Appendix A Kgs. Lyngby, February 15, 2013

Part of: **Supplying relevant supporting material for endorsement of the Polar Cap index by the International Association of Geomagnetism and Aeronomy.**

Note 1:
All computer programs used to determine the PC index have been provided by Dr. Alexander Janzhura from AARI. They have been thoroughly tested in their final version when implementing them for the PCN index at DTU Space, by Dr. Alexander Janzhura, Dr. Jürgen Matzka and Dr. Claudia Stolle. This document describes the programs as used at DTU Space for calculation of the definitive PCN, but they can be used just as well for the PCS index.

Note 2:
By the implementation of these programs at DTU Space, we were in particular able to demonstrate that the index is independently reproducible.

Note 3:
The WDC for Geomagnetism, Copenhagen, will make the program sourcecodes and this appendix available in a permanent and citable form.

Note on calculation 1 (observatory data):
Input data is minute means for the years 1997 to 2009 from the geomagnetic observatory Qaanaaq (THL) received from the WDC for Geomagnetism, Edinburgh, in IAGA 2002 files with geomagnetic vector data in components along geographic X, Y and Z.

Note on calculation 2 (sector structure):
The sector structure is determined for each minute by a two step smoothing process from the THL X and Y component daily median values, respectively. In the first step, a 7 day running mean is produced, which in the next day is again smoothed by a 7 day 'Robust Loess' (quadratic fit).

Note on calculation 3 (actual quiet day curve, actual QDC):
The actual QDC is calculated from THL X and THL Y minus the respective sector structure signal. It is calculated for a defined period (here 30 days) and the actual QDC is calculated for one day (UT) within the period (the day for which the actual QDC is calculated depends on when the most quiet conditions were met). For the definitive PCN index a series of actual QDCs are calculated by shifting the 30-day period by 10 days at a time.

Note on calculation 4 (merging electric field after Kan and Lee (1979), EKL)
From NASA's http://cdaweb.gsfc.nasa.gov/istp_public/, on February 7th 2013, we selected "OMNI database" and "OMNI_HRO_1MIN" and selected

- By (nT), GSM, determined from post-shift GSE components
- Bz (nT), GSM, determined from post-shift GSE components
- Flow speed (km/s), GSE

for 1997 to 2009 and saved this as ASCII data without header with name OMNI_YYYY.txt in the directory Appendix_A_file_archive\COEFF_CALC\OMNI_DATA (now zipped).
These files were then converted to Matlab data files OMNI_YYYY.mat.
From these, EKL was calculated (in mV/m) and stored in Ekl_YYYY.mat.
EKL is then averaged over 5 minutes and shifted by 15 minutes and stored in ekls_YYYY.mat.

Note on calculation 5 (geomagnetic disturbances in observatory data):
Subtract sector structure and QDC from THL X and THL Y and make 5 minute averages, save in dist_YYYY.m in directory Appendix_A_file_archive\COEFF_CALC\STEP1_getdist\.

Note on calculation 6 (angle phi):
Copy all files dist_YYYY.mat and ekls_YYYY.m to directory
Appendix_A_file_archive\COEFF_CALC\STEP2_fi\.
For each year, the optimal angle phi (for which the projection of geomagnetic disturbances correlates best with EKL shifted by 15 minutes, found by varying in steps of 5 degrees) is calculated and saved as matrix in F_YYYY.mat. It is stacked for all years and smoothed and saved as matrix in Fi_2d.mat.
Finally, phi is saved as a time series for every 5 minutes for a full calendar year in Fi_year.mat. The corresponding projections of the geomagnetic disturbances are calculated and saved in Hproj_YYYY.mat. File yeartime.mat is produced.

Note on calculation 7 (coefficients a and b):
Copy all files Hproj_YYYY.mat and ekls_YYYY.m and yeartime.mat to directory
Appendix_A_file_archive\COEFF_CALC\ STEP3_ab\.
For each year, the coefficients a and b for the optimal angle phi are calculated and saved as matrix in ab_YYYY.mat. They are is stacked for all years and smoothed and saved as matrix in ab_2d.mat.
Finally, a and b saved as a time series for every 5 minutes for a full calendar year in ab_year.mat.
Phi, a and b are interpolated to 1-minute resolution and combined in one file: coeff.mat.

Note on calcualtion 8 (calculation of PCN index):
Copy coeff. mat into directory Appendix_A_file_archive\PC_CALC\.

Technical note 1:
We have implemented the process by storing the data from THL in a mySQL database Version 5.1.67 under Linux.
Matlab programs and functions that have been used are running on Matlab Version 7.12.0.635 (R2011a) 64-bit (glnxa64) under Linux.

Technical note 2:
In the following pages, for each program, function, or file we include a table with two or more rows: the first row contains the name of the file; the second row shows the content of the file. Further rows give detailed explanation about certain function included in Matlab.
Each function is only explained once, at its first appearance.

Make QL-database 'magdat' with the table 'pcnthl':

pcnthl

Column	Type	Null	Default	Comments
id	int(11)	No		id
timestamp	timestamp	No	CURRENT_TIMESTAMP	time stamp
time	datetime	No		time of the measurement
x	mediumint(9)	Yes	*NULL*	X component
y	mediumint(9)	Yes	*NULL*	Y component
z	mediumint(9)	Yes	*NULL*	Z component
ss_x	float	Yes	*NULL*	SS(x)
ss_y	float	Yes	*NULL*	SS(y)
qdc_x	float	Yes	*NULL*	QDC(x)
qdc_y	float	Yes	*NULL*	QDC(y)
pc	float	Yes	*NULL*	PCN THL

Indexes:

Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null	Comment
PRIMARY	BTREE	Yes	No	time	19992788	A		
id	BTREE	Yes	No	id	19992788	A		
tamestamp	BTREE	No	No	timestamp	51263	A		

Make config-file for database access:

Appendix_A_file_archive\PC_DB_init\db_config.pc

```
127.0.01:3306
Login-name
Login-password
magdat
pcnthl
```

Script to start up database initialization for all year (here: example for 1997):

years2db.m

```
date2db (1997)
```

Function to initialize database (fill in NULL-values):

