LIMS/NIMS ELECTRO-MAGNETIC DATA ARCHIVING

Lana Erofeeva & Gary Egbert, 2006

1. Introduction

LIMS/NIMS data archiving involves different kinds of software and is done in few stages described below. Basically each step converts from one data format to other to provide SEED format required by IRIS on output. (For SEED format description see http://www.iris.edu/manuals). The process of SEED volume production is illustrated by the scheme in Fig.1:

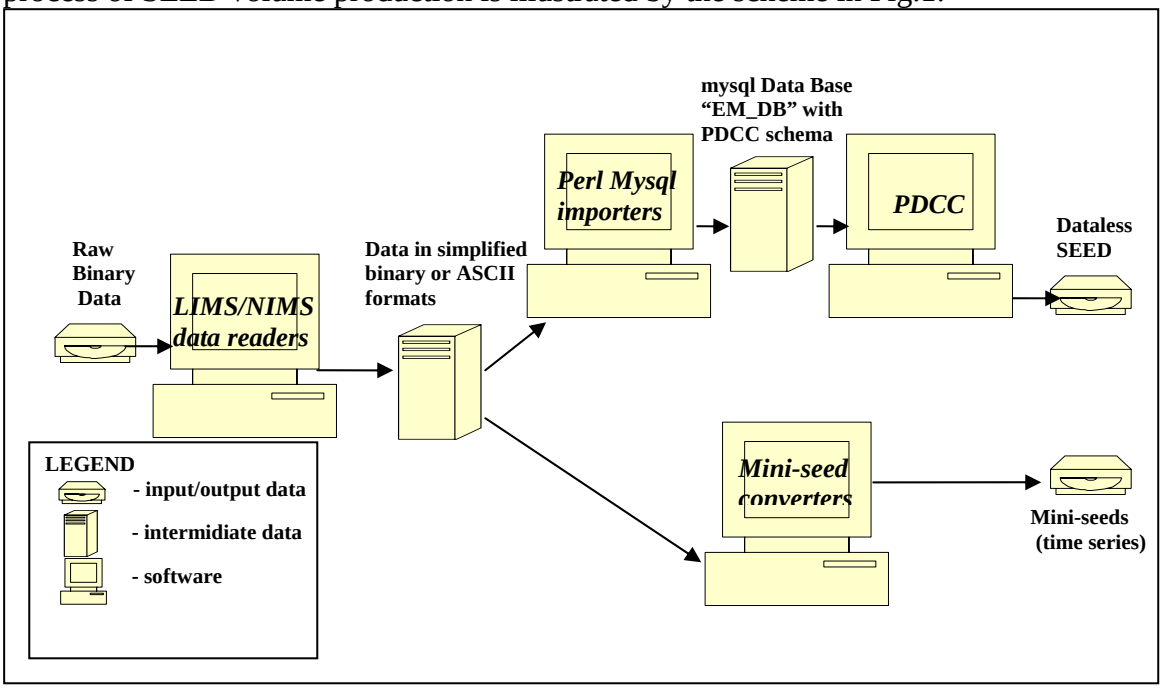


Fig.1

Raw binary data on input are in one of 3 formats:
-"*.1mp" format binary data produced by LIMS;
- *.bin format produced by NIMS MT1;
- hourly binary data files produced by NIMS HP200.

These format data are read with John Booker's LIMS/NIMS data readers written in Fortran: "mp2fts4seed" (LIMS) and "nimsread" (NIMS) and thus converted to simplified ASCII (* fts for LIMS) or binary (*bnn for NIMS) format data. Then processing split in two branches: dataless SEED production and mini-seeds production. Dataless SEED is produced first by importing data into mysql data base EM_DB (with a schema supported by PDCC (Java GUI by IRIS), see http://www.iris.edu/manuals/pdcc3.0/index.html) using Perl scripts, called in Fig.1 "mysql importers" (written by Lana Erofeeva), and second, by converting the data from EM_DB to Dataless SEED volume using the PDCC interface. The whole process is controlled by csh scripts and at the very last stage by the PDCC.

2. Installation

To support LIMS/NIMS data archiving process the following software should be installed:

- Perl;
- mysql data base(see http://dev.mysql.com/doc/mysql/en/Installing.html, freeware). As soon as the software is installed you need to run mysqld daemon as described in the manual at the link above;
- mysql Perl API (see http://dev.mysql.com/doc/refman/5.0/en/perl.html). It provides a generic interface for mysql database access. To use DBI, you must install the mysql DBI module, as well as a mysql DataBase Driver (DBD) module;
- PDCC (Portable Data Collection Center, see http://www.iris.edu/manuals/pdcc_intro.htm)
- LimsNims archiving package (this package of of csh/perl/fortran codes suppoting the archiving scheme described in the Introduction).

As soon as every software above is installed, the following environmental variables should be defined in your .cshrc.paths: LimsNimsBin (path to location where LimsNims package is installed followed by /bin); plbin (path to where Perl is installed); pdcc (path to where PDCC is installed), i.e.

```
setenv LimsNimsBin /home/gauss/lana/LimsNims/bin
setenv plbin /home/server/local/bin.solaris2x/perl
setenv pdcc /home/gauss/users/lana/PDCC3
```

All these paths should be added to your PATH environmental variable.

