} \\
&= \left(v^2B_T\frac{\left(1-\frac{B_z}{B_T}\right)^2}{4}\right)^{\frac{2}{3}} \\
&= \left(v^2\frac{(B_T-B_Z)^2}{4B_T}\right)^{\frac{2}{3}}
\end{aligned}
$$

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 ϵ' and $d\Phi_{MP}/dt$ are also referred to as *high solar wind input.*

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

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 ($R_E = 6371\,\text{km}$) from the center of the Earth. The average flow speed in the HRO data set is $434\,\text{km}\,\text{s}^{-1}$. 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.

Year	I4	1995 ... 2006
Day	I4	1 ... 365 or 366
Hour	I3	0 ... 23
Minute	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	
By, nT (GSM)	F8.2	Determined from post-shift GSE components
Bz, nT (GSM)	F8.2	Determined from post-shift GSE components
RMS SD B scalar, nT	F8.2	
RMS SD field vector, nT	F8.2	
Flow speed, km/s	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	
AE-index, nT	I6	
AL-index, nT	I6	
AU-index, nT	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 (ϵ' 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 (ϵ' 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 W/km^2] in Figures 1 and 2. Figures 1 shows the correlation using entire dataset from 1981, whereas different solar cycles (#22, #23, and #24) are subdivided in Figure 2.

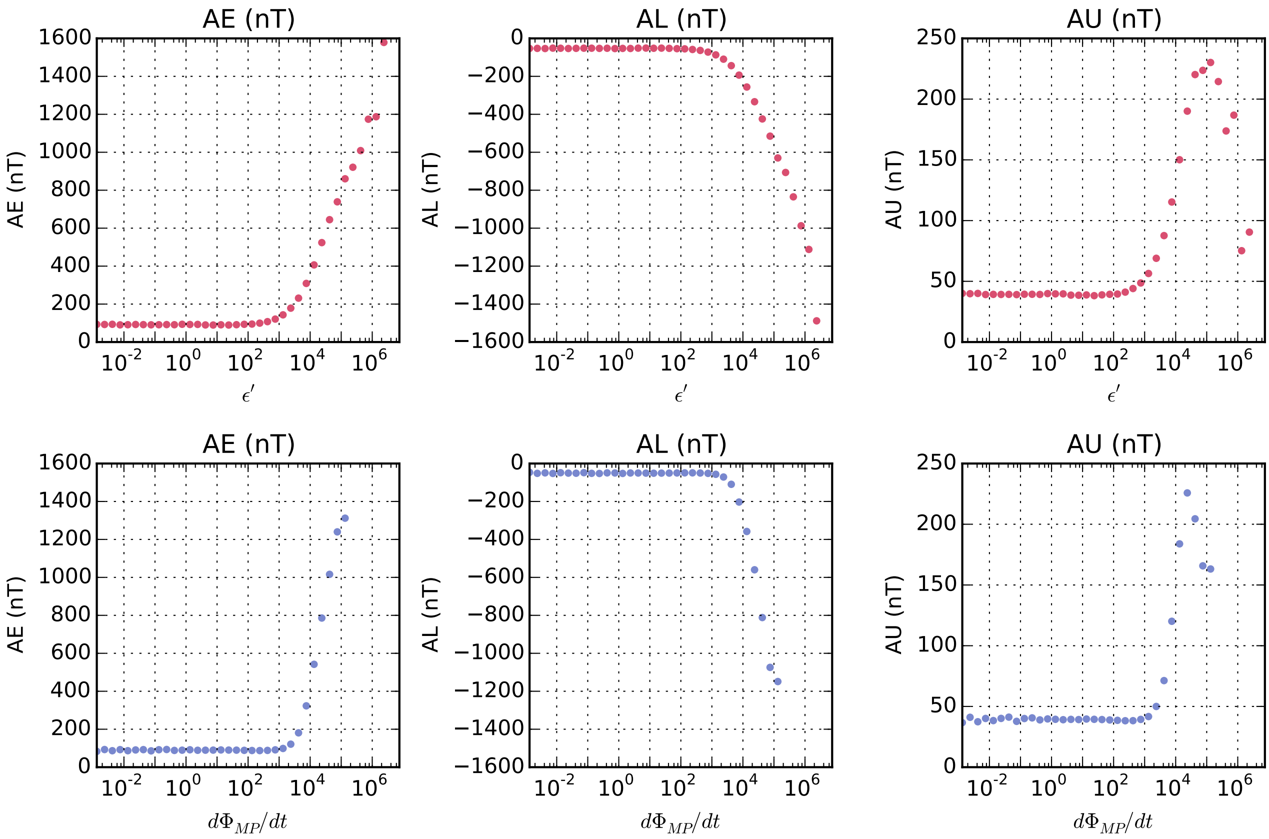


Figure 1: Correlations between the 5-minute resolution coupling functions and the 5-minute averaged geomagnetic AE indices (AE, AL, and AU) with 5-minute time lag (method A). The input values (x-axis) are divided into 40 evenly distributed bins.

Figure 2 indicates that the same low input ($\epsilon' < 10^4$ [$\times$ 0.01 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 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 W/km^2], and this will be studied in more detail in a later section.

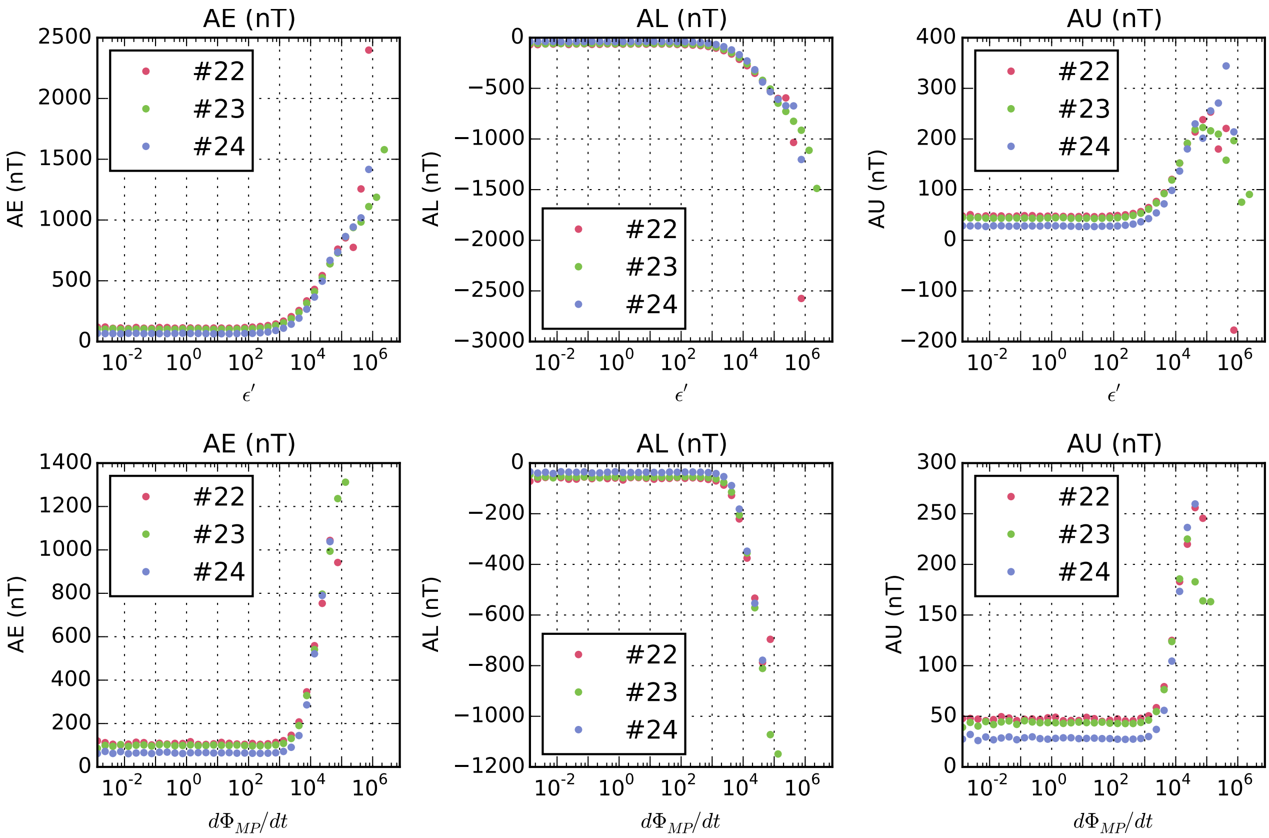


Figure 2: The same as Figure 1 but subdivided into different solar cycles.

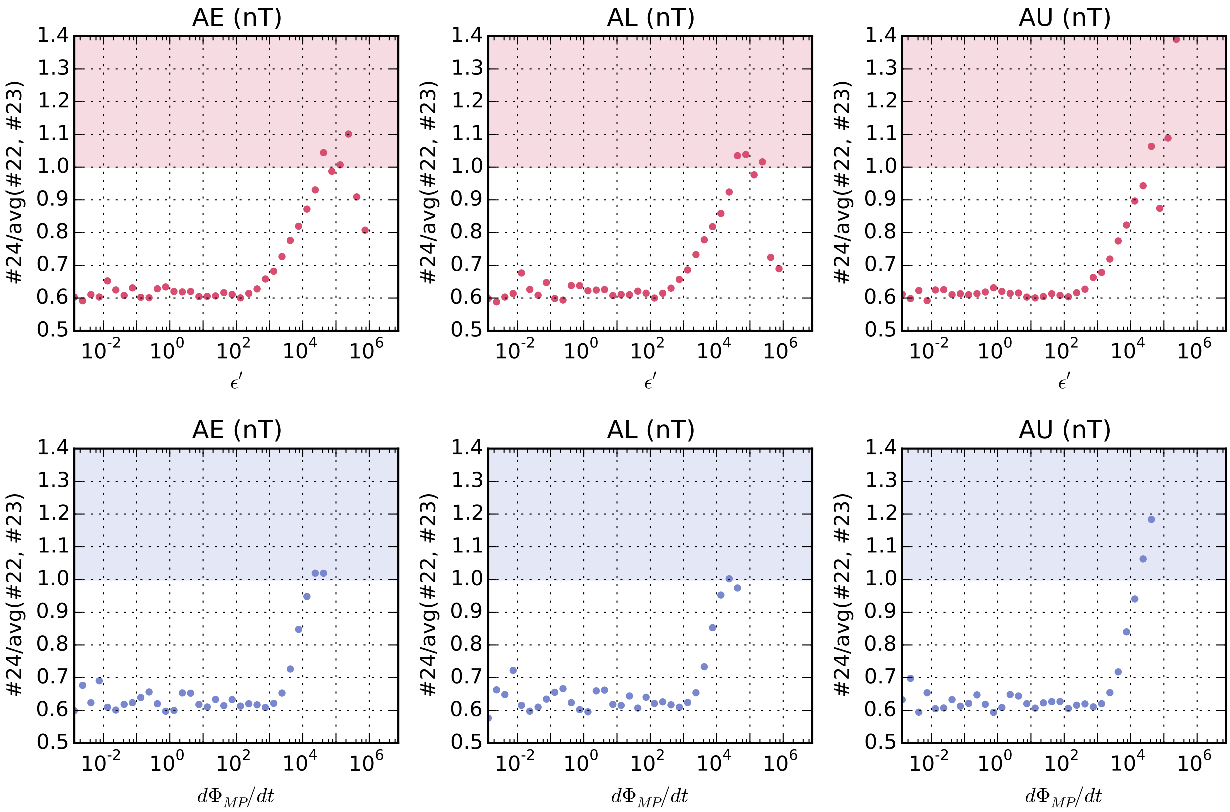


Figure 3: Cycle #24 divided by the average of #22 and #23 (method A).

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 aveaged AE, AL and AU during the following hour are correlated with ϵ' 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).