Appendix_A_file_archive\PC_DB_init\date2db.m

```
function date2db (yr);

% read config file db_config.pc

f = fopen('db_config.pc','r');
    serv = fgetl(f);
    log = fgetl(f);
    pass = fgetl(f);
    dbname  = fgetl(f);
    table = fgetl(f);

fclose(f);

% open DB

 mysql('open', serv,log,pass);
 mysql(['use ' dbname]);



    start_time =  datenum(yr,1,1,0,0,0)
      year_days = sum(eomday(yr, 1:12));




    for i= start_time:1/1440:start_time+year_days-1/1440

        tm = datestr(i);
        time = ['"' datestr(tm,'yyyy-mm-dd HH:MM:SS') '"'];
        disp(time)
        mysql(['INSERT INTO ' table ' (time, x, y, z) VALUES (' time ', NULL ,
NULL , NULL) ON DUPLICATE KEY UPDATE  x=NULL, y=NULL, z=NULL ;']);



    end

 mysql('close');
```

Script to start up archive data transfer to database:

Appendix_A_file_archive\PC_DB_init\ datayears2db.m

```
%% IAGA Data to DB
%  The procedure uploads data
%  from IAGA 2002 files to database
%  for some years
%%
station = 'thl';
for y = 1997:2009
 year = num2str(y);
 pash = ['/r14/magobs/PCN/data/THL_archive/' year '/']
 iaga2db(station,y,pash)
end
```

Function to transfer archive data to database, works for IAGA 2002 with X Y Z data:

Appendix_A_file_archive\PC_DB_init\iaga2db.m

```
function iaga2db (sta_name,year,dir)
% put the data from IAGA2002 files to the DataBase
% input params:
% sta_name - three letters IAGA name of the station ('vos')
% year - year of the IAGA files (2012)
% dir - path to the IAGA files
%-------------------------------------------------------------

% read config file db_config.pc

f = fopen('db_config.pc','r');
    serv = fgetl(f);
    log = fgetl(f);
    pass = fgetl(f);
    dbname  = fgetl(f);
    table = fgetl(f);

fclose(f);

% open DB

 mysql('open', serv,log,pass);
 mysql(['use ' dbname]);

% read every file in the year
start_date = datenum(year,1,1);
%days = yeardays(year);
days = sum(eomday(year,1:12));

    for actual = start_date:1:start_date+days-1;
        %thl20111230dmin.min
        iaga_name = [dir, sta_name, datestr(actual,'yyyymmdd'), 'dmin.min'];
        disp(iaga_name);
        fid = fopen(iaga_name);
        if fid > -1
          while ~feof(fid)
              tline = fgetl(fid);
              %2011-12-31 23:59:00.000 365      2510.90   -3229.90   56246.10
```

```
88888.80
              if num2str(year) == tline(1:4)

                  time = [''''  tline(1:19)  ''''];
                  disp(time);
                  f_x = str2num(tline(31:40));
                  f_y = str2num(tline(41:50));
                  f_z = str2num(tline(51:60));
                %
                %  HERE CONVERT THE XYZ TO HEZ coordinate system
                %
                    if (f_x < 88888)
                        xs=[''''  int2str(round(f_x))  ''''];
                    else
                        xs = 'NULL';
                    end
                    if (f_y < 88888)
                        ys=[''''  int2str(round(f_y))  ''''];
                    else
                        ys = 'NULL';
                    end
                    if (f_z < 88888)
                        zs=[''''  int2str(round(f_z))  ''''];
                    else
                        zs = 'NULL';
                    end

                   % add data to the DB

                  query = ['INSERT INTO ',table ,' (time,x,y,z) VALUES
('',time,'','',xs,'','',ys,'','',zs,'') ON DUPLICATE KEY UPDATE
x='',xs,'',y='',ys,'',z='',zs,'';'];


                  mysql(query);
                   %disp (query);
              end
          end

         fclose(fid);
        end


    end

mysql('close');
```

Script to start up sector structure interpolation for archive data:

Appendix_A_file_archive\PC_DB_init\year_ss.m

```
ss_bd_interp([1997 1 1],[1997 12 31]);
ss_bd_interp([1998 1 1],[1998 12 31]);
ss_bd_interp([1999 1 1],[1999 12 31]);
ss_bd_interp([2000 1 1],[2000 12 31]);
ss_bd_interp([2001 1 1],[2001 12 31]);
ss_bd_interp([2002 1 1],[2002 12 31]);
ss_bd_interp([2003 1 1],[2003 12 31]);
ss_bd_interp([2004 1 1],[2004 12 31]);
ss_bd_interp([2005 1 1],[2005 12 31]);
ss_bd_interp([2006 1 1],[2006 12 31]);
ss_bd_interp([2007 1 1],[2007 12 31]);
ss_bd_interp([2008 1 1],[2008 12 31]);
ss_bd_interp([2009 1 1],[2009 12 31]);
```

Function to call function to retrieve data from database and to call function to calculate sector structure and to call function to upload sector structure into database

Appendix_A_file_archive\PC_DB_init\ss_bd_interp.m

```
function ss_bd_interp (start_dt,end_dt)

%-----------------------------------------------------------------------
% The function interpolates SS between start date (start_dt)
% end end date (end_dt) - date time vectors (YYYY,MM,DD)
% and write the results to 'ss_x', 'ss_y'
% // function takes the data from pos = 'x','y'
%-----------------------------------------------------------------------

% check time does not exceed 1 year period
n= datenum([end_dt,00,00,00])-datenum([start_dt,00,00,00]);
if n>366
   disp ('length more then 1 year !!!');
   return;
end  %if

% Read DB
    var_x = dbget ( start_dt,[end_dt,23,59,0],'x');
    var_y = dbget ( start_dt,[end_dt,23,59,0],'y');
    % SS interpolation
     ss_x = ss_interp (var_x);
     ss_y = ss_interp (var_y);


% Write DB
    dbset ( start_dt,[end_dt,23,59,0],'ss_x',ss_x);
    dbset ( start_dt,[end_dt,23,59,0],'ss_y',ss_y);
```