At last the file mysqlInfo in LimsNims/bin should be edited. mysqlInfo is a 3 line ASCII file, defining mysql data base name corresponding to the PDCC schema, user name and user password, i.e.

```
DB=EM_DB
USER=serofeev
password=mypwd
```

NOTE: This information should exactly correspond to what is shown in the lines

```
dbName EM_DB
userName serofeev
passWord mypwd
```

of the PDCC configuration file PDCC/config/PDCC_config.txt. See PDCC manual on how to configure PDCC. Further in this document the data base is referred as *EM_DB*.

NOTE! Multi-user usage is not implemented for PDCC. Remedy: cp PDCC3 to your destination and edit the files as described above.

3. LimsNims Usage

LIMS data archiving

Suppose the LIMS data files are placed into a working directory, called by experiment name (i.e. TBT). For LIMS these files are organized in subdirectories for each station, called by a station name. Each station subdirectory contains binary time windows data files in *.1mp format. Files

hardware.<experiment_name> and *<experiment_name>site.flt* should be also provided on upper level (i.e. in TBT). These are *ascii* files, containing common hardware and survey information, such as filter parameters, latitude, longitude, elevation for each station etc.

TBT

hardware.tbt
tbtsite.flt
tbt504

tbt504a1.1mp
tbt504b1.1mp
tbt504c1.1mp
tbt504c2.1mp
tbt504z1.1mp
tbt504.log

Experiment directory in this example is called **TBT**. LIMS file name convention is accepted, that is experiment name for the example above is "tbt" and the only station name is "tbt504". Last 2 characters before "*.1mp*" extension define time window, i. e. *a1, b1, c1*.

Files **.1mp* contain header information and time series for different time windows. File *tbt504.log* is used to compute clock offset and drift (set to zero, if the file does not exist). This *ascii* file usually comes along with **.1mp* files.

To archive these data go into the directory **TBT** and run:

>arc_lims tbt>tbt.log

Here the script verbose output is redirected to log file *tbt.log*. Script *arc_lims* performs all stages of data archiving described in the Introduction and is half-automatic, since at the end it uses PDCC graphical interface, popped up at the screen, and requires some user interference. As of now PDCC does not allow running in a batch mode: as soon as it does, the next version of the LimsNims archiving scripts will be modified to be fully automatic. For now the following user action within PDCC graphical interface are needed (from pop-down PDCC menus):

1. File/Import/from Database
2. File/Export/Dataless SEED file
3. Exit

NOTE: Please wait patiently without clicking till PDCC is importing/exporting data. The waiting time might be quite substantial, depending on volume of the data you are trying to archive.
As a result the dataless SEED file with a chosen name (from PDCC prompt) will be created and bunch of mini-seeds will be created in **TBT/mini_seeds** subdirectory. Station codes in SEED file should contain only 5 characters while LIMS station names always have 6 characters, i.e. tbt504 in the example above. The coding rule for LIMS is to take capital of the first 2 letters and the 3 digits, i.e. for tbt504 the code will be TB504.

NIMS data archiving

NIMS data archiving success strongly depends on the raw binary files header contents (headers are ASCII and readable/editable with any editor, allowing to edit large files, such as "Textedit" on

Unix). These headers contain common hardware and survey information which is just typed in by NIMS users and might thus contain typos.
An example of the CORRECT header is shown below:

>>>>> User field
Next four lines are REQUIRED information; syntax is important
"org006a" <-- EXPERIMENT CODE/SITE NUMBER/RUN LETTER
2606-014 <-- SENSOR ID ; System box ID (if different)
100. 0. <-- N-S E WIRE LENGTH (m); HEADING (deg E mag N)
100. 90. <-- E-W E WIRE LENGTH (m); HEADING (deg E mag N)
Calib <-- N ELECTRODE ID
Calib <-- E ELECTRODE ID
Calib <-- S ELECTRODE ID
Calib <-- W ELECTRODE ID
Calib <-- GROUND ELECTRODE INFO
COMMENTS: High Pass filter test

Header lines used in archiving process are emphasized with bold. Syntax of the lines is really important (as noted in the second line of the header), but this warning is often ignored by field users. It is also important for the experiment code (line 3) to exactly correspond to the subdirectory name where the binary file with the header (or header.tmp) is located.

Another issue causing errors in data archiving is missing GPS records in raw data files. GPS records provide latitude, longitude, elevation, declination and starting time of the time series. If such record is missing then no archiving for the data is possible, unless a file *guessed_info* is provided. Example of the file content:

org007a 45.276113 -119.634113 861.9 16.2 2006-08-15 21:43:22
org007b 45.276113 -119.634113 861.9 16.2 2006-08-15 22:54:03

Where one should take this information from if no GPS record is provided? There are few possibilities:

- from other time windows for the station which do come with GPS records;
- from ASCII files, provided by NIMS users, called <station/window_name>.txt, containing parth of this information;
- starting time are figured with matlab GUI and script developed by Gary Egbert;
- declination can be calculated using calculator at http://www.ngdc.noaa.gov/seg/geomag/jsp/struts/calcPointIGRF

NOTE: Make sure that declinations are the same for all time windows, including the one with missing GPS record. Otherwise multiple set of channels for the same station will be produced. The “declination rule” is:

- if there are previous time windows with GPS records, use declination values for these windows;
- if no previous time windows with GPS record exist, use calculated declination value. However, if time windows with GPS record come in future, this value should be corrected.