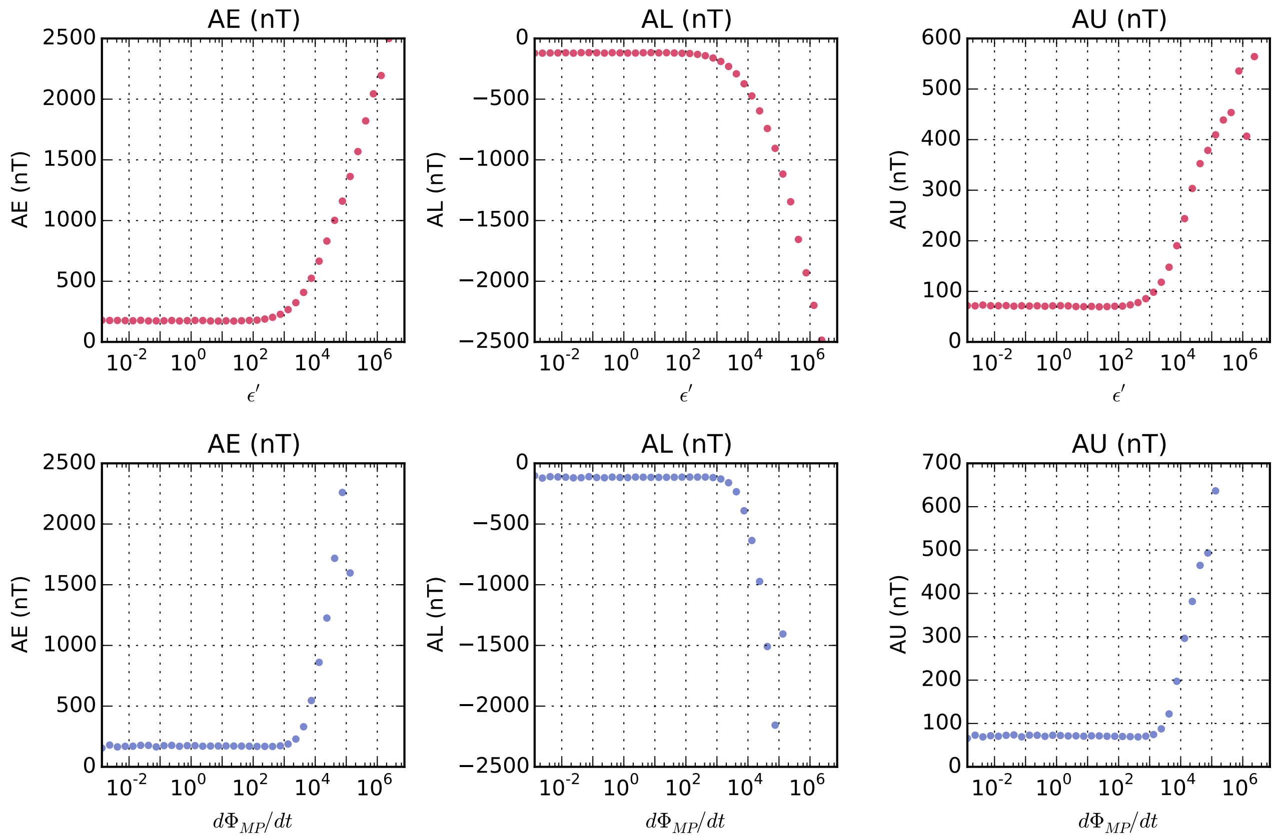


Figure 4: Correlations between the 5-minute resolution coupling functions and maximum 5-minute averaged values of geomagnetic AE indices (AE, AL, and AU) within following 1 hour (method B). The input values (x-axis) are divided into 40 evenly distributed bins. The entire 5-minute resolution OMNI data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

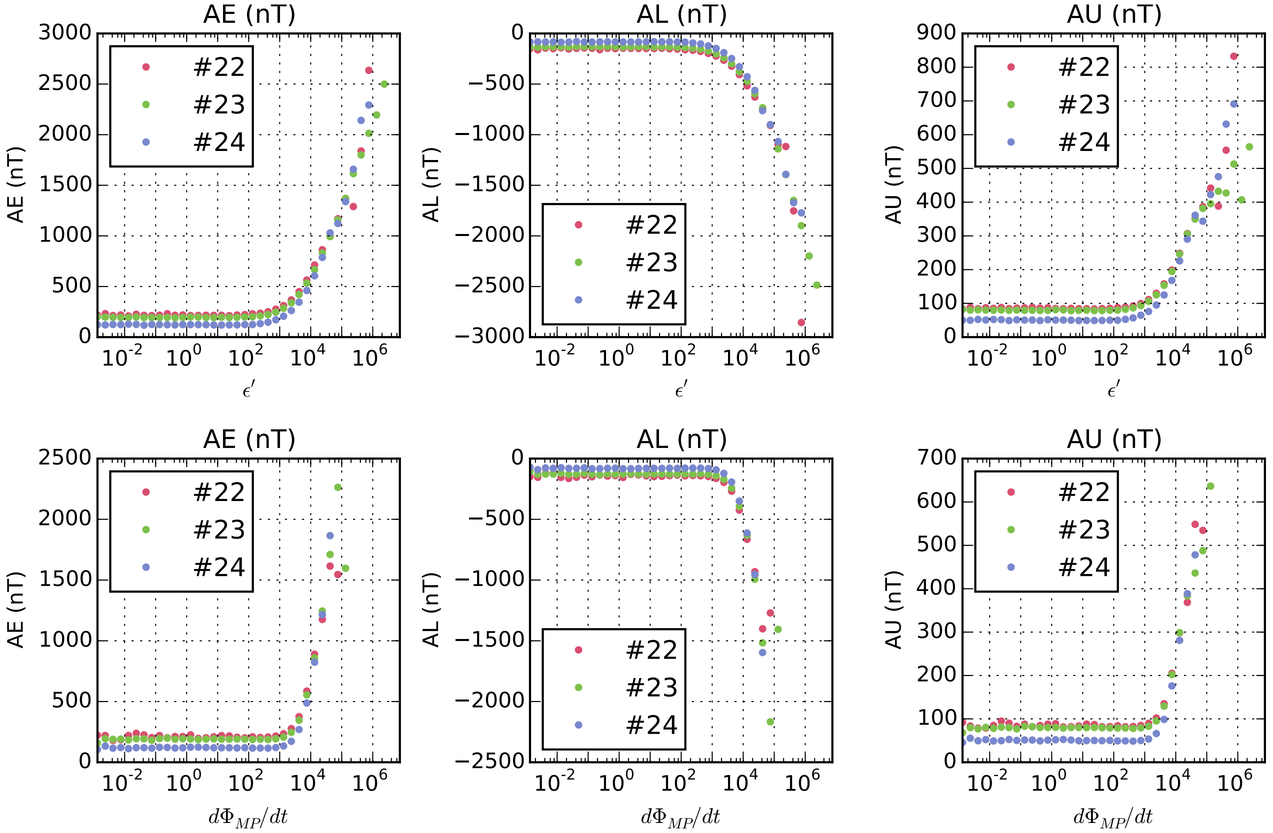


Figure 5: Correlations between the 5-minute resolution coupling functions and maximum 5-minute averaged values of geomagnetic AE indices (AE, AL, and AU) within following 1 hour (method B). The input values (x-axis) are divided into 40 evenly distributed bins, and data are subdivied into different solar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

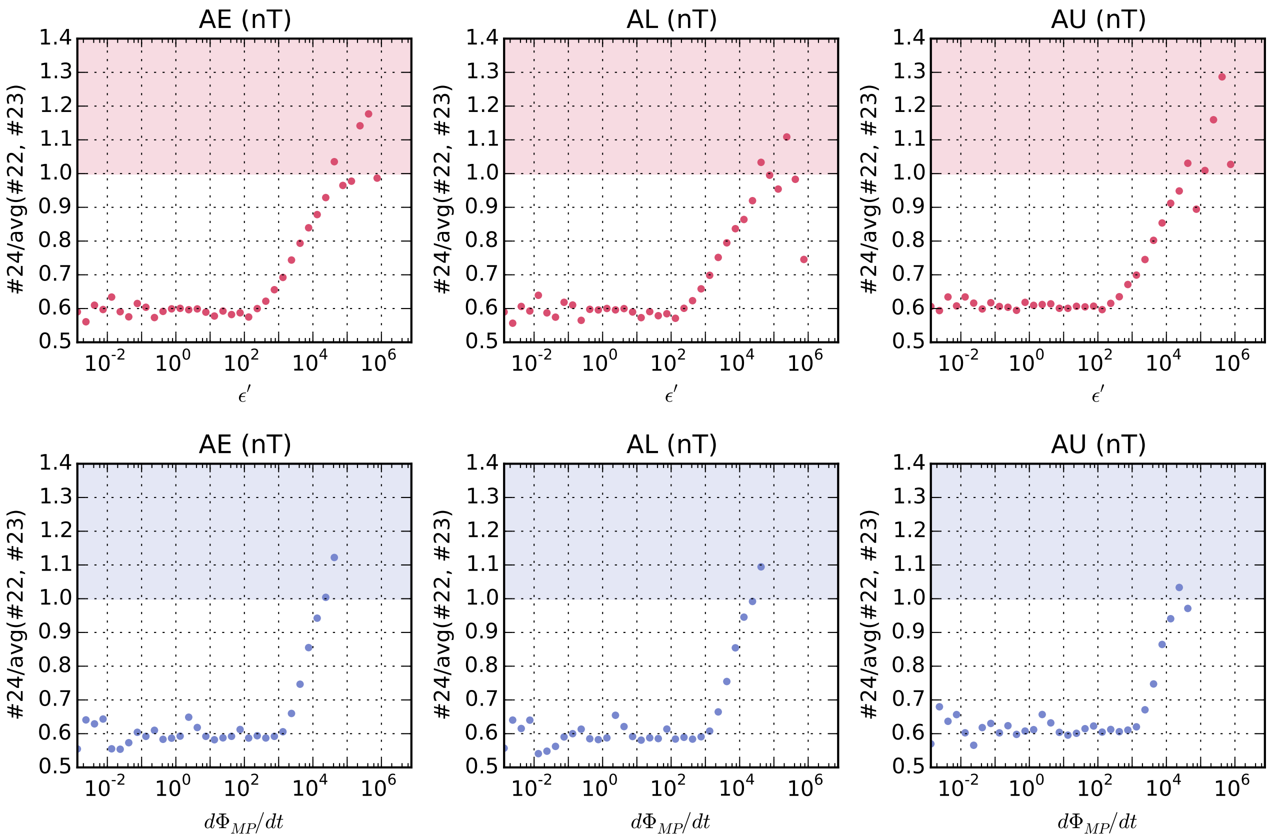


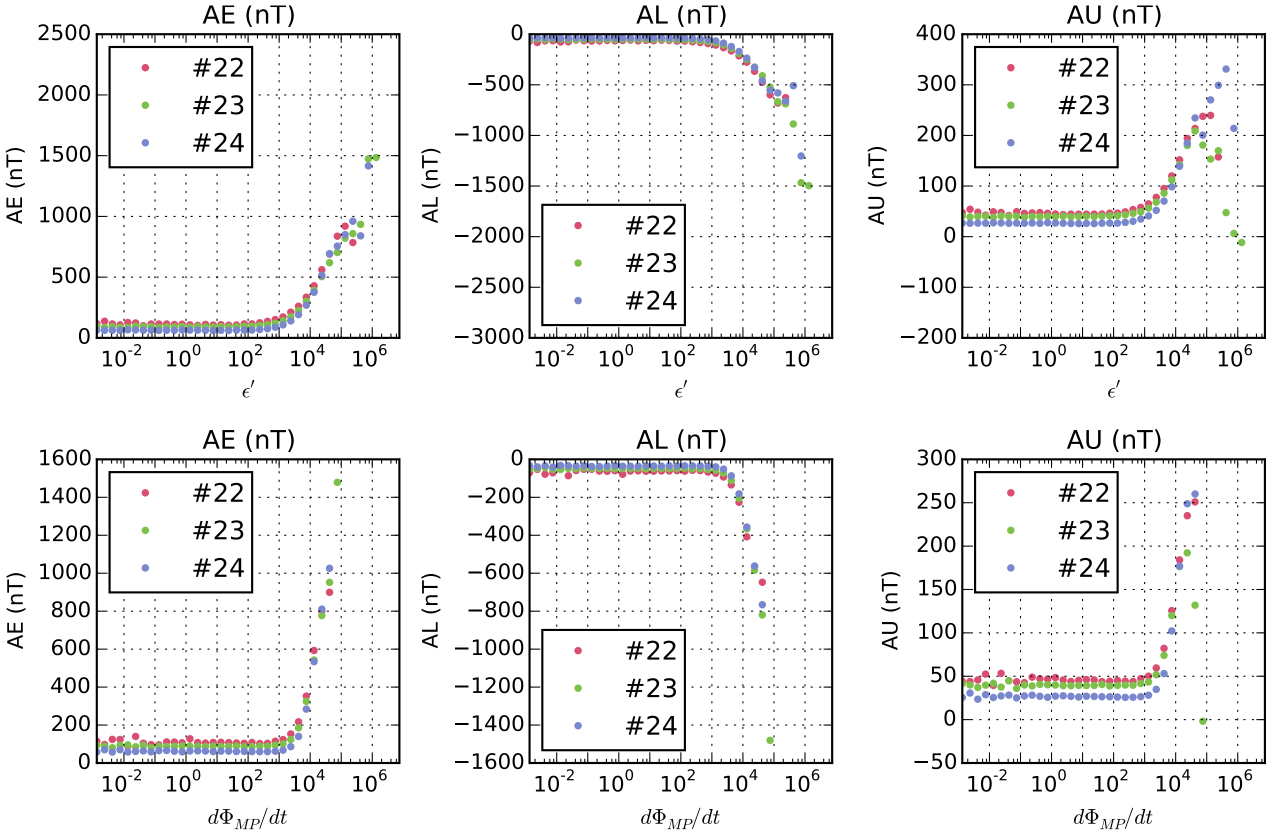
Figure 6: Cycle #24 divided by the average of #22 and #23 (method B).