Function to retrieve data from database

Appendix_A_file_archive\PC_DB_init\dbget.m

```
function out = dbget ( start_dt,end_dt, pos)
% get data from "pos" collumn of the database

% read config file db_cong.pc

f = fopen('db_config.pc','r');
    serv = fgetl(f);
    log = fgetl(f);
    pass = fgetl(f);
    dbname  = fgetl(f);
    table = fgetl(f);

fclose(f);

% make db formated time intervals
start_time =  datenum(start_dt);
end_time =  datenum(end_dt);
    s_time = ['"' datestr(start_time,'yyyy-mm-dd HH:MM:SS') '"'];
    e_time = ['"' datestr(end_time,'yyyy-mm-dd HH:MM:SS') '"'];

%get data from db
mysql('open', serv,log,pass);
  mysql(['use ' dbname]);
    out = mysql(['select ' pos ' from ' table ' where time >= ' s_time ' and
time <= ' e_time ';']);
  mysql('close');
```

Function to calculate sector structure and explanation of Matlab function 'smooth'

Appendix_A_file_archive\PC_DB_init\ ss_interp.m

```
function out= ss_interp (var)

% var - 1-minute variation array of the magmetic field
% out - smoothed period

day_points =1440; % 1-minute data
%day_points =288; % 5-minute data

% days
days = length(var)/day_points;

% find daily median value
 for i=1:days
     day_med(i) = nanmedian(var((i-1)*day_points+1:(i)*day_points));
 end
% smooth medians

     w = smooth(day_med,7,'moving'); %7days
     w = smooth(w,7,'rloess');

% interpolation to day point
```

```
out =interp1([1:days],w,[1:1/day_points:days+1],'spline','extrap');
out(length(out)-1)=[];
```

```
SMOOTH  Smooth data.
    Z = SMOOTH(Y,SPAN) smooths data Y using SPAN as the number of points used
    to compute each element of Z.

    Z = SMOOTH(Y,SPAN,METHOD) smooths data Y with specified METHOD. The
    available methods are:

            'moving'   - Moving average (default)
            'lowess'   - Lowess (linear fit)
            'loess'    - Loess (quadratic fit)
            'sgolay'   - Savitzky-Golay
            'rlowess'  - Robust Lowess (linear fit)
            'rloess'   - Robust Loess (quadratic fit)
```

```
INTERP1 1-D interpolation (table lookup)
    YI = INTERP1(X,Y,XI) interpolates to find YI, the values of the
    underlying function Y at the points in the array XI. X must be a
    vector of length N.
    If Y is a vector, then it must also have length N, and YI is the
    same size as XI.  If Y is an array of size [N,D1,D2,...,Dk], then
    the interpolation is performed for each D1-by-D2-by-...-Dk value
    in Y(i,:,:,...,:).
    If XI is a vector of length M, then YI has size [M,D1,D2,...,Dk].
    If XI is an array of size [M1,M2,...,Mj], then YI is of size
    [M1,M2,...,Mj,D1,D2,...,Dk].

    YI = INTERP1(Y,XI) assumes X = 1:N, where N is LENGTH(Y)
    for vector Y or SIZE(Y,1) for array Y.

    Interpolation is the same operation as "table lookup".  Described in
    "table lookup" terms, the "table" is [X,Y] and INTERP1 "looks-up"
    the elements of XI in X, and, based upon their location, returns
    values YI interpolated within the elements of Y.

    YI = INTERP1(X,Y,XI,METHOD) specifies alternate methods.
    The default is linear interpolation. Use an empty matrix [] to specify
    the default. Available methods are:

      'nearest'  - nearest neighbor interpolation
      'linear'   - linear interpolation
      'spline'   - piecewise cubic spline interpolation (SPLINE)
      'pchip'    - shape-preserving piecewise cubic interpolation
      'cubic'    - same as 'pchip'
      'v5cubic'  - the cubic interpolation from MATLAB 5, which does not
                   extrapolate and uses 'spline' if X is not equally
                   spaced.
```

Function to upload sector structure into database

Appendix_A_file_archive\PC_DB_init\ dbset.m

```
function dbset ( start_dt,end_dt, pos, varr);
% send data to "pos" column of the database

% read config file db_cong.pc

f = fopen('db_config.pc','r');
    serv = fgetl(f);
    log = fgetl(f);
    pass = fgetl(f);
    dbname  = fgetl(f);
    table = fgetl(f);

fclose(f);

% make db formated time intervals
start_time =  datenum(start_dt);
end_time =  datenum(end_dt);
    s_time = ['"' datestr(start_time,'yyyy-mm-dd HH:MM:SS') '"'];
    e_time = ['"' datestr(end_time,'yyyy-mm-dd HH:MM:SS') '"'];
    (end_time-start_time)*1440;
%set data to db
mysql('open', serv,log,pass);
  mysql(['use ' dbname]);

     k=1;

     for i= start_time:1/1440:end_time
        vr = varr(k);
        k=k+1;
            str = 'NULL';
            if ~isnan(vr);  str=['"' num2str(round(vr)) '"']; end;
        time = ['"' datestr(i,'yyyy-mm-dd HH:MM:SS') '"'];
        disp (time);
        sqlstr=['update ' table ' set ' pos ' = ' str ' where time = ' time
';'];
        mysql(sqlstr);

     end

  mysql('close');
```

Function to startup calculation of actual QDC:

Appendix_A_file_archive\PC_DB_init\ year_actual_qdc.m

```
%% Construct Actual QDC for one year
%
%%

function year_actual_qdc(year)

% end date of period for calculation actual QD:
ds = datenum([year 1 1 0 0 0]);
```

```
for dt=ds:10:ds+365
  [y, m, d] = datevec(dt);
  period = [y,m,d];
  disp(period);
  tic;
  qday_db(period);
  toc
end
```

Function to retrieve observatory data and sector structure data, call function to calculate actual QDC for a 30 day period, and store actual QDC on database:

Appendix_A_file_archive\PC_DB_init\qday_db.m

```
%% QDC Actual Day DB
%  Actual day QDC to DB
%%

function qday_db(start_dt);
% ------------------------------------------------------
% The function calculates QDC for 30-days dataset
% and upload QDC to "Actual Day" date in DB
% INPUT
% start_dt vector of start data [YYYY, MM, DD]
%
% ------------------------------------------------------

% filed end date of 30-days period
ds = datenum(start_dt);
[y, m, d] = datevec(ds+30);
end_dt = [y,m,d,23,59,00];

% Read DB
%   Read components
    var_x = dbget ( start_dt,end_dt,'x');
    var_y = dbget ( start_dt,end_dt,'y');
%   Read SS for the components
    ss_x = dbget ( start_dt,end_dt,'ss_x');
    ss_y = dbget ( start_dt,end_dt,'ss_y');
% Calculate COMP-SS
    by_x = var_x - ss_x;
    by_y = var_y - ss_y;

% Calculate QDC for every component

[qday_x, ActDay_x]= q_day(by_x);
[qday_y, ActDay_y]= q_day(by_y);

if ~isnan(ActDay_x)
    % Calculate date for the Actual Day;
    [y, m, d] = datevec(ds+ActDay_x);
    act_dt_x = [y, m, d];
     % Write QDC for Actual Day to DB
     dbset ( [act_dt_x,0,0,0],[act_dt_x,23,59,0],'qdc_x',qday_x);
end;
if ~isnan(ActDay_y)
    [y, m, d] = datevec(ds+ActDay_y);
    act_dt_y = [y, m, d];
```

```
     % Write QDC for Actual Day to DB
     dbset ( [act_dt_y,0,0,0],[act_dt_y,23,59,0],'qdc_y',qday_y);
end;
```

Function to calculate actual QDC for a 30 days period:

Appendix_A_file_archive\PC_DB_init\q_day.m

```
%% QDC calculation
% for actual day
%%
function [qday, ActDay]= q_day(arr);
% -------------------------------------------------------------
% The function calculates actual Quiet Day Curve (QDC) for
% INPUT array (arr) of data.
% arr - is 1-minutes values for 30 days = 43200 elements
% OUTPUT
% qday - array of 1400 elements (1-day) QDC values
% ActDay - actual day in 30 days array
% -------------------------------------------------------------

step=2;
arr_s=smooth(arr,120);
arr_g=gradient(arr);
len= size(arr,1);
q(1:len)=NaN;
a=len/1440;
day=[];
s=0;
p=0;
divq=0;
act(1:50000)=NaN;
ac=1;
flag = true;

while min(p)<120
 divq=divq + step;
 clear q  p;
 q(1:len)=NaN;


for i=31:len-31
   if max(abs(arr(i-30:i+30)-arr_s(i-30:i+30)))<divq & max(abs(arr_g(i-
30:i+30)))<divq
     q(i)=arr(i);
    act(ac)=i;
     ac=ac+1;
   end;
end
q=q';

for i=1:a
    n=i*1440;
     s=size(day,2)+1;
    day(1:1440,s)= q(n-1439:n);

end;

for i=1:1320
```

```
    p(i) = length(find(not(isnan( day(i:i+120,:)  ))));
end;

actual = step/divq;

 if actual<=0.05
  flag = false;
  break;
 end; %IF

end; %WHILE



if flag
    qqq=[day ;day; day];
    qqq_s  = smooth(nanmean(qqq'),240,'moving');%240
    qqq_s = smooth(qqq_s,240,'rlowess');
    qday = qqq_s(1440:2880-1);
    %!!!!! Actual Day !!!!!!!
    ActDay=round(nanmedian(act)/1440)

else
  qday(1:1440) = NaN;
  ActDay = NaN;
  actual = 0;
end;
```

Script to call functions for QDC interpolation (here example for 1997):

Appendix_A_file_archive\PC_DB_init\years_interp_qdc.m
`qdc_db_interp ([1997 1 1],[1997 12 31]);`

Function to call function to interpolate QDC from actual QDC

Appendix_A_file_archive\PC_DB_init\qdc_db_interp.m

```
%% QDC interpolation for period of time
% from start date to end date
%%
function qdc_db_interp (start_dt,end_dt)

%-----------------------------------------------------------------------
% The function interpolates QDC between start date (start_dt)
% and end date (end_dt) - date time vectors [YYYY,MM,DD]
% and write the results to 'qdc_e', 'qdc_h' columns of the DB
% // function takes the data from pos = 'e','h','ss_e','ss_h'
%-----------------------------------------------------------------------

% check time does not exceed 1 year period
d_start = datenum([start_dt,00,00,00]);
d_end = datenum([end_dt,00,00,00]);
 n = d_end - d_start;
    if n>366
        disp ('length more then 1 year !!!');
        return;
    end  %if

%=========
% STEP 1 =
%=========
% Calculate QDC for Actual Days for certain period
% 5-days step
% E end H components are includet to the function qday_db
% for dn = d_start:10:d_end-30 % -30 days because func qday_db takes day day+30
array
%    disp(datestr(dn,'dd-mm-yyyy'));
%    [y, m, d] = datevec(dn);
%    qday_db([y, m, d]);
%
% end % for dn
%=========
% STEP 2 =
%=========
% Interpolate every day QDC for certain period

% Read qdc_e and qdc_h from the DB
    qdc_x = dbget ( [start_dt,00,00,00],[end_dt,23,59,0],'qdc_x');
    qdc_y = dbget ( [start_dt,00,00,00],[end_dt,23,59,0],'qdc_y');

    % QDC interpolation for every day
     sqdc_x = qdc_interp (qdc_x);
     sqdc_y = qdc_interp (qdc_y);

 % Write every day QDC to DB
    dbset ( start_dt,[end_dt,23,59,0],'qdc_x',sqdc_x);
    dbset ( start_dt,[end_dt,23,59,0],'qdc_y',sqdc_y);
```

Function to interpolation/extrapolation of QDC for all days from actual QDC:

Appendix_A_file_archive\PC_DB_init\ qdc_interp.m

```
%% QDC Interpolation for every day
%
%%
function out_arr = qdc_interp(in_arr);
% ------------------------------------------------
%  The function interpolates QDC for every day from
%  Actual Day array
%  in_arr - 1D array of Actual QDC
%  out_arr - 1D array of Every Day QDC
% ------------------------------------------------
days = floor(length(in_arr)/1440);
% =======
% STEP 1
% construct 2D array of Actual QDC
j=0;
for i=0:days-1
  arr_day = in_arr(1+i*1440:1440+i*1440);
  if ~isnan(arr_day)
    j=j+1;
    qDay(1:1440,j) = arr_day;
    da(j)=i+1;
  end % if ~isnan
end % for i

% =======
% STEP 2
% interpolation and smoothing QDC for every day
 for i=1:1440
     interp_arr(i,:) =
smooth(interp1(da,qDay(i,:),[1:days],'nearest','extrap'),60,'lowess');

 end;

% =======
% STEP 3
% Make 1D attay of the QDC
 for i=1:size(interp_arr,2)
    l_arr((i-1)*1440+1 : i*1440) = interp_arr(1:1440,i);
 end
out_arr = smooth(l_arr,120,'loess');
```

