Preliminary project

“”

February 10, 2016

1 Reading in data

The NATLAB function file *read_it_in.m* takes all files in a given folder and reads them in, removing headers and bad data. The final data is more quickly accessed by loading in the resultant matrices.

This needs to be generalised to

- Being clever about distinguishing dates and station
- Saving other data in a sensible way - by month, year, station etc.
- Maybe sped up before running on the whole thing.

2 Reading about Power Spectral densities

Most of this is from “Spectral Analysis of Signals”, Petre Stoica and Randolph Moses.

The purpose of looking at these spectral densities is to see how total power is distributed by frequency. How we do so depends on whether we have a good model for the data (we can then use “parametric” methods) and whether the signal is deterministic or “random”, ie we must study it using statistical averages.

The first thing they look at is the energy spectral density. They can essentially project the energy distribution on orthonormal $e^{-i\omega t}$ (constant ω)(Parseval’s Theorem). This can similarly be found as the discrete-time Fourier transform of the auto-correlation of the sequence.

For the PSD we assume $\mathbb{E}\left\{y(t)\right\} = 0$. We take a function called the *auto covariance sequence* (or ACS) which is the epectation (mean) of two lagged signals $r(k) = \mathbb{E}\left\{y(t)y*(t-k)\right\}$. An assumption is that the signal is second order stationary sequence.

We actually look at the frequency distribution of this ACS r(k). We get PSD definition number one:

$$\phi(\omega) = \sum_{k=-\infty}^{\infty} r(k)e^{-i\omega k} \tag{1}$$

which when integrated gives the power in $\omega \pm \frac{d\omega}{2}$. Definition number two:

$$\phi(\omega) = \lim_{N\to\infty} \mathbb{E}\left\{ \frac{1}{N}\left|\sum_{t=1}^{N} y(t)e^{-i\omega t}\right|^2 \right\} \tag{2}$$

These definitions are equal if $\{r(k)\}$ decays quickly. (so if there is a big lag k the average goes back to zero). This definition is similar to energy spectral density.

We can find ϕ using a single interval as it is periodic. We should, however, check our sampling frequency is good enough to reconstruct signals.

IMPORTANT NOTE: This PSD is telling us about the ACS, and NOT about $y(t)$. That's why we chosse $r(k)$ so carefully.

Side note: if $y(t) \in \mathbb{R}$ then $\phi(\omega)$ is even.

Since the ACS depends on the lag we can rewrite $\phi(\omega)$ to make this more explicit; we can write $\phi(\omega)$ in terms of the PSD of the input or in terms of the PSD of a frequency-shifted signal.

When we are actaully calculating the PSD we must estimate the using finite amounts of data. We can use a *periodogram*

$$\hat{\phi}_p(\omega) = \frac{1}{N}\left|\sum_{t=1}^{N} y(t)e^{-i\omega t}\right|^2 \tag{3}$$

or a *corellogram*.

$$\hat{\phi}_c(\omega) = \sum_{k=-(N-1)}^{N-1} \hat{r}(k)e^{-i\omega k} \tag{4}$$

where $\hat{r}(k)$ is an estimate of $r(k)$ using the finite amount of data. We'll use the *standard biased ACS estimate*

$$\hat{r}(k) = \frac{1}{N}\sum_{t=k+1}^{N} y(t)y*(t-k) \qquad 0 \leq k \leq N-1 \tag{5}$$

.

The corellogram and periodogram are then the same. But they actually give poor PSD estimates because

- We are summing many small errors - these increase with N
- Estimations don't improve as N increases
- $\hat{r}(k)$ decays more quickly than $r(k)$ does. This is a problem if $r(k)$ isn't rapidly enough decaying.

If we calculate $\hat{\phi}_p$ using frequency samples it turns out we're basically just computing a DTFT. Is this useful to us? So FFTs are useful.

There are several problems with the periodogram - we encounter *leakage* and *smearing*. There is a spectral resolution limit associated with these. Bias is related to resolution. (The mean squared error in our estimate is a mix of the variance and bias). We can reduce bias by increasing N but doesn't help. Basically this estimate fluctuates around the PSD with a non-zero variance (even if large N). We get uncorrelated random variables and not the PSD required.

We need to modify this method. To do so we may have to increase bias (and hence decrease average resolution).

We need to get rid of the high statistical variability that is the main problem. Use the *Blackman-Tukey estimator*

$$\hat{\phi}_{BT}(\omega) = \sum_{k=-(M-1)}^{M-1} \omega(k)\hat{r}(k)e^{-i\omega k} \tag{6}$$

where the *lag window* $\{\omega(k)\}$ is even, $\omega(0) = 1, \omega(k) = 0$ for $|k| \geq M$, $\omega(k)$ decays smoothly with k and $M < N$.