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^2]) with many data points, but scattering becomes overwhelming or high wind inputs ($\epsilon' < 10^5$ [$\times$ 0.01 W/km^2]) as shown in Figure 9.

(a) Pre-solar maximum



(b) Post-solar maximum

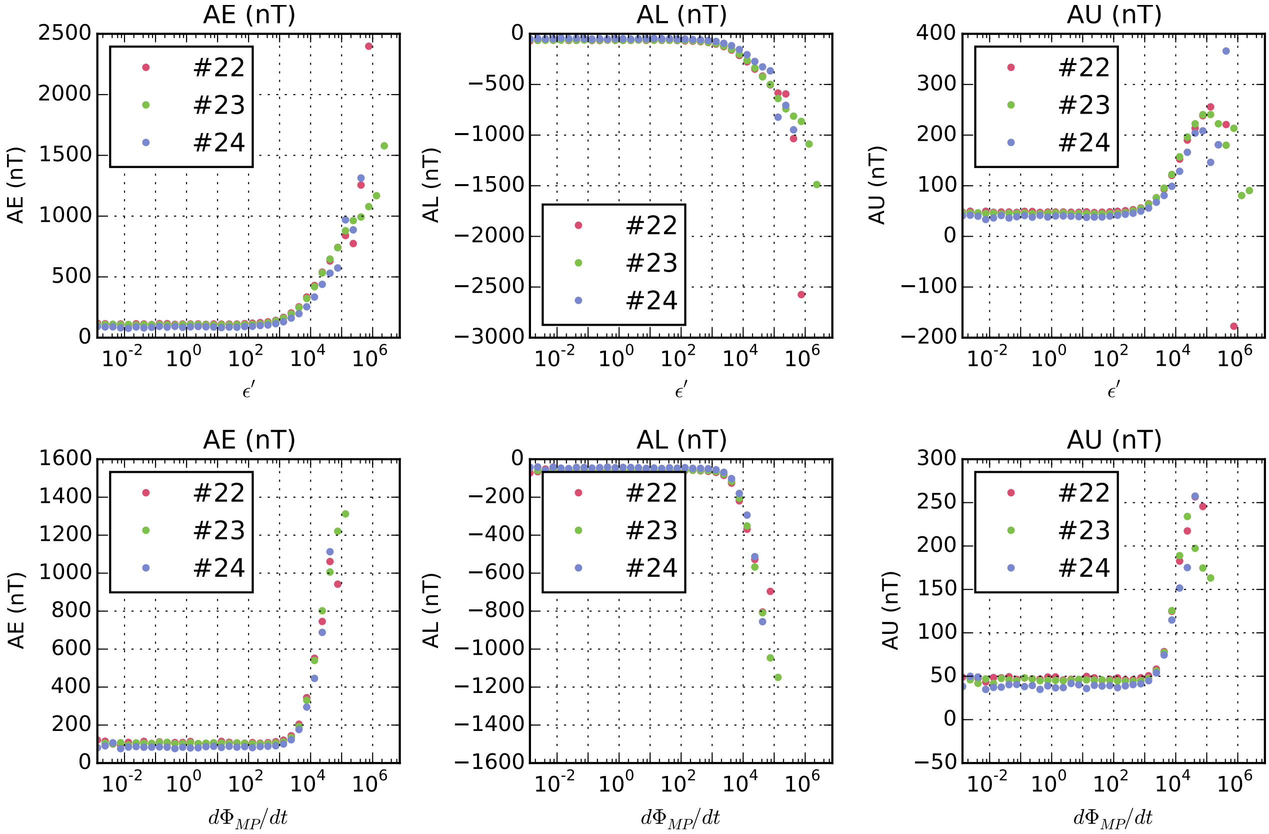
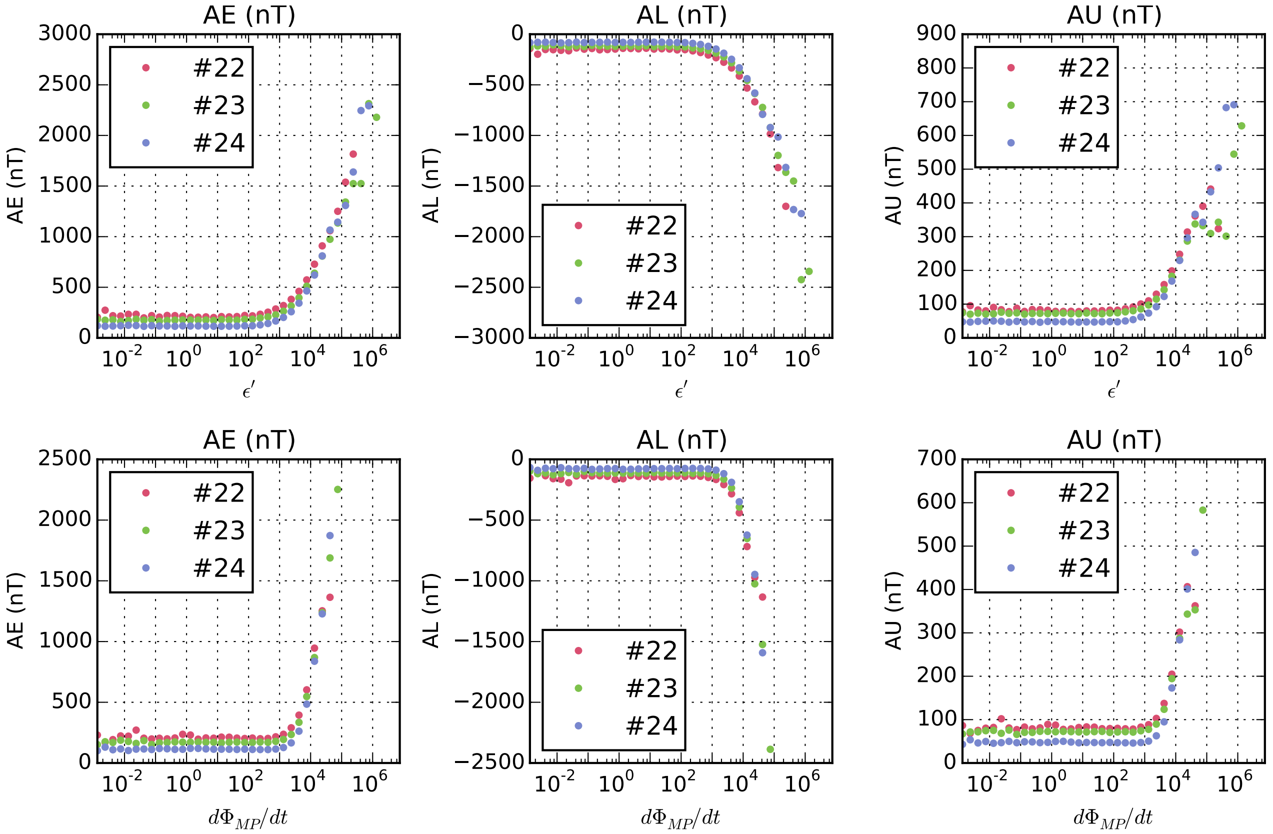


Figure 7: The same Figure 2 but further subdivided into (a) inclining phase and (b) declining phase for each solar cycle. The 5-minute delay is applied (method A).

(a) Pre-solar maximum



(b) Post-solar maximum

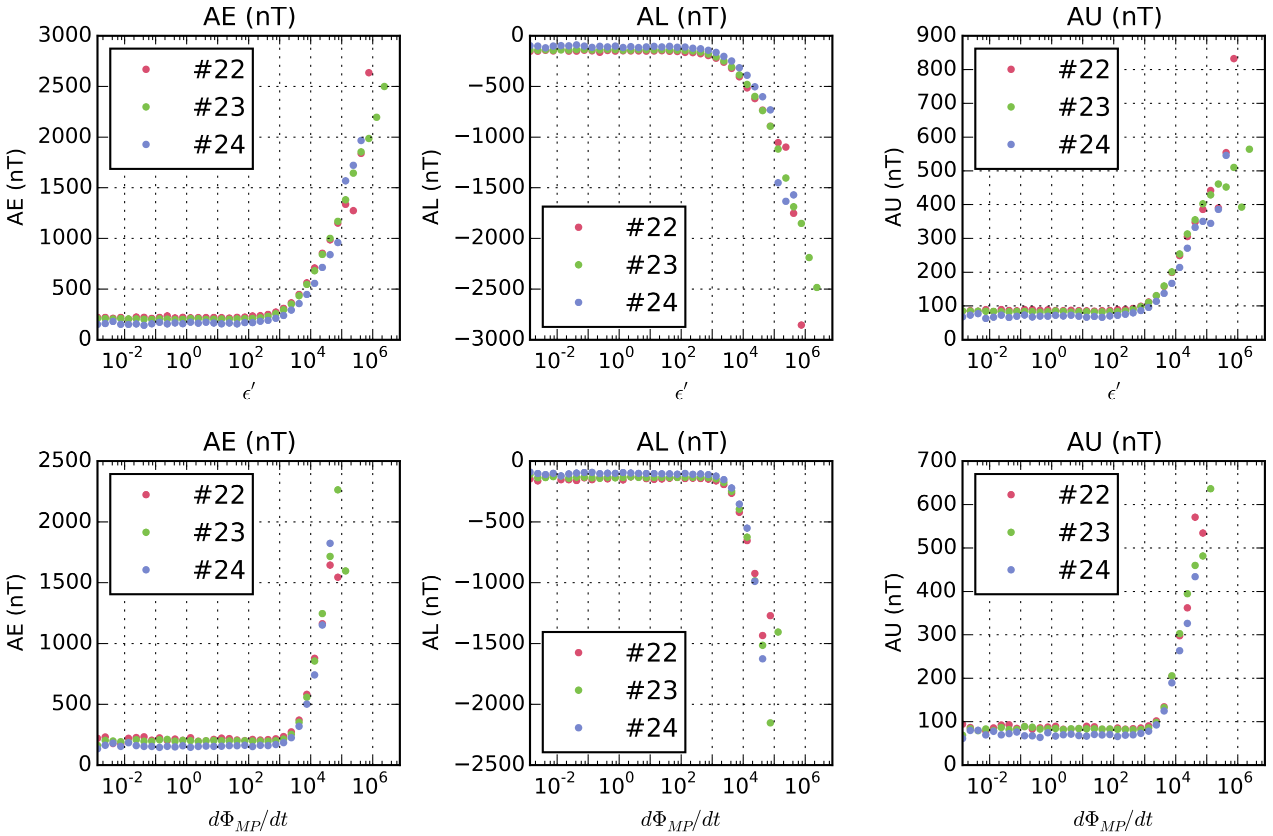
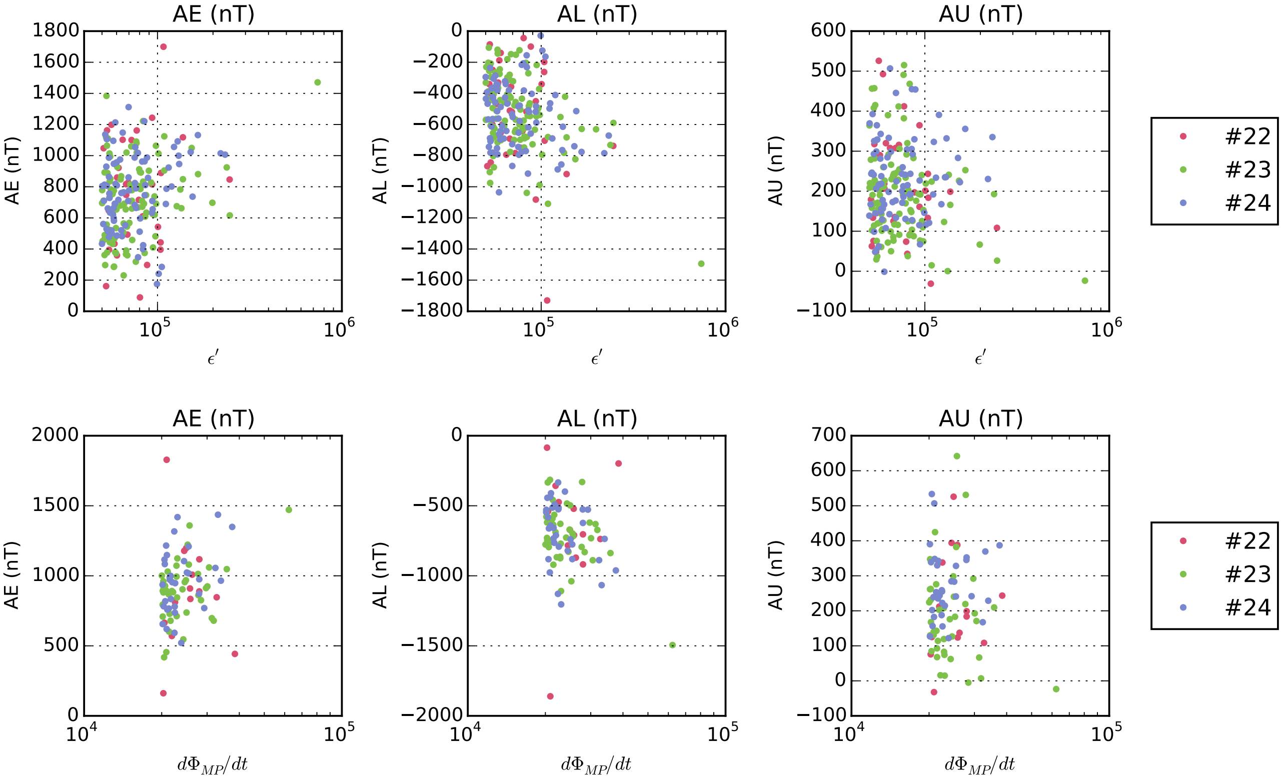


Figure 8: The same Figure 5 but further subdivided into (a) inclining phase and (b) declining phase for each solar cycle. Maximum 5-minute values of AE, AL, and AU within 1-hour is used (method B).