This process is not automated so far, but it should be to avoid spurious channel multiplying in the dataless volume. For now all stations should be visually checked for multiple channels through PDCC GUI or using *rdseed* output.

Suppose some NIMS data files are placed into a working directory, called by experiment name, for example **Pam** (level 0). For NIMS the original data files are organized in subdirectories for groups of stations, called by a primary station index (level 1). Each such subdirectory contains another set of subdirectories for each station/time window (level 2). Each station/time window subdirectory contains binary data files in hourly (HP200 NIMS) or *bnn* (MT1 NIMS) formats (level 3). For HP200 NIMS the file ***header.tmp*** should be also provided in each station subdirectory. As noted above for MT1 NIMS the *data.bnn* files as a rule contain *ascii* headers with common hardware and survey information. In the case such header is missing, the file ***header.tmp*** should be provided. For MT1 NIMS each subdirectory should contain only one *data.bnn* file.

EXAMPLE:

Pam
- *site_locations*
- **400**
  - **pam450n**
    - *header.tmp*
    - *s030918.019*
    - *s030918.020*
    - *s030918.021*
- **800c**
  - **pam852d**
    - *data.bnn*
  - **pam852f**

Experiment directory in this example is called **Pam**. The file *site_locations* contains information about extended station names, required by IRIS[1].

Example of the *site_locations* file:

```
pam450   -27.19501   -63.02934  Amama,ARGENTINA: J.Booker
pam460   -27.06767   -62.64465  Haase,ARGENTINA: J.Booker
pam470   -27.32824   -62.25401  Otumpa,ARGENTINA: J.Booker
```

Automatic script to create/append "*site_location*" file using text files provided by NIMS users is still to be developed. So far the "*site_location*" files have been created by simple editing.

If *site_locations* file is not provided (or entries are missing in the file for some station) the subdirectory names will be taken for extended station names by default. The stations names in the example above are "pam450","pam460" ,… ,"pam878". Different time windows are identified with one extra letter. File *header.tmp* in **pam450n** contains header information for hourly data in *s030918.** files for HP200 NIMS. The file *data.bnn* contains all information needed to archive MT1 NIMS data.

[1] IRIS requires for archived EM data to have site name in a form of Place, State, Country or Place, Country: PI name. This information is often not available if not provided in header.txt files. To get this information it is possible to use Google Earth (knowing latitude and longitude find closest marked location) or the EarthScope MT web site http://www.iris.iris.edu/ESMT/. There are might be some location information in the documents provided on this site.

NOTE: in the case of NIMS archiving upper directory name (i.e. Pam) not necessarily should correspond to first 3 letters in a station name (unlike in a LIMS case). For example, the directory structure might look like:

EarthScope

OR

org006a
org006b
orh005a
orh006a

However, the 3 level directory structure (experiment, group of stations, station& time windows) should be always supported.
As well as for LIMS case station codes in SEED file should contain only 5 characters, while NIMS station names always have 6 characters, i.e. org006a in the example above. The coding rule for NIMS is: if all 3 digits in the station name are not zeroes, take capital of first 2 letters and the 3 digits, else take all 3 letters and last 2 digits. For example for the station name pam852 the code will be PA852 while for ori005 the station code will be ORI05.

To archive **Pam/400** data go into the directory **Pam** and run:

>check_headers 400

You'll get the output like this:

HOURLY data (OLD NIMS) in this directory

SubDir	ExpCode	DipoleX(m)	AzimX	DipoleY(m)	AzimY
400/pam450n/header.tmp	+ pam450n	100.0	000.	100.0	090.
400/pam450p/header.tmp	+ pam450p	100.0	000.	100.0	090.
400/pam460n/header.tmp	+ pam460n	82.0	003.	100.0	090.
400/pam460o/header.tmp	+ pam460o	82.0	003.	100.0	090.
400/pam460p/header.tmp	+ pam460p	82.0	003.	100.0	090.
400/pam460p1/header.tmp	+ pam460p	82.0	003.	100.0	090.
400/pam460p2/header.tmp	+ pam460p	82.0	003.	100.0	090.
400/pam470n/header.tmp	+ pam470n	91.0	017.	100.0	116.
400/pam470o/header.tmp	+ pam470o	91.0	017.	100.0	116.

There are no mistakes in the headers for this example. "+" in second column indicates that EXPERIMENT CODE in header.tmp corresponds to the subdirectory name. It would be "-" otherwise. After editing for errors (if any) it is recommended to re-run *check_headers,* and only when no error will be found, one may start archiving. Because of the header errors (and missing GPS records) in the data it is recommended to do archiving in 3 steps (*mkbin, mkdataless, mkminiseeds*), i.e.

- *mkbin 400>mkbin.log*
- *mkdataless 400*
- *cd 400*
- *mkminiseeds*

It is recommended to look at the log file *mkbin.log* for "FATAL ERRORS: missing GPS record". If such messages are presented in the file, the file *guessed_info* should be created on level 2 (in Pam/400

for this example). Station/time windows with missing GPS record is easy to discover if after running *mkbin* do:
cd 400
ls –l */*.hed

Station/windows for which files *.hed are of zero size does not have GPS records, that is have to be listed in the file *guessed_info* on level 2 (i.e. in Pam/400 for this example). After this file is created (using sources mentioned above) the script *mkbin* should be re-run once again (another option is from every subdirectory (level 3), for which *.hed was of zero size, run *nimsreadz –b* from level 3, i.e. from the subdirectory).