Again this can be calculated via DTFT (maybe you should writ ehtis down?) This allows us to use one of a class of windows to get a "locally weighted average" of the periodogram. This snmoothing does reduce the resolution,. Variance: on order M/N. Spectral resolution: on order $1/M$.

They then talk loads about window design and choice but I don't need that yet. Window length determines the M and hence variance and resolution. *(We should generally choose $M \leq N/10$ to reduce st dev of estimated spectrum at least 3 times (wrt periodofgram)). The window shape trades off between leakage and smearing effects. You may want to read this bit to know how to pick a window.

The Blackman-Tukey estimator $\hat{\phi}_{BT}$ is essentially a lag-windowed correlogram. We can also use a lag-windowed periodogram $\hat{\phi}_W$. These have the same average behaviour but relation between $\hat{\phi}_W, \hat{\phi}_{BT}$ is not clear. So use the correlogram version! It's motivated more sensibly.

They then describe example spectral estimate methods. These may be worth reading, as well as computing various bits using fft which sounds useful. The Blackman-Tukey method cannot be calculated in a single FFT as in the unwindowed periodogram.

LATER NOTE: The Blackman-Tukey estimator is a windowed correlogram. (windowed with a lag window $w(k)$). However, we can rewrite this to see that we essentially have here a periodogram windowed with $W(\omega)$ the DTFT of $w(k)$, called a spectral window (via convolution).

$$\hat{\phi}_{BT}(\omega) = \hat{\phi}_p(\omega) * W(\omega) = \frac{1}{2\pi}\int_{-\pi}^{\pi} \hat{\phi}_p(\psi)W(\omega - \psi)d\psi \tag{7}$$

We can use windows on periodograms too - as we are summing over time t they are temporal windows. It is between the windowed correlograms (with lag window) and the windowed periodograms (temporal window) that the expectation is the same but other behaviour is not.

3 Co-ordinate systems

Usually X,Y,Z but there are other co-ordinates F (total intensity), H (horizontal intensity), I (inclination - angle between horizontal plane and field vector, positive downwards) and D (declination/magnetic variation - horizontal angle between true north and the field vector, positive eastwards)

$$\begin{aligned} D &= \arctan\frac{Y}{X} \\ I &= \arctan\frac{Z}{H} \\ H &= \sqrt{X^2 + Y^2} \\ F &= \sqrt{H^2 + Z^2} \end{aligned}$$

OR there is a separate system where H is the X coordinate wrt the magnetic north pole (ie trnasverse) and D points towards magnetic north.

4 Fixing data spikes

The easiest method to remove these seems to be to put in a threshold. However it may be more sensible to just look at the average difference between neighbouring values.

Looking at data sets, some bits are marked as dubious (?) when they are still very similar to the previous data. Should I even be cutting out all these non-. data?

Actually the data from Sept 2001 is ridiculously off in some places. Using a threshold makes it VERY tight, although it may be the only sensible option.

Correct that - some clearly off data is within a threshold. So I will need threshold AND better system (or just other system, if good enough).

5 Description of plot-D-ground-PSDs code

The final goal is to plot the PSDs of the in situ electric field, binned by solar wind speed. In the companion paper to the one I am reproducing, there are plots of ground D-component PSD against frequency for several stations at Kp values 1,3,6. I wanted to use this to compare my progress.

This is mostly doen, the only problem being that the amplitudes are several orders of magnitude off. Clearly this must be fixed, but everything else is fine. So I'll step through the code so far, then copy everything into a dated folder.

The Setup

- time resolution - use to caluclate smapling freq
- load in some of the pre-processed data (headers, bad data stripped out, saved into quick MATLAB matrix format)
- Setup of this is done in a separate function (setup-C-data) which cuts out data outside of a preset threshold, converts to H and D components, sorts into hours and removes any hours that are not completely full of good data. The format of these matrices are now three dimensional, with (X,Y,Z), X - 720 readings of the magnetometer,Y - date, h, d components, and Z is a layer for each slice (so Z $\sim$ 24 if we are looking at a single day or 720 for a month) eg for Z = 1, X = 1,2,3:

DATE(hour Z1, min1, sec1)	H(1)	D(1)
DATE(hour Z1, min2, sec2)	H(2)	D(2)
DATE(hour Z1, min3, sec3)	H(3)	D(3)
⋮	⋮	⋮

and for Z = 2

DATE(hour Z2, min1, sec1)	H(1)	D(1)
DATE(hour Z2, min2, sec2)	H(2)	D(2)
DATE(hour Z2, min3, sec3)	H(3)	D(3)
⋮	⋮	⋮