(a) Pre-solar maximum



(b) Post-solar maximum

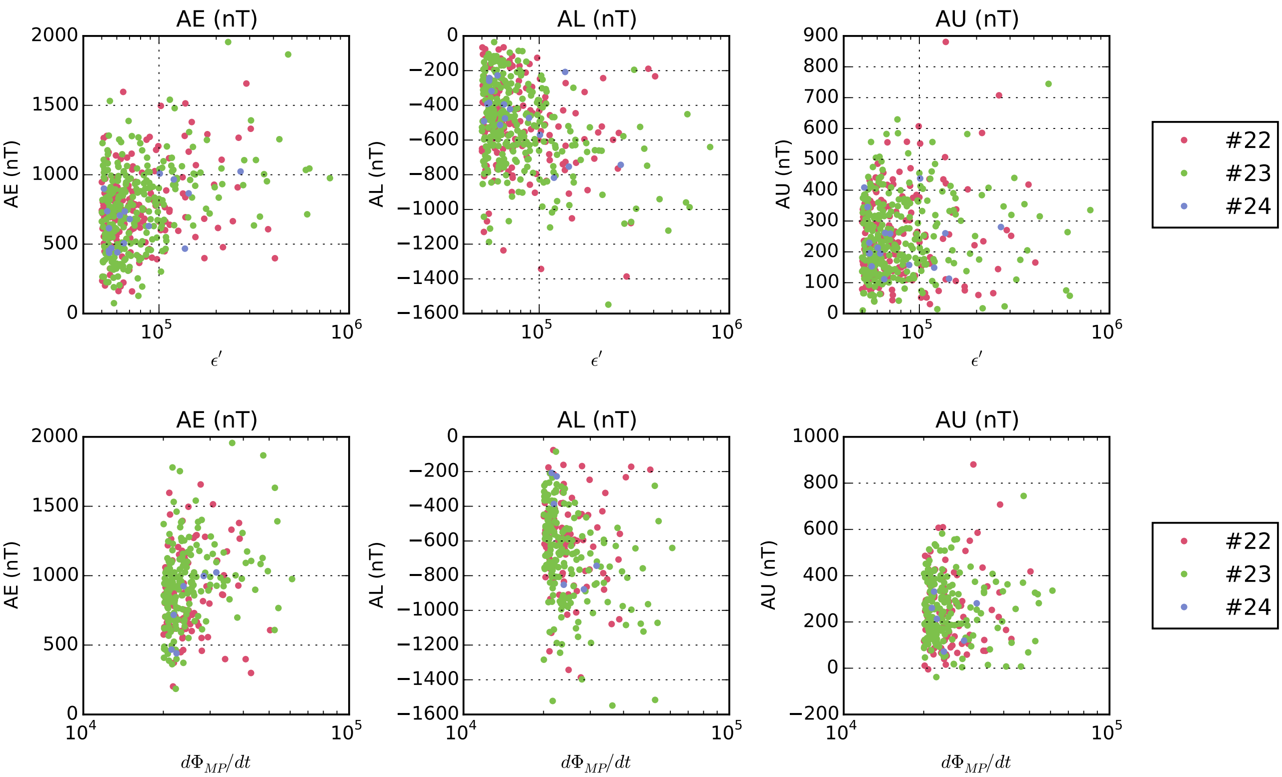


Figure 9: Scatter plot of AE, AL, AU against two different coupling functions limiting data to pre or post maximum.

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 (ϵ' 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.

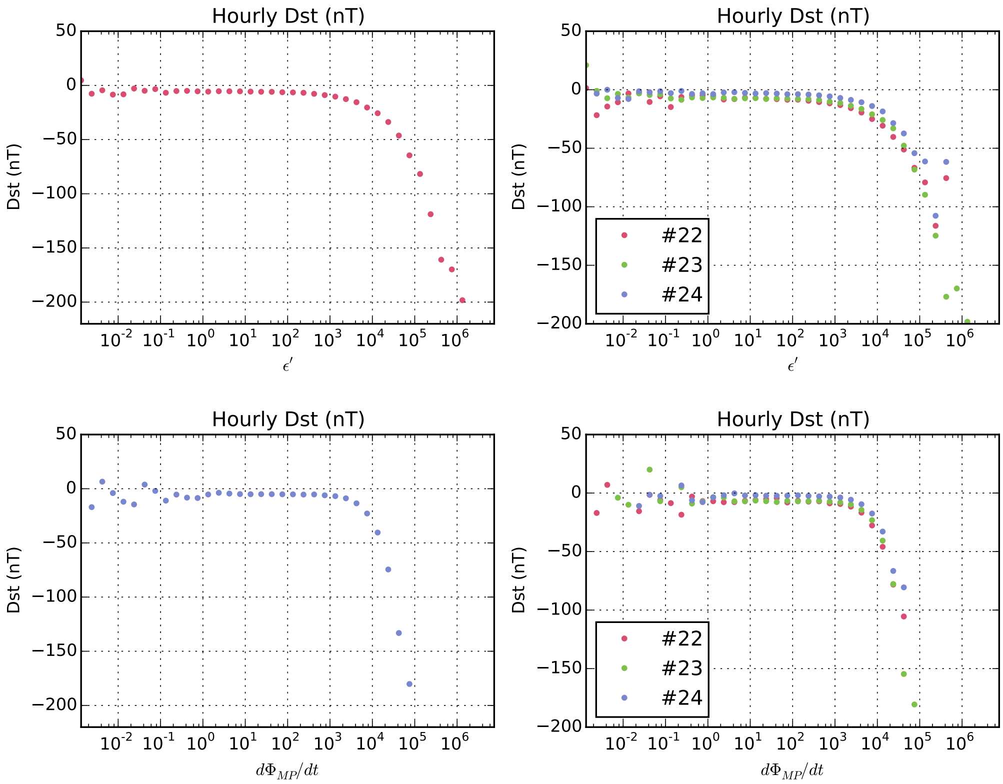
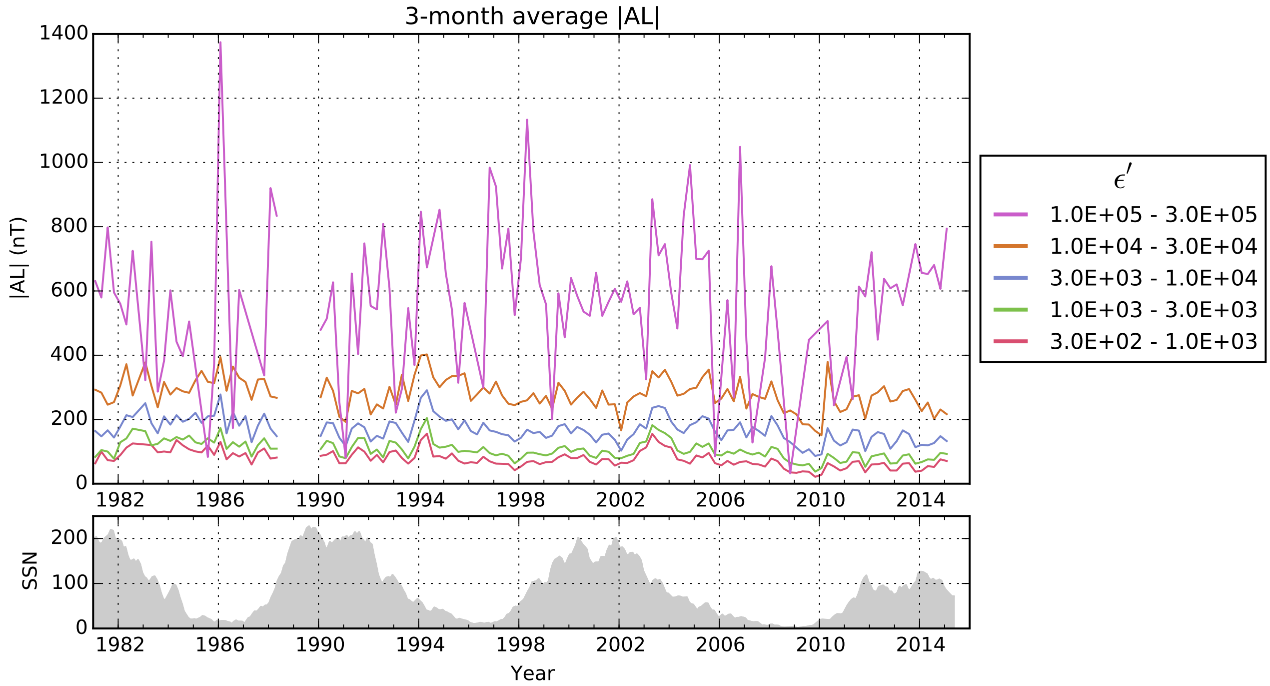


Figure 10: Hourly Dst index correlated with hourly average of ϵ' and $d\Phi_{MP}/dt$ calculated using 5-min resolution OMNI data. (similar to Method A)

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 (ϵ'). 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 (~ 1 year) increase in coupling efficiency.

(a) 3-month average



(b) 12-month running average

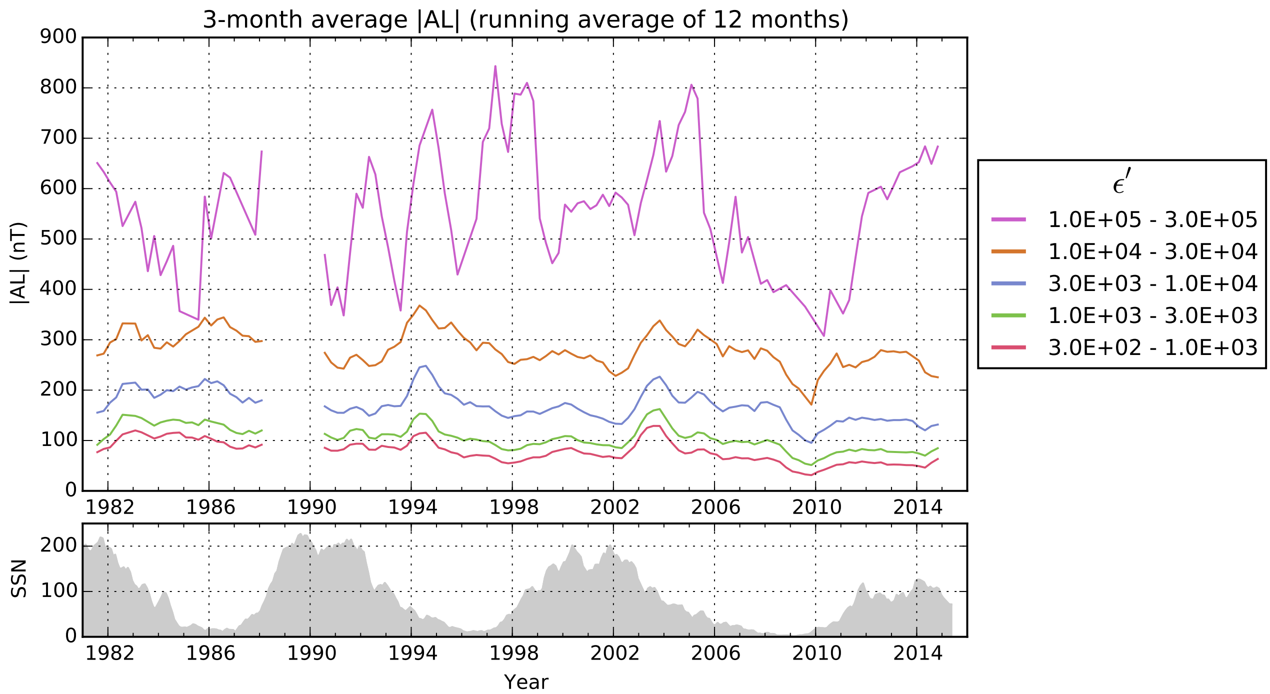


Figure 11: Variation of average |AL| for fixed ranges of ϵ' (in [0.01 W/km^2])

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 raw. In order to compare different solar cycles, it is important to consider the distribution of these events over time.