Script *mkdataless*.uses PDCC graphical interface, popped up at the screen and requires some user interference. The same simple user actions as for *arc_lims* within PDCC graphical interface are needed (from pop-down PDCC menus), i.e.:

4. File/Import/from Database
5. File/Export/Dataless SEED file
6. Exit

As a result the dataless SEED file with a chosen name (from PDCC prompt) will be created.

As a result of running *mkminiseeds* a bunch of mini-seeds will be written in **Pam/400/mini_seeds, Pam/800c/mini_seeds, EarthScope/OR/mini_seeds** subdirectories. If the are missing GPS records, please, do not run the script *mkminiseeds* unless you are completely sure about the content of the "*guessed_info*" file. If "guessed_info" was correcte after mini-seeds are done, it is safer to delete all mini-seeds and re-run *mkminiseeds* again.

Recent addition to the package written by Anna Kelbert is Perl script *make_seeds.pl*. The script is attempt to summarize all steps described above. The script checks for errors in the headers (by running *check_headers*) and the existence of GPS records in all *.bin files and exits if something is wrong, giving user a chance to correct errors. Then, the script may be re-run again: it will not repeat the steps that have already been successfully completed but only updates those files that need updating. To enforce the script to update the files listed in *guessed_info* it should be run it with two options, the first option being level 2 archived directory name, and the second option being "*update*". Running the script with all the options, including site, data and time, only updates "*guessed_info"*. The initial version of "*guessed_info"* should be provided.
To run the script *make_seeds.pl* the following additional environmental variables should be defined:

setenv MTdata <path_to_data>/EarthS # data location
setenv MTdataless <path_to_dataless_SEEDs>/EM_ARC # dataless SEED location

Once the script *make_seeds.pl* says that Stage 1 has been successfully completed, this is when all the *.dbn files have been generated[2] and *mt_*_lists* done (see Appendix on intermediate files).
After correcting start times, it is possible to regenerate files just for those sites listed in "*guessed_info"* using

[2] Then Gary could run prepare_dnff.pl, providing he defined another environmental variable
setenv MThome /home/egbert/../EarthScope/OR
where he proceses the data

make_seeds.pl <data_directory_name> update

Then, if processing Stage 1 is successfully completed, user is prompted to continue to generate the dataless SEED and the miniseeds.

Appending existing Dataless Volume

NIMS data might come either in one piece (when an experiment was completed by the time of archiving) or come in a sequence of few pieces, when an experiment is still continuing and archiving should be done in "real time". The strategy for this case approved by IRIS is:

- replace existing dataless volume when new chunk of data arrives;
- add new mini-seeds (without changing old mini-seeds).

LimsNims package is written to support this strategy. Let us consider example of archiving in few steps on example of EartScope data. The data are coming on CDs with time interval approximately 1-2 weeks. We created directory for the data, called EarthS, and put contents of each chunk of data into subdirectories OR1, OR2, OR3 etc. We also created subdirectory OR, where symbolic links to each data subdirectory in OR1, OR2 etc. set as soon as new CDs are in.
Thus subdirectory OR totals data located in OR1,OR2 etc.

Obviously it would be feasible just to re-do whole archiving procedure for the subdirectory OR each time when new CDs come. However to optimize procedure (without re-doing most time consuming steps) the following script sequence is recommended. Let us call a new subdirectory where we put this new data ORk.

1. From EarthS do:
   check_headers ORk
2. edit/correct data.bin headers, if necessary, re-run *check_headers*;
3. *mkbin ORk*
4. Look for "missing GPS records" errors as described above: make file ORk*/guessed_info* if there are missing GPS records, re-run *mkbin* (or *nimsreadz –b*) as described above.
5. Update EarthS*/site_locations* file if necessary (i.e. if new sites previously not listed in the files are presented in the new data set)
   NOTE: steps 1-5 can be accomplished with *make_seeds.pl*, stopping after Stage 1 is successfully completed.
6. Go to EarthS/OR and do:
   ln –s ../ORk/or .*
7. From EarthS do:
   mkdataless OR
8. From EarthS/ORk do:
   mkminiseeds

New dataless SEED created on step 6 and mini-seeds created on step 7 are the data which should be transferred to IRIS.

Appendix: Details of LIMS/NIMS archiving: intermediate stages and files

Hidden (from a LimsNims user) details of data archiving are different for LIMS and NIMS and are described in this Appendix.