- Next, read in the relevant hours worth of OMNI data. Convert Kp values from the weird 0, 3, 7... to sensible values

- Sort all the data (already sorted by hour) by Kp value. First, find all possible Kp values. Initialise the matrix to hold all these - a matrix (K,X,Y,Z) where K indicates the Kp index. Unfortunately MATLAB values start at 1 whilst Kp values start at 0, so the first matrix K = 1 actually corresponds to Kp = 0. So for each K we have a set of matrices X,Y,Z again, same as before. This does mean there will be some zero XY matrices out there we must be aware of - sadly we can't remove these as MATLAB doesn't have any other kind of structure than matrices. We DO remove any K slices where there are no entries whatsoever. This is less likely to be a problem for larger datasets but still needs to be done. We may want to make all these matrices with NaN rather than zero, as this will throw up problems if not filled rather than being quiet. To allocate by Kp value we must find the matchin OMNI data by hour. We do this with more datevec/datenum stuff. Note that we must keep track of where we are in each K-slice to prevent big gaps full of zeros. THe only other way would be with big temporary matrices. By keeping track, for hours $Z = 1, 2, 3, 4, 5 \ldots$

Kp=0	Kp=1	Kp=2	Kp=3
Z2	Z1	Z5	·
Z4	Z3	·	·
·	·	·	·

rather than big gaps

Kp=0	Kp=1	Kp=2	Kp=3
···	Z1	···	·
Z2	···	·	·
·	Z3	·	·
·	·	·	·

- Finally we go around for each Kp value (each K-slice). Frequencies must be set up in advanc, using the sampling frequency. We keep track of all psds for all K in a variable tot-psd-res, which is also full of useless NaN values. Rather than waiting for this variable to be completely filled, we do each Kp at a time, finding teh PSDs for each hour, adding them to a temporary variable, finding the median and then adding it to the plot. Note that we don't bother doing any of this unless it's a dayside value, as per the companion paper.

- Plotted!

6 Description of code (in-prog) as of 14 Aug

I've tried to move all options up to the top. The code has been rewritten so the matrices I'm using are less complicated.

First of all we load in data. Load in and setup CARISMA data, load in OMNI data and convert Kp to 0,1,2,3 etc.

Next, set up the main datasets I'll be using. We need a 4x#hrs matrix hr-info to hold all the information about each slice. For each num-hr this should hold the SW speed, Kp value, Eeq value and the hour (1-24) associated to the slice num-hr. We also bin the SW speed.

Then find the PSDs. These live in a 360x3x#hrs matrix (we lose the date/time column from hr-data).

First optional part: convert to equatiorial PSD using the equation

$$PSD^E_{eq} = \left[\frac{E_{eq}}{b_g}\right]^2 PSD^b_g \tag{8}$$

Second optional part: plot according to Kp values. Sort - for each Kp values, find all slices with that Kp and in dayside, then plot the median PSDs of this.

Third optional part: plot by SW speed bin. The guts of this are the same as plotting by Kp. It can currently do all 4 parts of the day on different plots but they just aren't the expected results. Plotting these just by ground PSD gives plots much more similar to the paper than eq. PSD. So I'm calculating that wrong, as I thought. This is quite messy now as it's plotting many graphs.

Next bits to do:

- Deal with loading in data -
    1. expand to all of the data,
    2. saved to matlab file once sorted by hour (so you import 720x4x#hrs matrix)(instead of sorting by hour every time)
    3. check all this data for thresholding OK
    4. get all OMNI data too.
- Tidy SW speed binning so that bin number translates when plotting - t must go nicely on legend
- Tidy up finding PSD so it's not being called three times.
- Consider also saving the PSDs found in MATLAB file too.
- Work out why we're not getting several SW lines per plot.
- Discover the missing order of magnitude when plotting Y-component ground PSDs by Kp (as per Ozeke paper)
- Find out why the equatorial PSD plots of X,Y are several orders of magnitude out
- Understand the Eeq ratio bit - how did they get and use this value? Am I calculating bg correctly? This will liekly answer the point above too.
- Can I merge the plotting functions? This would be tidier
- Also it turns out I mnisread what was in the plots. You can do the ground-based ones in Figh 1 of Rae easily - try it!