Table 3: K_p-index distribution for 1981-2015.

K_p-index =	6	6+	7-	7	7+	8-	8	8+	9-	9
1981-2015	502	315	248	160	131	100	52	45	33	9
Cycle #24	20	27	10	4	3	7	1	1	0	0

For the highest 5 levels of Kp (e.g. $Kp \geq 8-, 8, 8+, 9-, 9$), 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.

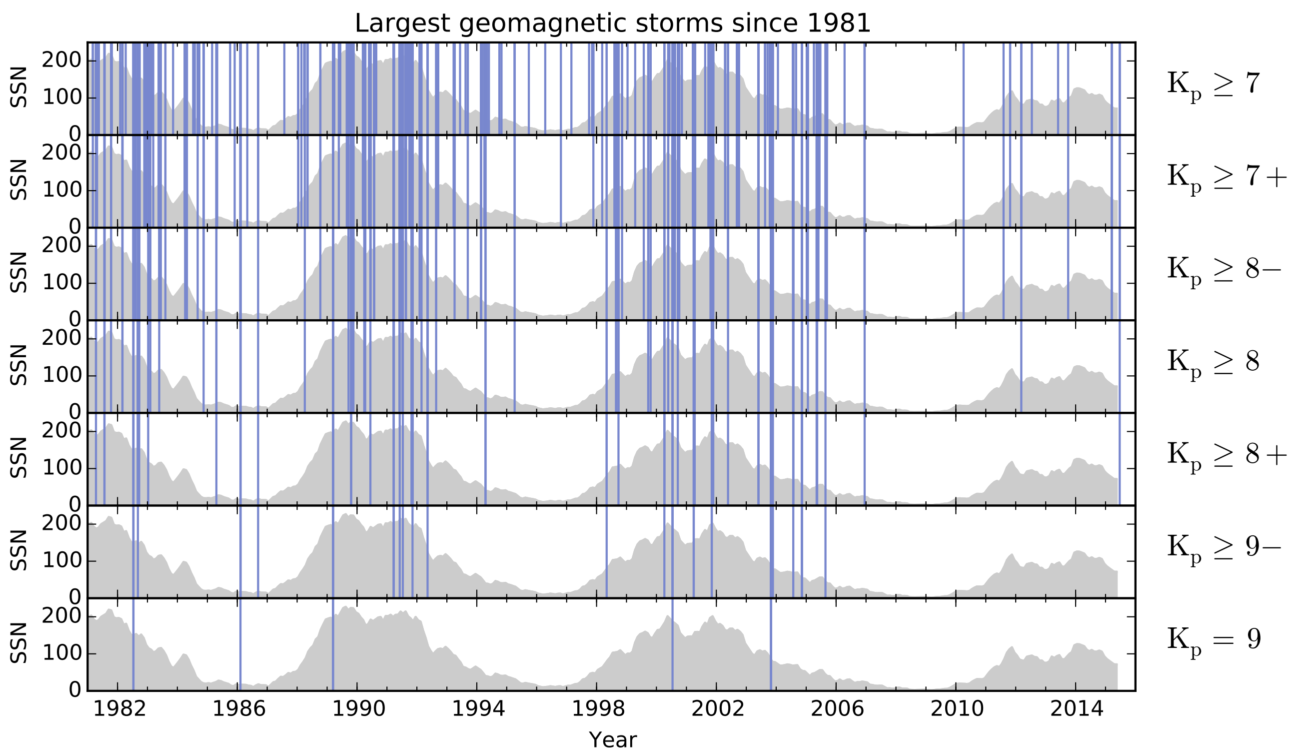


Figure 12: The occurrences of high Kp are plotted as vertical lines over the monthly sunspot number (running average of 5 months). Sunspot data is obtained from the World Data Center SILSO, Royal Observatory of Belgium, Brussels.

6.2 High ϵ' events

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

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

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

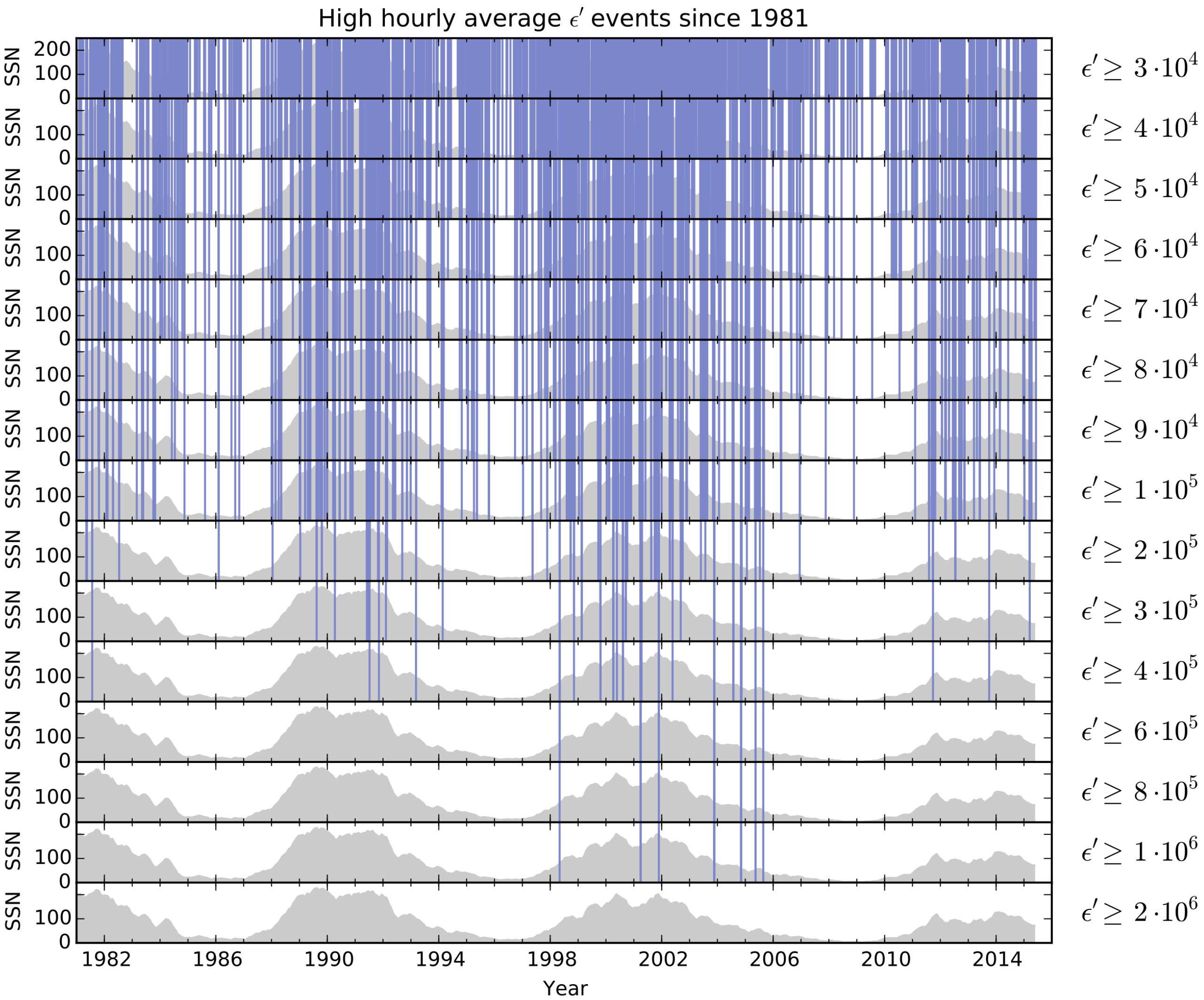


Figure 13: The same format as Figure 12 but for high ϵ' events.

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

Figure 9 shows scatter plots of the coupling functions (ϵ' 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 recomended.

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 effiency). 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 Niemegk, 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.

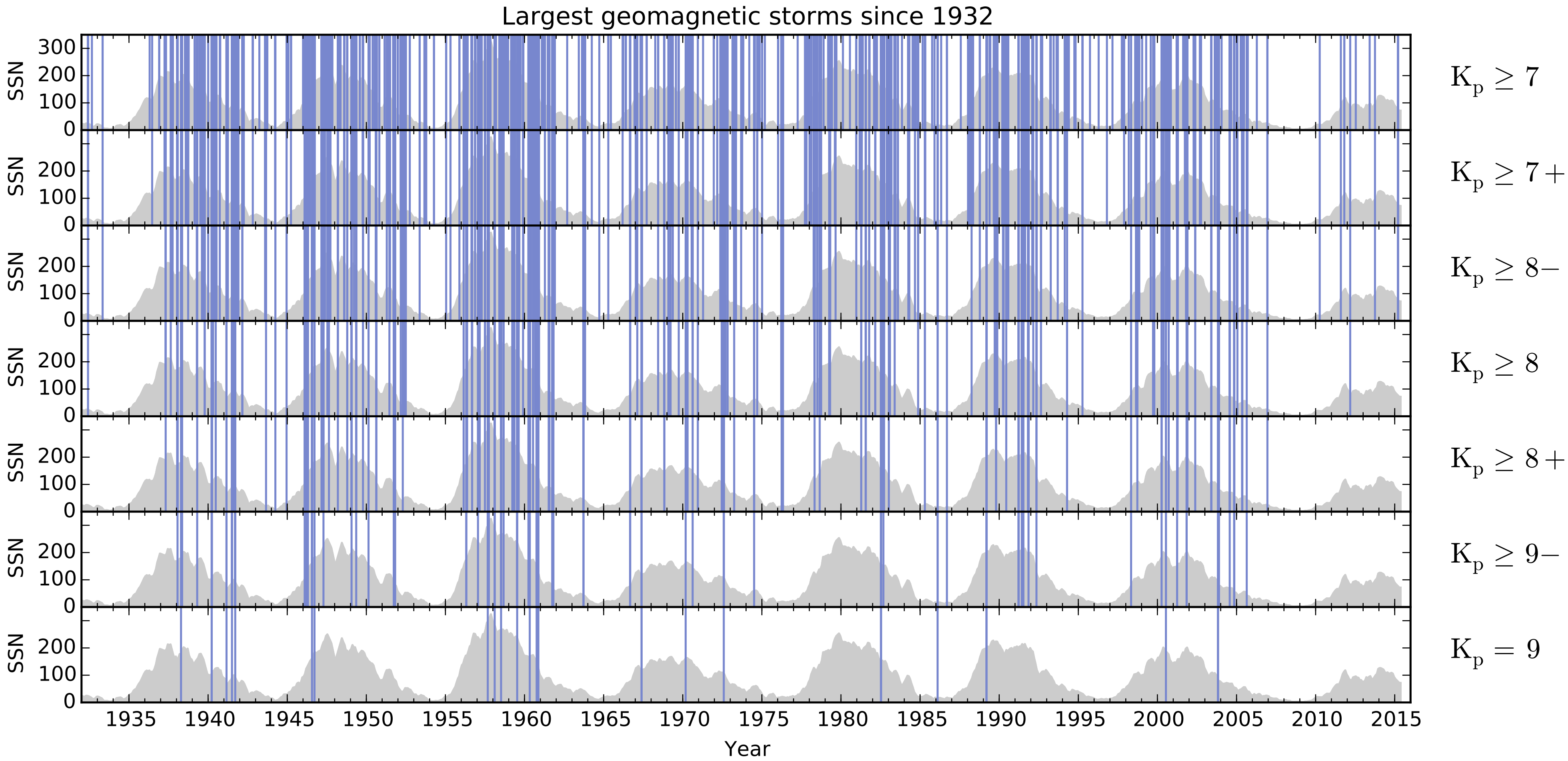


Figure 14: Largest geomagnetic storms since 1932 as marked by the K_p-index.