Script to start up EKL calculation from OMNI data and save in Matlab data files (only for 1998 shown here)

Appendix_A_file_archive\COEFF_CALC\OMNI_DATA\make_ekl-m

```
clear;

load('OMNI_1997.mat');
Ekl_1997 = omni2ekl(OMNI_1997);
save('Ekl_1997.mat','Ekl_1997');
clear;
```

Function to calculate merging electric field EKL:

Appendix_A_file_archive\COEFF_CALC\OMNI_DATA\omni2ekl.m

```
%% Convert OMNI data to E_kl
%
%%

function E_kl = omni2ekl(in_arr)

% --------------------------------------------------------------------
% converts OMNI data to Ekl
% input ASCII strings array contain
% Date, Time, By, Bz, Vsw
%
% --------------------------------------------------------------------
%%




for i = 1:length(in_arr)
      st = in_arr(i)
     dt(i) = datenum(st{1}(1:19), 'dd-mm-yyyy HH:MM:SS');
     by(i) = str2num(st{1}(25:37));
     bz(i) = str2num(st{1}(39:51));
     vsw(i)= str2num(st{1}(55:67));

     by(find(by>100))=NaN;
     bz(find(bz>100))=NaN;
     vsw(find(vsw>5000))=NaN;
    % Kan-Lee equation
     kor=sqrt(by.^2.+bz.^2);
     E_kl = vsw.*kor.*(sin(acos(bz./kor)./2).^2)./1000;
end
```

Script to call function for average EKL, then stores 15 minute time-shifted data (only 1997 shown):

Appendix_A_file_archive\COEFF_CALC\OMNI_DATA\make_ekl_5min.m

```
%% Procedure makes 5-min averaged file and shift 15 minutes forward (back in
time)
%
% The 15-minute forward shift is used for
% PC coefficient derivation
% as the ground magnetic field reacts 15-20 minute late
% after influence of IMF
%%
clear;
% ============================================
disp('1997');
load('Ekl_1997.mat')
ekls = avr5min(Ekl_1997);
last_1997 = ekls(length(ekls)-2:length(ekls)); % last 15 minutes
```

```
ekls_1997 = [NaN NaN NaN ekls(1: length(ekls)-3)];
save('ekls_1997.mat','ekls_1997');
clear Ekl* ekl*;
```

Function to average EKL from 1 minute values to 5 minute values:

Appendix_A_file_archive\COEFF_CALC\OMNI_DATA\avr5min.m

```
%% The function avarages 1-minute data to 5 minute
%
%%

function out5 = avr5min( in1 )

% ====================================================
% The function avarages 1-minute year array
% to 5-minute year array
% INPUT
% in1 - 1 demention array
% out5 - 5 minut avaraged array
% ====================================================

len = floor(length(in1)/5);
for i=0:len-1
  out5(1,i+1) = nanmean(in1(i*5+1:i*5+5));

end %for
```

Script to call function for averaging disturbances in observatory data, saves data in file (here only 1997):

Appendix_A_file_archive\COEFF_CALC\STEP1_getdist\make_dist_5min.m

```
%% Procedure makes 5-min averaged X Y file
%
% The procedure creates 5-minute averaged array
% of X and Y disturbed variations
% for PC coefficient calculation
%%
% ============================================


%-------------------------------------
disp('1997');
dist_1997 = dist_5m(1997);
save('dist_1997.mat','dist_1997');
clear dist*;
```

Function to calcuate disturbances in observatory data and 5 minute averaging

Appendix_A_file_archive\COEFF_CALC\STEP1_getdist\dist_5m.m

```
%% 5-min averaged X Y disturbances from the DB
%
%%

function out = dist_5m(year)

% ================================================
% input - year (YYYY)
% output - out structure of X_dist Y_dist 5-min averaged
%
% ================================================
    start_dt = [year,1,1,00,00,00];
    end_dt = [year,12,31,23,59,00];

   % Read X and Y data from the DB
    x = dbget ( start_dt,end_dt,'x');
    y = dbget ( start_dt,end_dt,'y');
    ss_x = dbget ( start_dt,end_dt,'ss_x');
    ss_y = dbget ( start_dt,end_dt,'ss_y');
    qdc_x = dbget ( start_dt,end_dt,'qdc_x');
    qdc_y = dbget ( start_dt,end_dt,'qdc_y');

   % substract SS and QDC variations
    dist_xs = x - ss_x - qdc_x;
    dist_ys = y - ss_y - qdc_y;
    out.x = avr5min(dist_xs);
    out.y = avr5min(dist_ys);
```

Function manually called to constructs an array containing for every 5 minutes for a whole year the hour and the minute:

U:\PCN\Appendix_A_file_archive\COEFF_CALC\STEP2_fi\maketime.m

```
%% Constructs yeartime array
%
%%
function maketime;

     dtv = datevec(1:1/1440*5:366-1/1440);
     yeartime = dtv (:,4:5)';
     save('yeartime.mat','yeartime');
```

Function manually called to call functions below:

U:\PCN\Appendix_A_file_archive\COEFF_CALC\STEP2_fi\fi_step1.m

```
%% Calculate Fi for 1997 - 2009
%
%%
function fi_step1;



for year = 1997:2009
 disp(year);
    Hproj_arr = fi_arr(year);
    fi_corr(year,Hproj_arr);
end;
```

Function calls function to calculate projections (5 degree steps) of disturbance every 5 minutes (fi stands for angle phi)

Appendix_A_file_archive\COEFF_CALC\ STEP2_fi\fi_arr.m

```
%% Function makes array of H_proj
%  5-min interval in the year
%  direction 0 - 355 deg with 5 deg step
%%

function Hproj_arr = fi_arr(year);

  % load dist file
  load(['dist_' num2str(year) '.mat']);
  eval (['dist = dist_' num2str(year)]);

  %load time file
  load('yeartime.mat');



  % calculate fi array

    for fi=0:5:355
       Hproj_arr((fi+5)/5,1:105120) = hproj(yeartime(1,:),yeartime(2,:),fi,
dist.x(1,1:105120), dist.y(1,1:105120));
    end % for fi
```

Function to calculate projections of disturbance:

Appendix_A_file_archive\COEFF_CALC\STEP2_fi\hproj.m

```
%% Function calculates Hproj
%
%
%%


% ======================================================
```

```
%  hour and minutes - UT time
%
%
%
% ========================================================


function H_proj = hproj(hour,minute,fi, dist_x, dist_y);
% !!!!!!!!!!!
  lon =291; % THL
% lon =106.9; % VOS
% !!!!!!!!!!!


    UT = hour.*15 + minute.*0.25; % TIME to DEG
    y = lon + fi + UT;
    H_proj = dist_x.*sin(y.*pi./180)-dist_y.*cos(y.*pi./180);
```

Function to determine optimal correlation between projection of disturbance and EKL and saves optimal phi-angel in yearly files F_YYYY.m, calling for this function makerr.m:

Appendix_A_file_archive\COEFF_CALC\STEP2_fi\fi_corr.m

```
%% Find best correlation of Ekl and array of H_proj for every Fi
%  5-min interval in the year
%  direction 0 - 355 deg with 5 deg step
%%

function fi_corr(year,Hproj_arr);


% load  file

load(['ekls_' num2str(year) '.mat']);
eval (['ekl = ekls_' num2str(year)]);

%load time file
  load('yeartime.mat');

%
H_proj = Hproj_arr';
time_a = yeartime';
% Correlation

for month=1:12

    clear R;
    for hr=0:23
        for min =0:5:55;
```

```
        day = 16+30*(month-1);
        day_min = day-15;
        day_max = day+15;

        min5 = day*288;
        min5_min = day_min*288;
        min5_max = day_max*288;
        clear tab*;
        clear temp*;

        temp1 = H_proj(min5_min:min5_max,:);
        temp_date = time_a(min5_min:min5_max,:);
        temp_ace = ekl(1,min5_min:min5_max)';
        temp_pos = find(and(temp_date(:,1)==hr,temp_date(:,2)==min));
        tabcorr_Esw = temp_ace(temp_pos);
        tabcorr_Hproj = temp1(temp_pos,:);

        %clear temp*
        tabcorr_Hproj(find(isnan(tabcorr_Esw)),:)=[];
        tabcorr_Esw(find(isnan(tabcorr_Esw)))=[];
        tabcorr_Esw(find(isnan(tabcorr_Hproj(:,1))))=[];
        tabcorr_Hproj(find(isnan(tabcorr_Hproj(:,1))),:)=[];
        %temp_ace(temp_pos);
                for gr = 1:72
                    rr = corrcoef(tabcorr_Esw(:),tabcorr_Hproj(:,gr));
                    if length(rr)>1
                     R((hr.*12)+(min+5)/5,gr) = rr(1,2);
                    else
                     R((hr.*12)+(min+5)/5,gr) = NaN;
                    end
                end; %for gr=
        end; % for min=

    end; % for hr=
    F(month,1:288) = makerr(R).*5-180;
    disp (month);
end; % for month


% Save Fi to file F_YYYY.mat

eval (['F_' num2str(year) '=F;']);
fn = ['F_' num2str(year)];
eval (['save(''' fn ''', ''' fn ''');']);
```

Function (called fi_corr.m) to find optimum correlation:

Appendix_A_file_archive\COEFF_CALC\STEP2_fi\ makerr.m

```
function ff = makerr (R);
smooth_level =90; %!!!!!
for i=1:288
   p5=polyfit([1:72],R(i,:),5);
   R(i,:)=polyval(p5,[1:72]);
```

```
end;
 re= min(R');
 for i=1:288
  ff(i) = find(R(i,:)==re(i));
 end; %i
 ff= smooth([ff ff ff],smooth_level,'lowess');
 %plot(ff);
 ff = ff(289:288*2);
figure;
contourf(R,10);
hold on;
plot(ff,[1:288],'--r','linewidth',2.5);
```

Stacks for all years matrix with phi-angles for all months and all UT-times and smoothes this matrix such that hour 23 and hour 00 (and month 12 and month 1) have smooth boundary. Saves matrix in Fi_2d.mat and plots matrix. Calls function to calculate phi every 5 minutes for the year and saves that as Fi_year.mat.

Appendix_A_file_archive\COEFF_CALC\STEP2_fi\ fi_step2.m

```
%% Avarage Fi
%
%
%%

function fi_step2;

% load all fi files from 1997 to 2009
i=0;
    for year = 1997:2009
      i=i+1;
      name = ['F_' num2str(year)];
      eval (['load(''' name ''');']);
      eval (['fa(:,:,' num2str(i) ') = ' name ';']);
    end;
% avarage
   f_avr = nanmean(fa,3);


 F_e = [f_avr f_avr f_avr ; f_avr f_avr f_avr ; f_avr f_avr f_avr];
 F_ex(:,:,1)= F_e;
 F_ex(:,:,2)= F_e;
 F_ex(:,:,3)= F_e;

 F_s = smooth3(F_ex,'gaussian',3,0.8);%7
 F_n = smooth3(F_s,'box',3);%5
 Fi_2d = F_n(13:12*2,289:288*2,3);
 Fi_year = coeff_for_year(Fi_2d)';
 Fi_year=smooth(Fi_year,24,'lowess');

 save('Fi_2d.mat','Fi_2d');
```

```
 save('Fi_year.mat','Fi_year');

contourf(Fi_2d,12);
```

```
SMOOTH3  Smooth 3D data.
    W = SMOOTH3(V) smoothes input data V. The smoothed data is returned
        in W.

    W = SMOOTH3(V, METHOD) METHOD can be either of the filters 'gaussian'
        or 'box' (default) and determines the convolution kernel.

    W = SMOOTH3(V, METHOD, SIZE) sets the size of the convolution kernel
        (default is [3 3 3]). If SIZE is a scalar, the size is interpreted
        as [SIZE SIZE SIZE].

    W = SMOOTH3(V, METHOD, SIZE, ARG) sets an attribute of the
        convolution kernel. When METHOD is 'gaussian', ARG is the standard
        deviation (default is .65).
```

MAKE PLOT OF FI !!!