7 Fixing the PSD

So I finally got my graphs to be in the right magnitude region. The FFT code is as described in the paper. Understanding the PSD is easier doen with Kyle Murphy's thesis. The third definition for PSD he uses is

$$PSD_k = \frac{1}{\Delta f}\mathcal{P}_k \tag{9}$$

which settles Parsevals theorem

$$\sum_{k=0}^{N/2} PSD_k = \frac{1}{N\Delta f}\sum_{n=0}^{N-1} |x_n|^2 \tag{10}$$

with $\mathcal{P}_k = 2\,|F_k|^2$. Note that the factor of two here represents the double summation of positive and negative frequencies (they are even so only half the values need counting). If we use a window (which we did) the correction factor should be included: $\mathcal{P}_k = \frac{2}{W} |F_k|^2$. I've used $W = \sum_{n=1}^{N} w_n^2$, since we square each contribution form the Hanning window $[w_1\ w_2\ \dots w_N]$ when calculatin $\mathcal{P}_k$.

Note that when looking at Jonny's code, data_corr is equivalent to N/W. Also I'm not sure why there's a factor of N in the formula for W in the paper - maybe to counteract the factor of N in IDL I'll talk about next.

So in IDL the fft calculation has a random $\frac{1}{N}$ at the front, compared to MATLAB. That means that my final coefficients for the PSD are:

$$PSD_k = \frac{2N}{WF_s} \frac{1}{N} |F_k|^2 = \frac{2}{WF_s} |F_k|^2 \tag{11}$$

In the paper and in the other code they use $\Delta f = 1/N\Delta t$, but I jump straigh to using F_s as it includes the milli factor I also used in the calculation of the associated frequencies.

(that is: $F_s = \frac{10^3}{\Delta t}$ and then frequencies are $\frac{F_s}{N} * [0\ 1\ 2 \dots N/2]$ Note the same factor $\frac{F_s}{N}$!)

8 Code description as of 11 Sept

Code is read in to matlab format (good data only) by a Python script *read_in_CANOPUS.py*. Data is saved the folder /data/processing in a file called station-raw-year-month, eg *GILL_raw_1990_1.mat*.

Then MATLAB processing is done, mostly in a single file *matlab_processing.m*. The stages of this are done in *data_prep.m* (converting to magnetic local time and recalculating correct datevec, sticking a datenum column on the front, rotating XY coords to magnetic north) after which point the data is in format

DATENUM	YEAR	MONTH	DAY	HOUR	MIN	SEC	X-VALUE	Y-VALUE	Z-VALUE
DATENUM	YEAR	MONTH	DAY	HOUR	MIN	SEC	X-VALUE	Y-VALUE	Z-VALUE
DATENUM	YEAR	MONTH	DAY	HOUR	MIN	SEC	X-VALUE	Y-VALUE	Z-VALUE
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

saved under *GILL_prepped_1990_1.mat* etc.

The next part of this is sorting by hour (*sort_by_hour.m* and *fix_moved_hours.m*) so that we now have 720 x 10 x #hours data for each month. These stages are *GILL_sorted1_1990_1.mat* and *GILL_sorted2_1990_1.mat*.

The final stage is thresholding, using *do_thresholding.m*. This is last because it may need different fiddling for each station. Any hour slices with missing data are removed. This final data is saved as *GILL_1990_1.mat*.

There are some functions I wrote to help with analysing this thresholding function - *plot_hourly_amps.m* and *plot_hourly_median_amps.m* do as their names suggest. There is also a function *plot_thresholded_hours.m*, highly useful, which plots any hours that have been thresholded out but are near the threshold. This should help identify sensible limits.

Criterion for keeping data is that it should be smooth. Although there are some mini-spikes in the data which are clearly wrong, I already know it's very difficult to identify them solely by "spike" behaviour. This means that although I don't exclude particularly active hours, nor do I exculde mini-spikes. I also don't threshold X,Y values - I may consider doing this but I don't think it's necessary.

I'm currently writing out plotting by SW speed /Kp value code into better format.

9 Changes in Dec

We now remove data when corresponding OMNI data is missing inside teh preprocessing (step six). This means that after thresholding, data is saved as GILL-sorted3-year-month, and after omni-sorting it is saved as GILL-year-month. This also should include the corresponding omni data mini-omni.

Also made a backup of code and the data needed to build from scratch: Bentley.zip, station-decs, station-mlt-midnights, omni2-all-years

The code is now run from two files: matlab-preprocessing (with omni data removal moved to here) and plotting-and-psds, which calls calculate-psds if required, and get-median-psds, which it then plots. We need to bin SW speeds when we want to use them - they are now always stored as speed rather than bin.

See entry in diary of 18^{th} Dec for new work directions.

10 4th Jan

Removed rotation to magnetic north, all preprep files now save over results file even if it already exists. This changed data_prep, sort_by_hour.

Put back in soon after. ALOS NOTE: git now used for versioning rather than this document, as of Feb 2016.