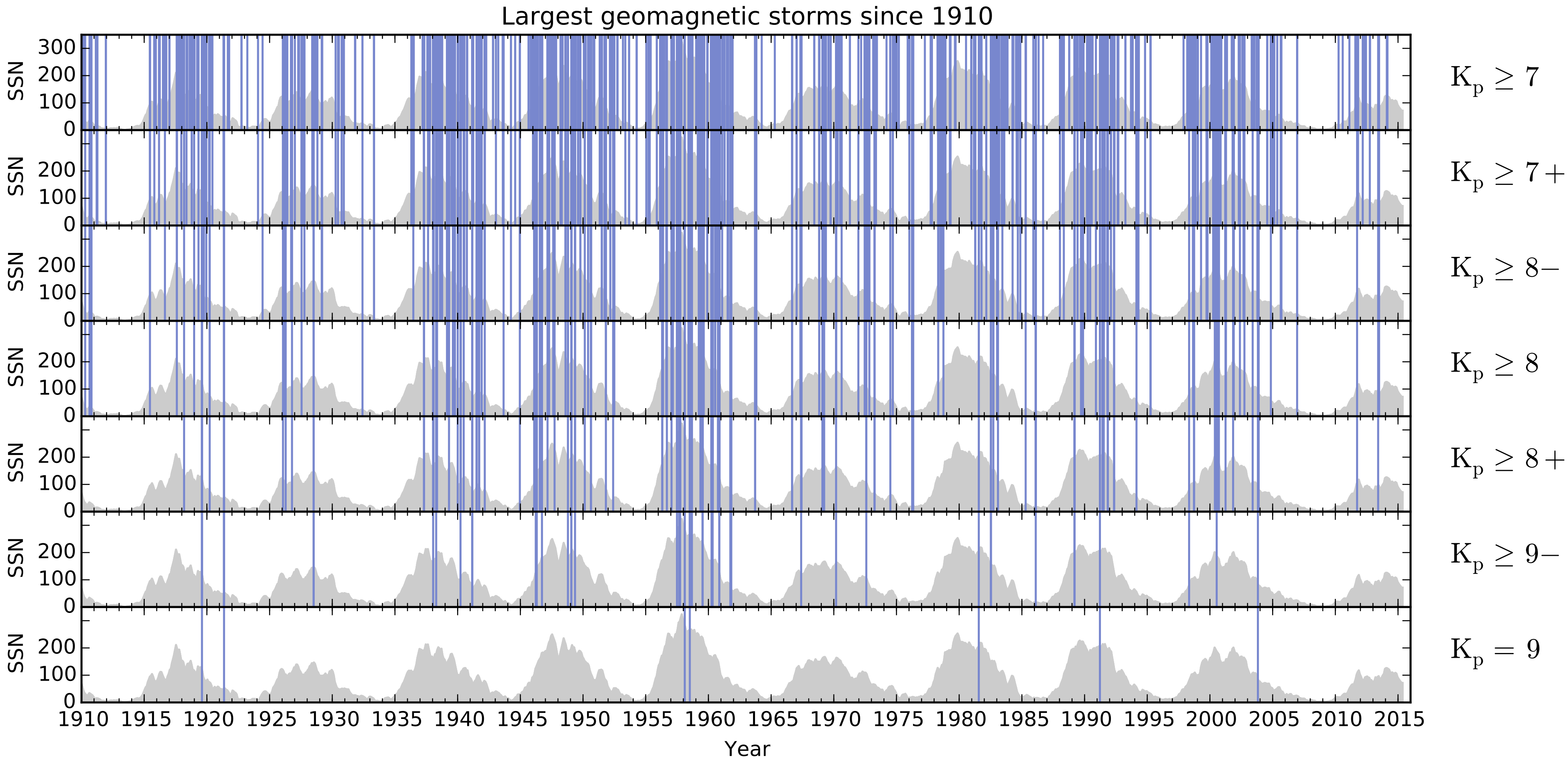


Figure 15: Largest geomagnetic storms since 1910 as marked by the K_p-equivalent index.

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---------2---------3---------4---------5---------6---------7
# Appendix-B1
# epsilon vs AE (ratio of solar cycle 24 to the previous cycles)
#                   = sunspot-kp/angeo/figure3.py (v2015)
#--------1---------2---------3---------4---------5---------6---------7
import pandas as pd
import numpy as np
colspecs = [(0, 4), # Year 1995 ... 2006
            (5, 8), # Day 1 ... 365 or 366
            (9, 11),# Hour 0 ... 23
            (12, 14), # Minute 0 ... 59 at start of average
            (92, 99), # By, nT (GSM)
            (100, 107), # Bz, nT (GSM)
            (124, 131), # Flow speed km/s
            (246, 251), # AE-index nT
            (252, 257), # AL-index nT
            (258, 263)  # AU-index NT
           ]
names = ['Year', 'Day', 'Hour', 'Minute', 'By', 'Bz', 'v', 'AE', 'AL', 'AU']
frame = pd.read_fwf('spdf.gsfc.nasa.gov/pub/data/omni/high_res_omni/omni_5min.asc', colspecs=colspecs, names=names)

frame['Bz'].replace(to_replace=9999.99, value=np.nan, inplace=True)
frame['By'].replace(to_replace=9999.99, value=np.nan, inplace=True)
frame['v'].replace(to_replace=99999.9, value=np.nan, inplace=True)
frame['AE'].replace(to_replace=99999, value=np.nan, inplace=True)
frame['AL'].replace(to_replace=99999, value=np.nan, inplace=True)
frame['AU'].replace(to_replace=99999, value=np.nan, inplace=True)

frame['Datetime'] = frame['Year'] * 10000000 + frame['Day'] * 10000 + frame['Hour'] * 100 + frame['Minute']
frame['Datetime'] = pd.to_datetime(frame['Datetime'], format='%Y%j%H%M')
frame.set_index('Datetime', inplace=True)
#frame.head()

d = frame.reset_index()
for index in range(len(d)):
    indices = d.loc[index:index+29][['AL', 'AE', 'AU']]
    d.set_value(index, 'AL', indices['AL'].min())
    d.set_value(index, 'AE', indices['AE'].max())
    d.set_value(index, 'AU', indices['AU'].max())

d.to_csv('data/omni_5min_highest_hourly_AEALAU.csv')

# You can skip the slow part above and load the cache/CSV file if available
import pandas as pd
frame = pd.read_csv('data/omni_5min_highest_hourly_AEALAU.csv')

import numpy as np
original_form = False

B_T = np.sqrt(frame['By'] ** 2 + frame['Bz'] ** 2)

if original_form:
    theta = np.arctan2(frame['By'], frame['Bz'])
    frame['akasofu'] = frame['v'] * B_T ** 2 * np.sin(theta/2) ** 4
```

```
    frame['newell'] = frame['v'] ** (4/3) * B_T ** (2/3) * np.sin(theta/2) ** (8/3)
else:
    frame['akasofu'] = frame['v'] * ((B_T - frame['Bz']) ** 2) / 4
    frame['newell'] = ( frame['v'] ** 2 * ((B_T - frame['Bz']) ** 2 / (4*B_T)) ) ** (2/3)

frame['Datetime'] = pd.to_datetime(frame['Datetime'])
frame = frame.set_index('Datetime')

import numpy as np
import pandas as pd
%matplotlib inline

import matplotlib.pyplot as plt

n_bands = 40
b = np.logspace(-3, 7, n_bands*2+1)

d = frame[['akasofu', 'newell', 'AE', 'AL', 'AU']]

def logarithmic_bands(data, coupling_function):
    bands = []
    for i in range(0, n_bands*2, 2):
        band_left = b[i]
        band_center = b[i+1]
        band_right = b[i+2]
        condition1 = data[coupling_function] >= band_left
        condition2 = data[coupling_function] < band_right
        conditions = condition1 & condition2
        AE_mean = data['AE'][conditions].mean()
        AL_mean = data['AL'][conditions].mean()
        AU_mean = data['AU'][conditions].mean()
        bands.append([band_center, AE_mean, AL_mean, AU_mean])
    return bands

akasofu_bands = logarithmic_bands(d, 'akasofu')
newell_bands = logarithmic_bands(d, 'newell')

# Plotting
# Paper plot
bands_frame = {}
for cycle_number in ['22', '23', '24']:
    # Before maximum
    condition1 = d.index > cycle[cycle_number][0]
    condition2 = d.index < cycle_maxima[cycle_number]

    cycle_frame = d[condition1 & condition2]
    akasofu_bands = logarithmic_bands(cycle_frame, 'akasofu')
    bands_frame[cycle_number] = pd.DataFrame(akasofu_bands).set_index(0)
    bands_frame[cycle_number].columns = ['AE', 'AL', 'AU']
#bands_22_23 = (bands_frame['22'] + bands_frame['23']) / 2


fig = plt.figure(figsize=(8.5, 5.8))
ax1 = fig.add_subplot(231)
```

```python
ax2 = fig.add_subplot(232)
ax3 = fig.add_subplot(233)
ax4 = fig.add_subplot(234)
ax5 = fig.add_subplot(235)
ax6 = fig.add_subplot(236)

bands_fraction = bands_frame['24'] / bands_frame['22']
bands_fraction.plot(style='.', y='AE', grid=True, ax=ax1, logx=True, color=colors[0], legend=None)
bands_fraction.plot(style='.', y='AL', grid=True, ax=ax2, logx=True, color=colors[0], legend=None)
bands_fraction.plot(style='.', y='AU', grid=True, ax=ax3, logx=True, color=colors[0], legend=None)

bands_fraction = bands_frame['23'] / bands_frame['22']
bands_fraction.plot(style='.', y='AE', grid=True, ax=ax1, logx=True, color=colors[2], legend=None)
bands_fraction.plot(style='.', y='AL', grid=True, ax=ax2, logx=True, color=colors[2], legend=None)
bands_fraction.plot(style='.', y='AU', grid=True, ax=ax3, logx=True, color=colors[2], legend=None)

bands_frame = {}
for cycle_number in ['22', '23', '24']:
    # After maximum
    condition1 = d.index > cycle_maxima[cycle_number]
    condition2 = d.index < cycle[cycle_number][1]

    cycle_frame = d[condition1 & condition2]
    akasofu_bands = logarithmic_bands(cycle_frame, 'akasofu')
    bands_frame[cycle_number] = pd.DataFrame(akasofu_bands).set_index(0)
    bands_frame[cycle_number].columns = ['AE', 'AL', 'AU']


bands_fraction = bands_frame['24'] / bands_frame['22']
bands_fraction.plot(style='.', y='AE', grid=True, ax=ax4, logx=True, color=colors[0], legend=None)
bands_fraction.plot(style='.', y='AL', grid=True, ax=ax5, logx=True, color=colors[0], legend=None)
bands_fraction.plot(style='.', y='AU', grid=True, ax=ax6, logx=True, color=colors[0], legend=None)

bands_fraction = bands_frame['23'] / bands_frame['22']
bands_fraction.plot(style='.', y='AE', grid=True, ax=ax4, logx=True, color=colors[2], legend=None)
bands_fraction.plot(style='.', y='AL', grid=True, ax=ax5, logx=True, color=colors[2], legend=None)
bands_fraction.plot(style='.', y='AU', grid=True, ax=ax6, logx=True, color=colors[2], legend=None)


for ax in [ax1, ax2, ax3, ax4, ax5, ax6]:
    ax.set_xlabel('$\epsilon\'$')
    ax.set_ylim([.4, 1.6])
    ax.axhspan(1.0, 1.6, color=colors[0], alpha=0.2)

for ax in [ax1, ax4]:
    ax.set_title('AE (nT)')
for ax in [ax2, ax5]:
    ax.set_title('AL (nT)')
for ax in [ax3, ax6]:
    ax.set_title('AU (nT)')
for ax in [ax1,ax2,ax3,ax4,ax5,ax6]:
    ax.set_ylabel('Ratio')

for ax in [ax1,ax2,ax3, ax4, ax5, ax6]:
```

```
    lines, labels = ax.get_legend_handles_labels()
    ax.legend(lines[::-1], ['#24', '#23'][::-1], loc='upper left', numpoints=1)

plt.tight_layout()

for ax in [ax1,ax2,ax3,ax4,ax5,ax6]:
    xticks = ax.xaxis.get_major_ticks()
    for i in range(1, len(xticks), 2):
        xticks[i].label1.set_visible(False)
    #ax.set_xlim([10**1, 10**6])

fig.subplots_adjust(hspace=0.4)

fig.savefig('Pre_Post_Maximum_OMNI_akasofu_newell_AE_AL_AU_cycle_paper.eps')
#--------1---------2---------3---------4---------5---------6---------7
# Appendix-B2
# scatter plot of epsilon vs AE (only for large epsilon)
#                   = sunspot-kp/angeo/figure4.py (v2015)
#--------1---------2---------3---------4---------5---------6---------7
import pandas as pd
method = 'hourly' #'hourly'