Function to calculate phi every 5 minutes for the year by interpolation:

Appendix_A_file_archive\COEFF_CALC\STEP2_fi\ coeff_for_year.m

```
%% Interpolate 1-D array from 2-D array for coefficients matrix;
%
%%

function fi_year = coeff_for_year(f_avr);



    days_in_year = 366;
    f_tmp =[f_avr f_avr f_avr;f_avr f_avr f_avr; f_avr f_avr f_avr];
    [X,Y] = meshgrid([15 :days_in_year/12:days_in_year*3-15],[1:288*3]);
    [XI,YI] = meshgrid([1:days_in_year*3],[1:288*3]);
    fi_tmp1 = interp2(X,Y,f_tmp',XI,YI,'spline');
    fi_avr = fi_tmp1([288+1:288*2],[days_in_year+1:days_in_year*2]);
    for i=1:days_in_year;
        fi_year([1+(i-1)*288 :(i)*288],1) = fi_avr(:,i);
    end;
```

Function called manually to calculate projection of disturbance for each year and save in Hproj.m (contains the part of the disturbance of the magnetic field which is produced by cross polar cap ionospheric electric field):

Appendix_A_file_archive\COEFF_CALC\STEP2_fi\ make_hproj.m

```
%% Calculates H_proj for the years
%
%
%%%
function make_hproj

    % load Fi data
    load('Fi_year.mat');

    % load yeartime array
    load('yeartime.mat');

   for year = 1997:2009

      disp(year);
      % load X_dist and Y_dist
      load(['dist_' num2str(year) '.mat']);
      eval (['dist = dist_' num2str(year)]);

      % Calculate H_proj
      H_proj = hproj(yeartime(1,:),yeartime(2,:),Fi_year(1,1:105120),
dist.x(1,1:105120), dist.y(1,1:105120));

      % Save H_proj to the file
        eval (['Hproj_' num2str(year) '=H_proj;']);
        fn = ['Hproj_' num2str(year)];
        eval (['save(''' fn ''', ''' fn ''');']);
    end;
```

Function manually started, to call function to calculate regression coefficients a (slope) and b (intercept) between

Appendix_A_file_archive\COEFF_CALC\STEP3_ab\ab_step1

```
%% Calculate parameters a and b for 1997 - 2009
%
%%



function ab_step1;



for year = 1997:2009
   disp(year);
   ab_regress (year)
end;
```

Function to calculate regression coefficients a (slope) and b (intercept) and save those in matrix for each year in files ab_YYYY.mat:

Appendix_A_file_archive\COEFF_CALC\STEP3_ab\ab_regress.m

```
%% Find best linear fit of Ekl and H_proj
%
%
%%
function ab_regress (year);

    % load Esw file (5-min and 15 min shifted forward
    load(['ekls_' num2str(year) '.mat']);
    eval (['ekl = ekls_' num2str(year) ';']);
    %ekl = smooth(ekl,5,'rloess');

    % load H_proj file
    load(['Hproj_' num2str(year) '.mat']);
    eval (['H_proj = Hproj_' num2str(year) ';']);
    %H_proj = smooth(H_proj,5,'rloess');

    %load time file
    load('yeartime.mat');
    time_a = yeartime';


% Correlation

    for mn=1:12
        disp(mn);
        for hr=0:23
            for min =0:5:55;

                day = 16+30*(mn-1);
                day_min = day-15;
                day_max = day+15;

                min5 = day*288;
                min5_min = day_min*288;
                min5_max = day_max*288;
                clear temp*;
                clear tcorr*;

                temp1 = H_proj(min5_min:min5_max)';
                temp_dt = time_a(min5_min:min5_max,:);
                temp_ace = ekl(1,min5_min:min5_max)';


                temp_pos = find(and(temp_dt(:,1)==hr,temp_dt(:,2)==min));
                tcorr_Esw = temp_ace(temp_pos);
                tcorr_Hproj = temp1(temp_pos);

                tcorr_Hproj(find(isnan(tcorr_Esw)))=[];
                tcorr_Esw(find(isnan(tcorr_Esw)))=[];
```

```
                tcorr_Esw(find(isnan(tcorr_Hproj)))=[];
                tcorr_Hproj(find(isnan(tcorr_Hproj)))=[];

                R = polyfit(tcorr_Esw,tcorr_Hproj,1);
                ab.a(mn,(hr.*12)+(min+5)/5) = R(1);
                ab.b(mn,(hr.*12)+(min+5)/5) = R(2);


            end; % for min=
        end; % for hr=
    end; % mn


% Save regression coeff to file ab_YYYY.mat

eval (['ab_' num2str(year) '=ab;']);
fn = ['ab_' num2str(year)];
eval (['save(''' fn ''', ''' fn ''');']);
```

Stacks for all years matrix with a and b coefficients for all months and all UT-times and smoothes this matrix such that hour 23 and hour 00 (and month 12 and month 1) have smooth boundary. Saves matrix in ab_2d.mat and plots matrix. Calls function to calculate a and b every 5 minutes for the year and saves that as ab_year.mat.

Appendix_A_file_archive\COEFF_CALC\STEP3_ab\ab_step2.m

```
%% Average a and b coefficients
%
%
%%

function ab_step2;

% load all fi files from 1997 to 2009
i=0;
    for year = 1997:2009
      i=i+1;
      name = ['ab_' num2str(year)];
      eval (['load(''' name ''');']);
      eval (['a_years(:,:,' num2str(i) ') = ' name '.a;']);
      eval (['b_years(:,:,' num2str(i) ') = ' name '.b;']);
    end;
% avarage
   a_avr = nanmean(a_years,3);
   b_avr = nanmean(b_years,3);

   c_avr = a_avr;
     c_e = [c_avr c_avr c_avr ; c_avr c_avr c_avr ; c_avr c_avr c_avr];
     c_ex(:,:,1)= c_e;
     c_ex(:,:,2)= c_e;
     c_ex(:,:,3)= c_e;
```

```
     c_ex(:,:,4)= c_e;
     c_ex(:,:,5)= c_e;
     c_s = smooth3(c_ex,'gaussian',7,0.8);%7
     c_n = smooth3(c_s,'box',5);%5
     c_2d = c_n(13:12*2,289:288*2,3);
     ab_2d.a = c_2d;

   c_avr = b_avr;
     c_e = [c_avr c_avr c_avr ; c_avr c_avr c_avr ; c_avr c_avr c_avr];
     c_ex(:,:,1)= c_e;
     c_ex(:,:,2)= c_e;
     c_ex(:,:,3)= c_e;
     c_ex(:,:,4)= c_e;
     c_ex(:,:,5)= c_e;
     c_s = smooth3(c_ex,'gaussian',9,0.1);%7
     c_n = smooth3(c_s,'box',7);%5
     c_2d = c_n(13:12*2,289:288*2,3);
     ab_2d.b = c_2d;

 subplot(2,1,1);
 contourf(ab_2d.b,12);
 subplot(2,1,2);
 contourf(ab_2d.a,12);



 ab_year.a = coeff_for_year(ab_2d.a)';
 ab_year.b = coeff_for_year(ab_2d.b)';
 ab_year.a=smooth(ab_year.a,24,'lowess');
 ab_year.b=smooth(ab_year.b,24,'lowess');

 save('ab_2d.mat','ab_2d');
 save('ab_year.mat','ab_year');
```

MAKE PLOT OF a, b !!!