11 More on PSDs

First thing, the frequency axis after a Fourier transform - you need to convert the index to a physical value. This can be done two ways. The first is to work outthe frequency resolution, $f_s = frac1T = \frac{1}{N\delta t}$ where N is the number of points in the data set and δt is the time resolution, or the time between sampling, so here it is 5s. Then teh frequency axis is $\{f_0, f_1 = f_s, f_2 = 2f_s, \ldots\}$ up to Nyquist frequency $f_{Ny} = \frac{1}{2\delta t}$.

The other way to look at it is the maximum frequency we can distinguish, $F_{max} = F_{Ny} = \frac{1}{2\delta t}$. Then we have $\frac{N}{2}$ steps from zero to F_{max}, so $F_{max} * linspace(0, 1, N/2)$.

Now, talking about PSDs. Parseval's identity originally links the the norm across two orthonormal bases, to say that they are equal in both. Since the norm is often some measure of "energy" this establishes conservation of energy between two bases - for you, the time and frequency domains. This is why our definition of PSD must satisfy Parseval! And why we extend "power" P_k to get this definition of PSD instead.

They use the definition of PSD as:

$$PSD_k = \frac{1}{\delta f} P_k = \frac{1}{\delta f} 2\left|F_k\right|^2 \quad k \in \left[1 : \frac{N}{2} - 1\right] \tag{12}$$

(and $P_k = \left|F_k\right|^2$ $k = 0$ and $N/2$?? where his Fourier transform is

$$F_k = \frac{1}{N} \sum_{n=0}^{N-1} x_n \exp\left[\frac{-2\pi i k n}{N}\right] \tag{13}$$

So using this we can prove Parseval's theorem:

$$\sum_{k=0}^{\frac{N}{2}} PSD_k = \sum_{k=0}^{N-1} \frac{1}{\Delta f} |F_k|^2 \tag{14}$$

$$= \frac{1}{\Delta f} \sum_{k=0}^{N-1} \left(\left| \frac{1}{N} \sum_{n=0}^{N-1} x_n \exp \left[\frac{-i2\pi nk}{N} \right] \right| \right)^2 \tag{15}$$

$$= \frac{1}{\Delta f} \sum_{k=0}^{N-1} \left(\left| \frac{1}{N} \sum_{n=0}^{N-1} x_n \exp \left[\frac{-i2\pi nk}{N} \right] \right| \right) \left(\left| \frac{1}{N} \sum_{m=0}^{N-1} x_m \exp \left[\frac{-i2\pi mk}{N} \right] \right| \right)^2 \tag{16}$$

$$= \frac{1}{\Delta f} \frac{1}{N^2} \sum_{k=0}^{N-1} \sum_{n=0}^{N-1} \sum_{m=0}^{N-1} |x_n| \, |x_m| \exp \left[\frac{-i2\pi(n-m)k}{N} \right] \tag{17}$$

$$= \frac{1}{\Delta f} \frac{1}{N^2} \sum_{k=0}^{N-1} \sum_{n=0}^{N-1} |x_n|^2 \tag{18}$$

$$= \frac{1}{N\Delta f} \sum_{n=0}^{N-1} |x_n|^2 \tag{19}$$

This means that for your PSD to be worth anything, you must use the same Fourier transform (ie N in the same place) But. . . how can this definition of PSD be window-length independent? We have to use it each time? Why is this the one he uses? He says this is from Brautigam, 2005, so I'll read that to find out.

In the meantime I've found the correct (possibly) PSD to use. Since $F_k^{IDL} = \frac{1}{N} F_k^{MATLAB}$, but actually $F_k^{paper} = F_k^{MATLAB}$, we have

$$PSD_k = \frac{2}{\Delta f W} |F_k^{paper}|^2 \tag{20}$$

$$= \frac{2}{\Delta f W} |F_k^{MATLAB}|^2 \tag{21}$$

But this is a different definition. Does it now satisfy Parsevals, at the minimum by proportionality? Yes - you get

$$\sum_{k=0}^{\frac{N}{2}} PSD_k = \sum_{k=0}^{\frac{N}{2}} \frac{2}{\Delta f W} |F_k|^2 = \ldots = \frac{1}{\Delta f} \sum_{k=0}^{N-1} |x_n|^2 \tag{22}$$

So this satisfies the spirit of Parsevals equation, in that energy in the frequecny domain is proportional to enrgy in the time domain, but differs slightly from other descriptions of PSD. There are slightly different constants in both numerical recipes and Kyle Murphy's masters thesis. As long as we're looking at patterns rather than exact quantities it's fine. . . hopefully this extends to being window-length independent!