# This CSV file has already been produced in figure3.py
frame = pd.read_csv('data/omni_5min_highest_'+method+'_AEALAU.csv')


# Calculate akasofu
import numpy as np
original_form = False

B_T = np.sqrt(frame['By'] ** 2 + frame['Bz'] ** 2)

if original_form:
    theta = np.arctan2(frame['By'], frame['Bz'])
    frame['akasofu'] = frame['v'] * B_T ** 2 * np.sin(theta/2) ** 4
    frame['newell'] = frame['v'] ** (4/3) * B_T ** (2/3) * np.sin(theta/2) ** (8/3)
else:
    frame['akasofu'] = frame['v'] * ((B_T - frame['Bz']) ** 2) / 4
    frame['newell'] = ( frame['v'] ** 2 * ((B_T - frame['Bz']) ** 2 / (4*B_T)) ) ** (2/3)

frame['akasofu'] = frame['akasofu'] / 100 # to W/km^2


# Convert datetime index to datetime objects
frame['Datetime'] = pd.to_datetime(frame['Datetime'])
frame = frame.set_index('Datetime')


# Define logarithmic bands function
import numpy as np
import pandas as pd
%matplotlib inline

import matplotlib.pyplot as plt
```

```python
n_bands = 40
b = np.logspace(-3-2, 7-2, n_bands*2+1)

d = frame[['akasofu', 'newell', 'AE', 'AL', 'AU']]

def logarithmic_bands(data, coupling_function):
    bands = []
    for i in range(0, n_bands*2, 2):
        band_left = b[i]
        band_center = b[i+1]
        band_right = b[i+2]
        condition1 = data[coupling_function] >= band_left
        condition2 = data[coupling_function] < band_right
        conditions = condition1 & condition2
        AE_mean = data['AE'][conditions].mean()
        AL_mean = data['AL'][conditions].mean()
        AU_mean = data['AU'][conditions].mean()
        bands.append([band_center,
                      AE_mean, AL_mean, AU_mean,
                      np.count_nonzero(conditions)])#/np.count_nonzero(~np.isnan(data[coupling_function]))])
    return bands


# Prepare dataframe (delay is optional and set to 0
#                    because we are using previously calculated peak hourly values directly)
delay = 0
akasofu = frame['akasofu'].values#[:-delay]
newell = frame['newell'].values#[:-delay]
AE = frame['AE'].values#[delay:]
AL = frame['AL'].values#[delay:]
AU = frame['AU'].values#[delay:]

d = pd.concat([pd.Series(frame.index.values), pd.Series(akasofu), pd.Series(newell), pd.Series(AE), pd.Series(AL),
               pd.Series(AU)], axis=1)
d.columns = ['Datetime', 'akasofu', 'newell', 'AE', 'AL', 'AU']


# Remove minute/second information (and average per hour)
d['Datetime'] = d['Datetime'].apply(lambda x: x.replace(minute=0, second=0))
d = d.groupby('Datetime').mean()


# Plot pre-maximum
%matplotlib inline
import matplotlib.pyplot as plt
from datetime import datetime, timedelta
colors = ["#D94E70",
"#7DC14B",
"#7887CE",
"#D4752C",
"#CA5DCA"]

fig = plt.figure(figsize=(8.5, 5.8*1.5))
```

```
ax1 = fig.add_subplot(231)
ax2 = fig.add_subplot(232)
ax3 = fig.add_subplot(233)
ax4 = fig.add_subplot(234)
ax5 = fig.add_subplot(235)
ax6 = fig.add_subplot(236)
axes = [ax1,ax2,ax3,ax4,ax5,ax6]


cycle = {}
cycle['21post'] = pd.to_datetime(['01-1800', '09-1986'], format='%m-%Y')
cycle['22pre'] = pd.to_datetime(['09-1986', '03-1991'], format='%m-%Y')
cycle['22post'] = pd.to_datetime(['03-1991', '08-1996'], format='%m-%Y')
cycle['23pre'] = pd.to_datetime(['08-1996', '12-2001'], format='%m-%Y')
cycle['23post'] = pd.to_datetime(['12-2001', '12-2008'], format='%m-%Y')
cycle['24pre'] = pd.to_datetime(['12-2008', '05-2014'], format='%m-%Y')
cycle['24post'] = pd.to_datetime(['05-2014', '01-2020'], format='%m-%Y')

cycle_colors = {}
cycle_colors['21'] = '#363636'
cycle_colors['22'] = colors[2]
cycle_colors['23'] = colors[1]
cycle_colors['24'] = colors[0]

for cycle_number in ['22', '23', '24']:
    condition1 = d.index >= cycle[cycle_number+'pre'][0]
    #condition1 = d.index > cycle_maxima[cycle_number]
    #condition2 = d.index < cycle[cycle_number][1]
    condition2 = d.index < cycle[cycle_number+'pre'][1]
    cycle_frame = d[condition1 & condition2]
    high_condition = cycle_frame['akasofu'] >= 5 * 10 ** 2
    print('Cycle number', cycle_number, 'has', np.count_nonzero(high_condition), 'events')
    #print(cycle_frame[high_condition])
    last_dt = None
    consecutive_events = []
    for dt, row in cycle_frame[high_condition].iterrows():
        if last_dt != (dt - timedelta(hours=1)):
            # Add new group of consecutive values
            consecutive_events.append([])

        consecutive_events[len(consecutive_events)-1].append(row)
        last_dt = dt
    print('Consecutive events:', len(consecutive_events))

    for i, index in enumerate(['AE', 'AL', 'AU']):
        datapoints = []
        for consecutive_event in consecutive_events:
            ak = np.mean([row['akasofu'] for row in consecutive_event])
            y = np.mean([row[index] for row in consecutive_event])
            datapoints.append([ak, y])
        events_scatter = pd.DataFrame(datapoints)
        events_scatter.columns = ['akasofu', index+cycle_number]
        #print(events_scatter.head())
        events_scatter.plot(style='.', x='akasofu', y=index+cycle_number, grid=True, ax=axes[i], logx=True, color=cycle_colors[cycle_number])
```

```
    high_condition = cycle_frame['newell'] >= 2 * 10 ** 4
    print('Cycle number', cycle_number, 'has', np.count_nonzero(high_condition), 'events')
    #print(cycle_frame[high_condition])
    last_dt = None
    consecutive_events = []
    for dt, row in cycle_frame[high_condition].iterrows():
        if last_dt != (dt - timedelta(hours=1)):
            # Add new group of consecutive values
            consecutive_events.append([])

        consecutive_events[len(consecutive_events)-1].append(row)
        last_dt = dt
    print('Consecutive events:', len(consecutive_events))

    for i, index in enumerate(['AE', 'AL', 'AU']):
        datapoints = []
        for consecutive_event in consecutive_events:
            ne = np.mean([row['newell'] for row in consecutive_event])
            y = np.mean([row[index] for row in consecutive_event])
            datapoints.append([ne, y])
        events_scatter = pd.DataFrame(datapoints)
        events_scatter.columns = ['newell', index+cycle_number]
        #print(events_scatter.head())
        events_scatter.plot(style='.', x='newell', y=index+cycle_number, grid=True, ax=axes[i+3], logx=True, color=cycle_colors[cycle_number])


for ax in [ax1,ax2, ax4, ax5]:
    #lines, labels = ax.get_legend_handles_labels()
    #ax.legend(lines, ['#22', '#23', '#24'], loc='best', numpoints=1)
    ax.legend().remove()

legend = []
for ax in [ax3, ax6]:
    lines, labels = ax.get_legend_handles_labels()
    legend.append(ax.legend(lines, ['#22', '#23', '#24'], numpoints=1, loc='center left', bbox_to_anchor=(1.12, 0.5)))


for ax in [ax1, ax2, ax3]:
    ax.set_xlabel('$\epsilon\'\/\mathrm{[W/km^2]}$')
for ax in [ax4, ax5, ax6]:
    ax.set_xlabel('$d\Phi_{MP}/dt$')

for ax in [ax1, ax4]:
    ax.set_ylabel('AE (nT)')
    ax.set_title('AE (nT)')
for ax in [ax2, ax5]:
    ax.set_ylabel('AL (nT)')
    ax.set_title('AL (nT)')
for ax in [ax3, ax6]:
    ax.set_ylabel('AU (nT)')
    ax.set_title('AU (nT)')

plt.tight_layout()
```

```
for ax in [ax1,ax2,ax3,ax4,ax5,ax6]:
    #xticks = ax.xaxis.get_major_ticks()
    #for i in range(1, len(xticks), 2):
    #    xticks[i].label1.set_visible(False)
    ax.set_xlim([4*10**2, 10**4])
for ax in [ax4,ax5,ax6]:
    ax.set_xlim([1*10**4, 10**5])

fig.savefig('Pre-maximum-high-epsilon-scatter.eps', bbox_extra_artists=(legend[0],legend[1],), bbox_inches='tight')


# Plot post-maximum
%matplotlib inline
import matplotlib.pyplot as plt
from datetime import datetime, timedelta
colors = ["#D94E70",
"#7DC14B",
"#7887CE",
"#D4752C",
"#CA5DCA"]

fig = plt.figure(figsize=(8.5, 5.8*1.5))
ax1 = fig.add_subplot(231)
ax2 = fig.add_subplot(232)
ax3 = fig.add_subplot(233)
ax4 = fig.add_subplot(234)
ax5 = fig.add_subplot(235)
ax6 = fig.add_subplot(236)
axes = [ax1,ax2,ax3,ax4,ax5,ax6]


cycle = {}
cycle['21post'] = pd.to_datetime(['01-1800', '09-1986'], format='%m-%Y')
cycle['22pre'] = pd.to_datetime(['09-1986', '03-1991'], format='%m-%Y')
cycle['22post'] = pd.to_datetime(['03-1991', '08-1996'], format='%m-%Y')
cycle['23pre'] = pd.to_datetime(['08-1996', '12-2001'], format='%m-%Y')
cycle['23post'] = pd.to_datetime(['12-2001', '12-2008'], format='%m-%Y')
cycle['24pre'] = pd.to_datetime(['12-2008', '05-2014'], format='%m-%Y')
cycle['24post'] = pd.to_datetime(['05-2014', '01-2020'], format='%m-%Y')

cycle_colors = {}
cycle_colors['21'] = '#363636'
cycle_colors['22'] = colors[2]
cycle_colors['23'] = colors[1]
cycle_colors['24'] = colors[0]

for cycle_number in ['21', '22', '23', '24']:
    condition1 = d.index >= cycle[cycle_number+'post'][0]
    #condition1 = d.index > cycle_maxima[cycle_number]
    #condition2 = d.index < cycle[cycle_number][1]
    condition2 = d.index < cycle[cycle_number+'post'][1]
    cycle_frame = d[condition1 & condition2]
    high_condition = cycle_frame['akasofu'] >= 5 * 10 ** 2
    print('Cycle number', cycle_number, 'has', np.count_nonzero(high_condition), 'events')
```

```
    #print(cycle_frame[high_condition])
    last_dt = None
    consecutive_events = []
    for dt, row in cycle_frame[high_condition].iterrows():
        if last_dt != (dt - timedelta(hours=1)):
            # Add new group of consecutive values
            consecutive_events.append([])

        consecutive_events[len(consecutive_events)-1].append(row)
        last_dt = dt
    print('Consecutive events:', len(consecutive_events))

    for i, index in enumerate(['AE', 'AL', 'AU']):
        datapoints = []
        for consecutive_event in consecutive_events:
            ak = np.mean([row['akasofu'] for row in consecutive_event])
            y = np.mean([row[index] for row in consecutive_event])
            datapoints.append([ak, y])
        events_scatter = pd.DataFrame(datapoints)
        events_scatter.columns = ['akasofu', index+cycle_number]
        #print(events_scatter.head())
        events_scatter.plot(style='.', x='akasofu', y=index+cycle_number, grid=True, ax=axes[i], logx=True, color=cycle_colors[cycle_number])

    high_condition = cycle_frame['newell'] >= 2 * 10 ** 4
    print('Cycle number', cycle_number, 'has', np.count_nonzero(high_condition), 'events')
    #print(cycle_frame[high_condition])
    last_dt = None
    consecutive_events = []
    for dt, row in cycle_frame[high_condition].iterrows():
        if last_dt != (dt - timedelta(hours=1)):
            # Add new group of consecutive values
            consecutive_events.append([])

        consecutive_events[len(consecutive_events)-1].append(row)
        last_dt = dt
    print('Consecutive events:', len(consecutive_events))

    for i, index in enumerate(['AE', 'AL', 'AU']):
        datapoints = []
        for consecutive_event in consecutive_events:
            ne = np.mean([row['newell'] for row in consecutive_event])
            y = np.mean([row[index] for row in consecutive_event])
            datapoints.append([ne, y])
        events_scatter = pd.DataFrame(datapoints)
        events_scatter.columns = ['newell', index+cycle_number]
        #print(events_scatter.head())
        events_scatter.plot(style='.', x='newell', y=index+cycle_number, grid=True, ax=axes[i+3], logx=True, color=cycle_colors[cycle_number])


for ax in [ax1,ax2, ax4, ax5]:
    #lines, labels = ax.get_legend_handles_labels()
    #ax.legend(lines, ['#22', '#23', '#24'], loc='best', numpoints=1)
    ax.legend().remove()
```