Lims/Nims Data readers

Lims/Nims data readers read raw LIMS/NIMS data and convert them into intermediate ASCII or binary formats. These data readers are written in Fortran by John Booker with slight adjustments (introduced by Lana Erofeeva mostly to adopt no-user-interference approach and slight output format changes) *mt2fts4eed* and *nimsreadz*. The source files for these codes are located in LimsNims/srcL and LimsNims/srcN correspondingly. Executables are placed in LimsNims/bin directory during each re-compiling. As a result of *mt2fts4seed* intermediate ASCII *.fts and *.sp files are created and placed into **fts_sp_files** subdirectory on level 1 for LIMS. For the LIMS archiving example in section 2, these will be files **TBT/fts_sp_files/***tbt504a1.fts*, **TBT/fts_sp_files/***tbt504a1.sp* etc. For NIMS as a result of *nimsreadz* binary files *.bin and ascii files *.hed are created and placed on level 3 into **<Experiment_Name>/<Group_name>/<Station/window_name>** subdirectory. For LIMS first archiving example in section 2, these will be files **Pam/400/pam450n/***pam450n.bin* and **Pam/400/pam450n/** *pam450n.hed* etc. These executable calls are made within scripts *arc_lims*, *arc_lnims* and *mkbin*.

mkdataless intermediate files

In the process of archiving some important intermediate files are created on level 1 in LIMS case and on level 2 in NIMS case. These files present extraction of essential information taken from data headers further collected into mysql EM_DB, and then exported to dataless SEED file with PDCC. Looking at these files one can also spot mistakes missed when checking headers or use them for input to other (not included in LimsNims package) processing programs.
These files are:
mt_sta_list
The ASCII file is the table of columns:
<station/time_window>|<instrument>|<lat>|<lon>|<elvation>|<start date & time>|…
<end date & time>|<base_shift_1>|<base_shift_2>|<base_shift_3>|<comments>

mt_sp_list
The ASCII file is the table of columns:
<station/time_window>|<hconv>|<econv>|<azimuthX>|<azimuthY>|<gain>|…
<dipole_lengthx>|<dipole_length_y>|<sampling_interval>|<clock_offset>|…
<sensor_orientatioon_x>|<sensor_orientatioon_y>|<tilt>
See comments on some of the listed in the file parameters below. Note, that if in *mt_sta_list* all time windows are listed, in *mt_sp_file* only windows with different system parameters are listed. That means for later time windows the same parameters will be written into EM_DB. Sometimes these parameters might be different for different time windows (i.e. dipole or azimuth was changed for the next time window). In this case corresponding number of lines will be written into *mt_sp_list* file.

mt_chan_list
The ASCII file is the table of columns, listing all channels for all stations for all time windows as:
<station/time_window>|<channel>|<instrument>|<azimuth/sensor_orientation>|…
<start date & time>|<end date & time>|<low_pass_cutoff>|<high_pass_cutoff>|<comments>

Mini-seed converters

The mini-seed LIMS converter *mt2seed* is originally written in C by Sheghui Lee, using a code converting to "steim" format by Guy Stewart. Few bug fixes were done in both codes, and they were also slightly modified by Lana Erofeeva. The mini-seed converter *nims2seed* was originally based on *mt2seed* and was modified to allow reading *.bin (output by *nimsread*) files and different coding convention by Lana Erofeeva. The converters take time series from *.fts or *.bnn files and output them in mini-seed format put into mini_seeds subdirectory. This subdirectory is located on level 1 for LIMS and on level 2 for NIMS. Mini-seed files are most space consuming and produce most irritation in IRIS when transferred erroneous and should be re-placed. Thus careful attention is required to see if making mini_seeds procedure succeeded for all time windows. Before making mini-seeds start time should be verified in the case when GPS failed. A mini-seed name is combined from station code, channel name, starting date and time and network name (EM in our case). For example for station code ORF09 channel MFE starting on 2006-09-20 18:50:00, the mini_seed name will be ORF09.MFE.2006.263.18.50.0.0.EM (September 20 is 263 day of year 2006).
Mini-seed converter source codes are located in LimsNims/SEED.

Mysql EM_DB tables content.

The tables listed below are filled for every new SEED volume produced with LimsNims package (with the script *mkdataless*).

mysql> select * from data_format_30;

id	Name	family_type	number_keys
1	Steim 1	50	6

mysql> select * from decoder_keys_30;

id	ref_id	seq	decoder_key(p.174 SEED manual,appendix D)
1	1	0	F1 P4 W4 D C2 R1 P8 W4 D C2
2	1	1	P0 W4 N15 S2,0,1
3	1	2	T0 X W4
4	1	3	T1 Y4 W1 D C2
5	1	4	T2 Y2 W2 D C2
6	1	5	T3 N0 W4 D C2

Blockette 30 (2 tables above) is always the same for any EM_DB dataless volume both for LIMS and NIMS data.

mysql> select * from generic_abbrev_33;

id	description	
1	EMSOC Network	
2	NIMS HP200	
3	NIMS MT1	

mysql> select * from units_abbrev_34;

id	unit_name	unit_description	
1	T	Tesla	
2	V/M	Volts/Meter	

3	V	Volts	
4	COUNTS	Digital Counts	

Blockette 33 is obviously different for LIMS and NIMS. Example for NIMS data is shown above. Blockette 34 is always the same both for LIMS and NIMS.

Following are the tables, including common station information.

LIMS example:
mysql> select * from station_50;

id	Station	Latitude	Longitude	Elevation	num_channels	num_comments	site_name	Network_refid	word order	start_time	end_time	Update_flag	Network_code
1	TB720	28.03	91.929	4440	5	1	Tsona, China	1	3210	2001-05-24 11:00:00	2001-06-29 18:42:15	N	EM

NIMS example:
mysql> select * from station_50;

id	Station	Latitude	Longitude	Elevation	num_channels	num_comments	site_name	Network_refid	word order	start_time	end_time	Update_flag	Network_code
1	ORF08	45.794	118.742W	510.3	5	2	Helix, OR, USA	1	3210	2006-09-04 17:43:59	2006-09-25 18:39:37	N	EM
2	ORF09	45.709	117.903W	899.2	5	1	Palmer Junction,OR,USA	1	3210	2006-09-10 21:58:05	2006-10-0415:20:57	N	EM

Blockette 50 contains information for each station in the volume. First column contains station id, referenced in the channel blockette 52 column 2. Channel number for each station can be 5 or10,15 etc. Physically LIMS/NIMS have 5 channels. However if channel parameters changed during experiment (for example dipole length changed), then new group of channels is created (each channel is doubled, despite only one might have different parameter values).