Function to calculate a and b every 5 minutes for the year by interpolation:

Appendix_A_file_archive\COEFF_CALC\STEP3_ab\ coeff_for_year.m

```
%% Interpolate 1-D array from 2-D array of coefficients matrix;
%
%%

function fi_year = coeff_for_year(f_avr);


    days_in_year = 366;
    f_tmp =[f_avr f_avr f_avr;f_avr f_avr f_avr; f_avr f_avr f_avr];
    [X,Y] = meshgrid([15 :days_in_year/12:days_in_year*3-15],[1:288*3]);
    [XI,YI] = meshgrid([1:days_in_year*3],[1:288*3]);
    fi_tmp1 = interp2(X,Y,f_tmp',XI,YI,'spline');
    fi_avr = fi_tmp1([288+1:288*2],[days_in_year+1:days_in_year*2]);
```

```
    for i=1:days_in_year;
        fi_year([1+(i-1)*288 :(i)*288],1) = fi_avr(:,i);
    end;
%plot(fi_year);
```

Phi, a and b are interpolated to 1-minute resolution and combined in one file:

Appendix_A_file_archive\COEFF_CALC\STEP3_ab\ make_coeff_1min.m

```
%% Make 1-min file of coefficients
% from 5-min avaraged files
%
%%

function make_coeff_1min;



    % load coefficient files
    load('Fi_year.mat');
    load('ab_year.mat');

    % interpolates the coefficients to 1-minute resolution
    coeff.f = interp(Fi_year,5);
    coeff.a = interp(ab_year.a,5);
    coeff.b = interp(ab_year.b,5);

    %save all coeff to file
    save('coeff.mat','coeff');
```

Script to start up PCN index calculation for a number of years:

Appendix_A_file_archive\PC_CALC\years._pc.m

```
%% Calculates PC for some years
%
%%

for year = 1997:2009

    pc_db([year 1 1 0 0], [year 12 31 23 59]);

end;
```

Function to retrieve THL data, sector structure and QDC from database, calculate PCN index and store PCN index in database:

Appendix_A_file_archive\PC_CALC\ pc_db.m

```
%% PC index to DB
%
%%
function PC = pc_db(start_dt, end_dt);
% ------------------------------------------------------
% The function calculates PC and insert the index to DB
% ------------------------------------------------------
%=========
% STEP 1 =
%=========
% Read data from the DB
   % Read e and h components from the DB
    x = dbget ( [start_dt,00],[end_dt,00],'x');
    y = dbget ( [start_dt,00],[end_dt,00],'y');
   % Read ss_e and ss_h (SS variations) from the DB
    ss_x = dbget ( [start_dt,00],[end_dt,00],'ss_x');
    ss_y = dbget ( [start_dt,00],[end_dt,00],'ss_y');
   % Read qdc_e and qdc_h (QDC variations) from the DB
    qdc_x = dbget ( [start_dt,00],[end_dt,00],'qdc_x');
    qdc_y = dbget ( [start_dt,00],[end_dt,00],'qdc_y');
% calculate disturbed part of the variations
    dist_x = x - ss_x - qdc_x;
    dist_y = y - ss_y - qdc_y;

%=========
% STEP 2 =
%=========
% Load load phi, alpha and betta coefficients
    load('coeff.mat');
%=========
% STEP 3 =
%=========
% H Projection and PC calculation
d_start = datenum([start_dt,00]);
d_end = datenum([end_dt,00]);

i = 0;
for dt = d_start:1/1440:d_end
    i=i+1;
    time_arr = datevec(dt);
    min_in_the_year = round( (dt - datenum([time_arr(1) 1 1 0 0 0])) *1440)+1
% calculate H_proj
   H_proj = hproj(time_arr(4),time_arr(5), coeff.f(min_in_the_year), dist_x(i),
dist_y(i));
% calculate PC
   PC(i) = (H_proj-coeff.b(min_in_the_year))./coeff.a(min_in_the_year);
end;
%=========
% STEP 4 =
%=========
% H Projection and PC to DB
dbsetpc ([start_dt,00],[end_dt,00],'pc', PC);
```

Function to upload PCN to database:

Appendix_A_file_archive\PC_CALC\ dbsetpc.m

```
function dbsetpc ( start_dt,end_dt, pos, varr);
% send data to "pos" column of the database

% read config file db_cong.pc

f = fopen('db_config.pc','r');
    serv = fgetl(f);
    log = fgetl(f);
    pass = fgetl(f);
    dbname  = fgetl(f);
    table = fgetl(f);

fclose(f);

% make db formated time intervals
start_time =  datenum(start_dt);
end_time =  datenum(end_dt);
    s_time = ['"' datestr(start_time,'yyyy-mm-dd HH:MM:SS') '"'];
    e_time = ['"' datestr(end_time,'yyyy-mm-dd HH:MM:SS') '"'];
    (end_time-start_time)*1440;
%set data to db
mysql('open', serv,log,pass);
  mysql(['use ' dbname]);

     k=1;

     for i= start_time:1/1440:end_time
        vr = varr(k);
        k=k+1;
            str = 'NULL';
            if ~isnan(vr);  str=['"' num2str(round(vr*100)/100) '"']; end;
        time = ['"' datestr(i,'yyyy-mm-dd HH:MM:SS') '"'];
        disp (time);
        sqlstr=['update ' table ' set ' pos ' = ' str ' where time = ' time
';'];
        mysql(sqlstr);

     end

  mysql('close');
```