```
legend = []
for ax in [ax3, ax6]:
    lines, labels = ax.get_legend_handles_labels()
    legend.append(ax.legend(lines, ['#21', '#22', '#23', '#24'], numpoints=1, loc='center left', bbox_to_anchor=(1.12, 0.5)))


for ax in [ax1, ax2, ax3]:
    ax.set_xlabel('$\epsilon\'\/\mathrm{[W/km^2]}$')
for ax in [ax4, ax5, ax6]:
    ax.set_xlabel('$d\Phi_{MP}/dt$')

for ax in [ax1, ax4]:
    ax.set_ylabel('AE (nT)')
    ax.set_title('AE (nT)')
for ax in [ax2, ax5]:
    ax.set_ylabel('AL (nT)')
    ax.set_title('AL (nT)')
for ax in [ax3, ax6]:
    ax.set_ylabel('AU (nT)')
    ax.set_title('AU (nT)')

plt.tight_layout()
for ax in [ax1,ax2,ax3,ax4,ax5,ax6]:
    #xticks = ax.xaxis.get_major_ticks()
    #for i in range(1, len(xticks), 2):
    #    xticks[i].label1.set_visible(False)
    ax.set_xlim([4*10**2, 10**4])
for ax in [ax4,ax5,ax6]:
    ax.set_xlim([1*10**4, 10**5])

ax4.set_ylim([0, 4000])
plt.tight_layout()

fig.savefig('Large_Post-maximum-high-epsilon-scatter.eps', bbox_extra_artists=(legend[0],legend[1],), bbox_inches='tight')
#--------1---------2---------3---------4---------5---------6---------7
# Appendix-B3
# time series plot of AE for fixed epsilon
#                   = sunspot-kp/angeo/figure2.py (v2015)
#--------1---------2---------3---------4---------5---------6---------7
import pandas as pd
import numpy as np
colspecs = [(0, 4), # Year 1995 ... 2006
            (5, 8), # Day 1 ... 365 or 366
            (9, 11),# Hour 0 ... 23
            (12, 14), # Minute 0 ... 59 at start of average
            (92, 99), # By, nT (GSM)
            (100, 107), # Bz, nT (GSM)
            (124, 131), # Flow speed km/s
            (246, 251), # AE-index nT
            (252, 257), # AL-index nT
            (258, 263)  # AU-index NT
           ]
names = ['Year', 'Day', 'Hour', 'Minute', 'By', 'Bz', 'v', 'AE', 'AL', 'AU']
frame = pd.read_fwf('spdf.gsfc.nasa.gov/pub/data/omni/high_res_omni/omni_5min.asc', colspecs=colspecs, names=names)
```

```python

frame['Bz'].replace(to_replace=9999.99, value=np.nan, inplace=True)
frame['By'].replace(to_replace=9999.99, value=np.nan, inplace=True)
frame['v'].replace(to_replace=99999.9, value=np.nan, inplace=True)
frame['AE'].replace(to_replace=99999, value=np.nan, inplace=True)
frame['AL'].replace(to_replace=99999, value=np.nan, inplace=True)
frame['AU'].replace(to_replace=99999, value=np.nan, inplace=True)

frame['Datetime'] = frame['Year'] * 10000000 + frame['Day'] * 10000 + frame['Hour'] * 100 + frame['Minute']
frame['Datetime'] = pd.to_datetime(frame['Datetime'], format='%Y%j%H%M')
frame.set_index('Datetime', inplace=True)

# Calculate epsilon and newell using alternate form
original_form = False

B_T = np.sqrt(frame['By'] ** 2 + frame['Bz'] ** 2)

if original_form:
    theta = np.arctan2(frame['By'], frame['Bz'])
    frame['akasofu'] = frame['v'] * B_T ** 2 * np.sin(theta/2) ** 4
    frame['newell'] = frame['v'] ** (4/3) * B_T ** (2/3) * np.sin(theta/2) ** (8/3)
else:
    frame['akasofu'] = frame['v'] * ((B_T - frame['Bz']) ** 2) / 4
    frame['newell'] = ( frame['v'] ** 2 * ((B_T - frame['Bz']) ** 2 / (4*B_T)) ) ** (2/3)

# frame[['akasofu', 'newell']].describe().applymap(lambda x: '{:.2G}'.format(x))
# akasofu           newell
# count 2.3E+06     2.3E+06
# mean  4.9E+03     3.9E+03
# std   2E+04       4.2E+03
# min   0           0
# 25%   1.7E+02     8.6E+02
# 50%   1.2E+03     2.7E+03
# 75%   4.3E+03     5.5E+03
# max   1.9E+06     1.1E+05

# Check data coverage date
# frame['akasofu'][~np.isnan(frame['akasofu'])].tail(1)
# 2015-07-30 23:50:00    16890.425449

# Group |AL| per 3 months and use 5-min delay
delay = 1
akasofu = frame['akasofu'].values[:-delay]
newell = frame['newell'].values[:-delay]
AE = np.abs(frame['AE'].values[delay:])
AL = np.abs(frame['AL'].values[delay:])
AU = np.abs(frame['AU'].values[delay:])

d = pd.concat([pd.Series(frame.index.values[delay:]),
               pd.Series(akasofu), pd.Series(newell),
               pd.Series(AE), pd.Series(AL), pd.Series(AU)], axis=1)

d.columns = ['Datetime', 'akasofu', 'newell', 'AE', 'AL', 'AU']
months = ['02', '05', '08', '11'] # Feb, May, Aug, Nov
```

```python
d['Datetime group'] = d['Datetime'].apply(lambda x: str(x.year) + '-' + months[int(np.floor((x.month - 1) / 3))])
d.set_index('Datetime', inplace=True)

d['Datetime group'] = pd.to_datetime(d['Datetime group'], format='%Y-%m')

# Turn data into bands (note that units are not yet converted to W/km2)
bands = [[300, 1000],
         [1000, 3000],
         [3000, 10000],
         [10000, 30000],
         [100000, 300000]]
AL_bands = []
for band in bands:
    condition1 = d['akasofu'] >= band[0]
    condition2 = d['akasofu'] < band[1]
    AL_bands.append(d[condition1 & condition2][['Datetime group', 'AL']])


# Plotting code (and add Sunspot number plot)
%matplotlib inline
import matplotlib.pyplot as plt
import matplotlib.dates as dates
import matplotlib.gridspec as gridspec

colors = ["#D94E70",
"#7DC14B",
"#7887CE",
"#D4752C",
"#CA5DCA"]


fig = plt.figure(figsize=(8, 6))
gs = gridspec.GridSpec(2, 1, height_ratios=[4,1])

ax1 = plt.subplot(gs[0])
ax2 = plt.subplot(gs[1], sharex=ax1)

for i, AL_band in enumerate(AL_bands):
    result = AL_band.groupby('Datetime group').mean()
    #result = pd.rolling_mean(result, 4, center=True)
    result.plot(ax=ax1, color=colors[i], grid=True, rot=0)

ax1.set_ylabel('|AL| (nT)')

lines, labels = ax1.get_legend_handles_labels()
legend = ax1.legend(lines[::-1],
          [str('{:.1E}'.format(band[0]/100)) + ' - ' + str('{:.1E}'.format(band[1]/100)) for band in bands[::-1]],
          loc='center left', bbox_to_anchor=(1, 0.5), title='$\epsilon\'\/\mathrm{[W/km^2]}$', fontsize=11)
plt.setp(legend.get_title(),fontsize=16)
for legobj in legend.legendHandles:
    legobj.set_linewidth(2.0)

box = ax1.get_position()
ax1.set_position([box.x0, box.y0, box.width * 0.8, box.height])
```

```
ax1.set_title('3-month average |AL|')

from datetime import datetime, timedelta
def convert_partial_year(number):
    year = int(number)
    d = timedelta(days=(number - year)*365)
    day_one = datetime(year,1,1)
    date = d + day_one
    return date
SN = pd.read_csv('SN_m_tot_V2.0.csv', sep=';', header=None)
SN['Datetime'] = SN[2].apply(convert_partial_year)

SN.set_index('Datetime', inplace=True)
SN[3] = pd.rolling_mean(SN[3], 5, center=True)
ax = SN.plot(ax=ax2, y=3, legend=None, color='0.8', zorder=1, rot=0, grid=True)
ax.fill_between(SN.index, 0, SN[3], color='0.8')
ax.set_xlim(pd.to_datetime([1981, 2016], format='%Y'))
ax.set_ylim([0, 250])

for ax in [ax1, ax2]:
    ax.set_xlabel('Year')
    ax.xaxis.set_minor_locator(dates.YearLocator())

    for label in ax.get_xticklabels():
        label.set_horizontalalignment('center')

ax1.set_xlabel('')
fig.subplots_adjust(hspace=0.115)
ax2.set_ylabel('SSN')
plt.yticks(np.arange(0, 250, 100))

fig.savefig('HRO-Time-Series-AL-paper.eps', bbox_extra_artists=(legend,), bbox_inches='tight')
#--------1---------2---------3---------4---------5---------6---------7
# Appendix-B4
# time series plot of Sun-Spot Number(RI) with large events marked
#                   = sunspot-kp/angeo/figure5.py (v2015)
#--------1---------2---------3---------4---------5---------6---------7
import pandas as pd
import numpy as np
colspecs = [(0, 4), # Year 1995 ... 2006
            (5, 8), # Day 1 ... 365 or 366
            (9, 11),# Hour 0 ... 23
            (12, 14), # Minute 0 ... 59 at start of average
            (60, 67), # Field magnitude average, nT
            (76, 83), # By, nT (GSE)
            (84, 91), # Bz, nT (GSE)
            (124, 131), # Flow speed km/s
            (246, 251), # AE-index nT
            (252, 257), # AL-index nT
            (258, 263)  # AU-index NT
           ]
names = ['Year', 'Day', 'Hour', 'Minute', 'B', 'By', 'Bz', 'v', 'AE', 'AL', 'AU']
frame = pd.read_fwf('spdf.gsfc.nasa.gov/pub/data/omni/high_res_omni/omni_5min.asc', colspecs=colspecs, names=names)
```

```
frame['Bz'].replace(to_replace=9999.99, value=np.nan, inplace=True)
frame['By'].replace(to_replace=9999.99, value=np.nan, inplace=True)
frame['B'].replace(to_replace=9999.99, value=np.nan, inplace=True)
frame['AE'].replace(to_replace=99999, value=np.nan, inplace=True)
frame['AL'].replace(to_replace=99999, value=np.nan, inplace=True)
frame['AU'].replace(to_replace=99999, value=np.nan, inplace=True)
frame['v'].replace(to_replace=99999.9, value=np.nan, inplace=True)

# Don't use minutes, we are interested in hourly averages only
frame['Datetime'] = frame['Year'] * 10000000 + frame['Day'] * 10000 + frame['Hour'] * 100

theta = np.arctan2(frame['By'], frame['Bz'])
B_T = np.sqrt(frame['By'] ** 2 + frame['Bz'] ** 2)
frame['akasofu'] = frame['v'] * B_T ** 2 * np.sin(theta/2) ** 4
frame['newell'] = frame['v'] ** (4/3) * B_T ** (2/3) * np.sin(theta/2) ** (8/3)

# Calculate hourly averages
frame = frame.groupby('Datetime').mean()

# Turn integer index of hourly averages into datetimes
frame = frame.reset_index()
frame['Datetime'] = pd.to_datetime(frame['Datetime'], format='%Y%j%H%M')
frame.set_index('Datetime', inplace=True)

print(frame['akasofu'].describe())
# count    211359.000000
# mean       1103.194195
# std        3069.888314
# min           0.000000
# 25%         110.016861
# 50%         403.112641
# 75%        1113.352047
# max      226225.913370
# Name: akasofu, dtype: float64

# Plotting logic:
#for i, lower_limit in enumerate([
#        1*10 ** 4,
#        2*10 ** 4,
#        3*10 ** 4,
#        4*10 ** 4,
#        5*10 ** 4,
#        6*10 ** 4,
#        7*10 ** 4,
#        8*10 ** 4,
#        9*10 ** 4,
#        1*10 ** 5,
#        2*10 ** 5,
#        3*10 ** 5]):
#    condition = frame['akasofu'] >= lower_limit
#    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

www.irf.se