NIMS example:
mysql> select * from station_comment_51;

id	station_id	start_time	end_time	comment_refid	comment_level
1	1	2006-09-04 17:43:59	2006-09-15 20:17:40	1	0
2	1	2006-09-15 21:56:21	2006-09-25 18:39:37	2	0
3	2	2006-09-10 21:58:05	2006-10-04 15:20:57	1	0

Blockette 51 contains comment links for each station in the volume (unless station does not have any comments). For example, station ORF08 (station id =1) has 2 comments acting in time intervals shown in the first 2 rows, columns 3 and 4 of the table. Comment_refid column refers to actual comment from the blockette 31, containing all possible comments for all stations in the volume (two in this example, see the following table). Thus first row of the station comment table corresponds to the first row of comment description blockette, and second comment for the same station corresponds to second row of the blockette 31 table.

NIMS example
mysql> select * from comment_desc_31;

id	class	description	level_units_refid
1	S	High Pass filter test	0
2	S	High Pass filter test NO GPS rec->lat,lon,elev, start_time NOMINAL	0

Blockette 52 contains information on each channel for each stations. Channels *F* are magnetic, and channels *Q* are electric. First letter is "L" for LIMS (low frequency) and "M" for NIMS (medium frequency) in correspondence with SEED manual rule for naming channels. Azimuth to real North for each channel is calculated as azimuth_to_magnetic_North (taken from the raw files header) plus declination.

LIMS example
mysql> select * from channel_52;[3]

id	Station_id	Channel	units_signal_response	Units_calibration_input	Latitude	Longitude	Elevation	Azimuth	dip	Sample_rate	start_time	end_time
1	1	VFN	1	1	28.03	91.929	4440	0	0	0.2	2001-05-24 11:00:00	2001-06-29 18:42:15
2	1	VFE	1	1	28.03	91.929	4440	90	0	0.2	2001-05-24 11:00:00	2001-06-29 18:42:15
3	1	VFZ	1	1	28.03	91.929	4440	0	90	0.2	2001-05-24 11:00:00	2001-06-29 18:42:15
4	1	VQN	3	2	28.03	91.929	4440	0	0	0.2	2001-05-24 11:00:00	2001-06-29 18:42:15
5	1	VQE	3	2	28.03	91.929	4440	90	0	0.2	2001-05-24 11:00:00	2001-06-29 18:42:15

NIMS example
mysql> select * from channel_52;

id	Station_id	Channel	units_signal_response	units_calibration_input	Latitude	Longitude	Elevation	Azimuth	dip	Sample_rate	start_time	end_time
1	1	MFN	1	1	45.7935	-118.74201	509.9	16.7	0	8	2006-09-04 17:43:59	2006-09-25 18:39:37
2	1	MFE	1	1	45.7935	-118.74201	509.9	106.7	0	8	2006-09-04 17:43:59	2006-09-25 18:39:37
3	1	MFZ	1	1	45.7935	-118.74201	509.9	0	90	8	2006-09-04 17:43:59	2006-09-25 18:39:37
4	1	MQN	3	2	45.7935	-118.74201	509.9	16.7	0	8	2006-09-04 17:43:59	2006-09-25 18:39:37
5	1	MQE	3	2	45.7935	-118.74201	509.9	106.7	0	8	2006-09-04 17:43:59	2006-09-25 18:39:37
6	2	MFN	1	1	45.70861	-117.90287	900.7	16.5	0	8	2006-09-10 21:58:05	2006-10-04 15:20:57
7	2	MFE	1	1	45.70861	-117.90287	900.7	106.5	0	8	2006-09-10 21:58:05	2006-10-04 15:20:57
8	2	MFZ	1	1	45.70861	-117.90287	900.7	0	90	8	2006-09-10 21:58:05	2006-10-04 15:20:57
9	2	MQN	3	2	45.70861	-117.90287	900.7	16.5	0	8	2006-09-10 21:58:05	2006-10-04 15:20:57
10	2	MQE	3	2	45.70861	-117.90287	900.7	106.5	0	8	2006-09-10 21:58:05	2006-10-04 15:20:57

[3] Only essential columns are shown in channel_52 table;

Blocket 53 tables describe analog stages of filter systems for infinite response digital filters. Filter system design is different for LIMS and NIMS and described in two separate sections.

LIMS Filter/Decimation Blocketts (53,57,58,61)

Analogue anti-alias six-pole Bessel low-pass filter is applied to LIMS data on each channel. Analogue single-pole Butterworth high-pass filters are applied on the LIMS electric channels only. The calibrated values for both filters are given in the file *hardware.<experiment_name>* provided with the data in "*1mp" binary format, and normally look like this:

Chan	Calib	Low-pass T0(s)	High-pass T0(s)
1	1.00	0.1	0.00
2	1.00	0.1	0.00
3	1.00	0.1	0.00
4	1.00	0.1	30000.00
5	1.00	0.1	30000.00

For magnetic channels the low pass filter is applied on stage 1, for electric channels the high-pass filter applied on stage 1 and the low-pass filter is applied on stage 2. Stage 1 for electric channels includes multiplying by dipole length (m), which is also taken into account when filling the *sensitivity_gain_58* blockette. The low-pass and high-pass filter poles and zeroes, depending on T0, taken from *hardware.<experiment_name>* file, are calculated for each new experiment by *Perl* script PerlL/*import_blkt_53.pl* (called by the control script *arc_lims*).

mysql>select * from poles_zeroes_53;

id	Chan_id	Proto_id	Transfer_fn_type	stage_number	Signal_in_units_refid	Signal_out_units_refid	normalization_factor	Normalization_freq	Number_complex_zeroes	Number_complex_poles
1	1	0	A	1	1	4	58404571357.5925	0	0	6
2	2	0	A	1	1	4	58404571357.5925	0	0	6
3	3	0	A	1	1	4	58404571357.5925	0	0	6
4	4	0	A	1	2	3	1.00000555554012	0.01	1	1
5	4	0	A	2	3	4	58404630311.4163	0.01	0	6
6	5	0	A	1	2	3	1.00000555554012	0.01	1	1
7	5	0	A	2	3	4	58404630311.4163	0.01	0	6

For magnetic channels normalization frequency $f_n=0$, for electric channels $f_n=0.01$, and normalization factors were calculated from complex poles and zeroes as:

$$A_1 = \left| \frac{\prod_j (2\pi f_n - p_j)}{\prod_k (2\pi f_n - z_k)} \right|, \quad A_0 = \left| \prod_j p_j \right| \quad (no\ zeros,\ \ f_n = 0)$$

The poles and zeroes values for T0=0.1 (low-pass) and T0=30000 (high-pass) are shown in the next two tables ("ref_id" column corresponds to the "id" column in the table *poles_zeroes_53*). For example for *chan_id=1*, *ref_id=1*, normalization factor for 6 poles Bessel low pass filter is (here precision is limited to 2 digits unlike in the table above):

(-33.55+i59.90)* (-33.55-i59.90)*(-49.81+i35.02)* (-49.81-i35.02)*(-56.65+i11.57)* (-56.65-i11.57) ~= 5.84e10

For high pass filter (i.e. for *ref_id=4*) we have 1 pole, calculated as -2π/T0=-2π/30000=-2.09e-4, and normalization factor was calculated as $A_1 = \frac{(2\pi \cdot 0.01 + 2.09e-4)}{2\pi \cdot 0.01} = 1.0000006$

mysql> select * from complex_zeroes_53;

Id	ref_id	seq	real_val	imaginary_val	real_error	imaginary_error
1	4	0	0	0	0	0
2	6	0	0	0	0	0

mysql> select * from complex_poles_53;

id	ref_id	seq	real_val	imaginary_val	real_error	imaginary_error
1	1	0	-33.5462372106204	59.9028482625725	0	0
2	1	1	-33.5462372106204	-59.9028482625725	0	0
3	1	2	-49.8079118230134	35.0159649606428	0	0
4	1	3	-49.8079118230134	-35.0159649606428	0	0
5	1	4	-56.6452355851319	11.5669854701223	0	0
6	1	5	-56.6452355851319	-11.5669854701223	0	0
7	2	0	-33.5462372106204	59.9028482625725	0	0
8	2	1	-33.5462372106204	-59.9028482625725	0	0
9	2	2	-49.8079118230134	35.0159649606428	0	0
10	2	3	-49.8079118230134	-35.0159649606428	0	0
11	2	4	-56.6452355851319	11.5669854701223	0	0
12	2	5	-56.6452355851319	-11.5669854701223	0	0
13	3	0	-33.5462372106204	59.9028482625725	0	0
14	3	1	-33.5462372106204	-59.9028482625725	0	0
15	3	2	-49.8079118230134	35.0159649606428	0	0
16	3	3	-49.8079118230134	-35.0159649606428	0	0
17	3	4	-56.6452355851319	11.5669854701223	0	0
18	3	5	-56.6452355851319	-11.5669854701223	0	0
19	4	0	-0.000209439510239319	0	0	0
20	5	0	-33.5462372106204	59.9028482625725	0	0
21	5	1	-33.5462372106204	-59.9028482625725	0	0
22	5	2	-49.8079118230134	35.0159649606428	0	0
23	5	3	-49.8079118230134	-35.0159649606428	0	0
24	5	4	-56.6452355851319	11.5669854701223	0	0
25	5	5	-56.6452355851319	-11.5669854701223	0	0
26	6	0	-0.000209439510239319	0	0	0
27	7	0	-33.5462372106204	59.9028482625725	0	0
28	7	1	-33.5462372106204	-59.9028482625725	0	0
29	7	2	-49.8079118230134	35.0159649606428	0	0
30	7	3	-49.8079118230134	-35.0159649606428	0	0
31	7	4	-56.6452355851319	11.5669854701223	0	0
32	7	5	-56.6452355851319	-11.5669854701223	0	0

Besides low and high pass analogues filter, digital anti-alias 3-stage Chebyshev FIR filters are applied to the LIMS data on all 5 channels. The digital filtration schemes process a high sample rate data stream; filter; then decimate, to produce the desired output.

Implementation of Digital Filter (d) 24 hz -> 0.2 hz

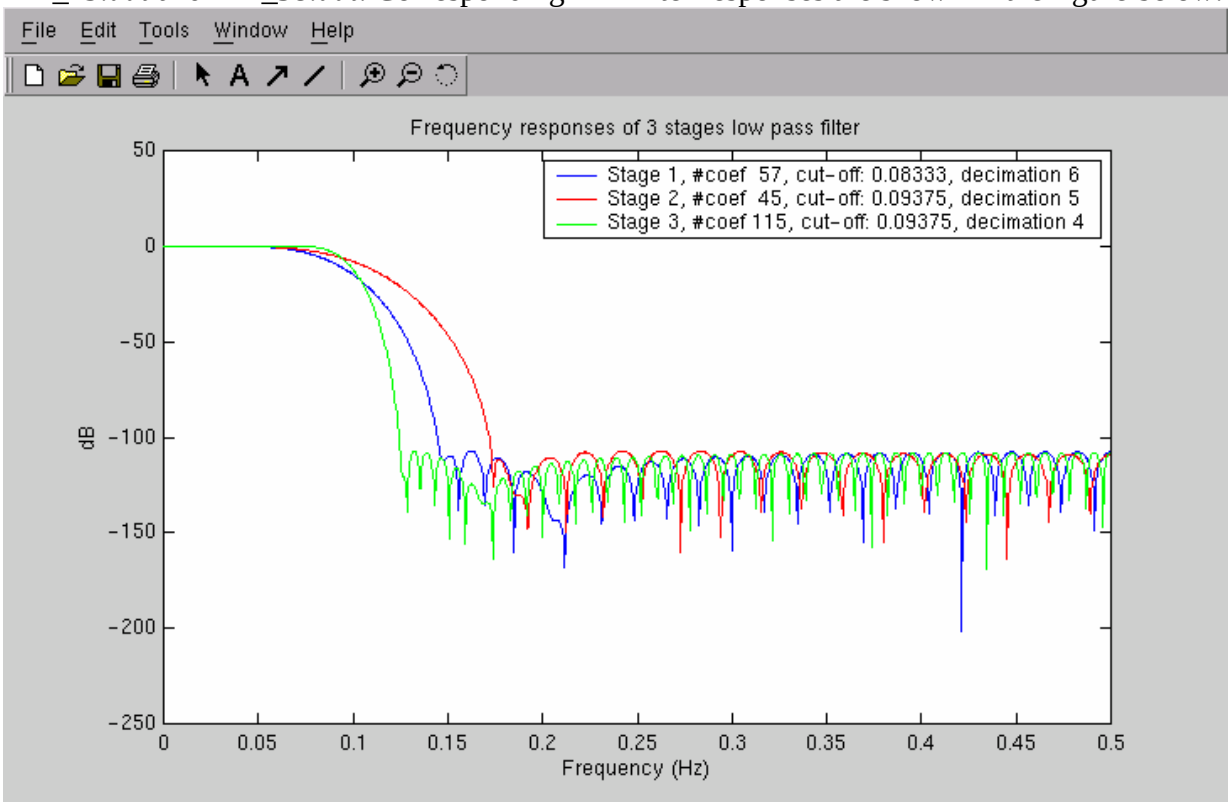
Stage	Decimation Factor	# filter coeff.	Downcounter reset value	ring buffer delay time
1	6	57	6	5 seconds (120 samples @ 24 hz)
2	5	45	30	10 seconds (40 samples @ 4 hz)
3	4	115	24	90 seconds (72 samples @ .8 hz)

Considering high/low pass Bessel filters already described above, for magnetic channels the FIR filter stages are 2, 3, 4, and for magnetic channels - 3, 4, 5. Blockette 61 is used to specify FIR digital filter coefficients. We assume type B symmetry (odd number of filter coefficients) for all FIR stages. Note, that the last column in the fir_symmetry_61 table is called `number_coeff`, when actually it shows number of factors (see SEED manual). For the B symmetry type number of factors is $N_{fac}=(N_{coef}+1)/2$.

mysql> select * from fir_symmetry_61;

id	chan_id	proto_id	stage_ number	response_ name	Sym metry_ code	Signal _in_ units_ refid	Signal _out_ units_ refid	Number _coeff
1	1	NULL	2	NULL	B	4	4	29
2	1	NULL	3	NULL	B	4	4	23
3	1	NULL	4	NULL	B	4	4	58
4	2	NULL	2	NULL	B	4	4	29
5	2	NULL	3	NULL	B	4	4	23
6	2	NULL	4	NULL	B	4	4	58
7	3	NULL	2	NULL	B	4	4	29
8	3	NULL	3	NULL	B	4	4	23
9	3	NULL	4	NULL	B	4	4	58
10	4	NULL	2	NULL	B	4	4	29
11	4	NULL	3	NULL	B	4	4	23
12	4	NULL	4	NULL	B	4	4	58
13	5	NULL	2	NULL	B	4	4	29
14	5	NULL	3	NULL	B	4	4	23
15	5	NULL	4	NULL	B	4	4	58

The FIR filter coefficients are calculated only once with a *matlab* script *FIRf.m* (based on Appendix C of MT LUNCHBOX) in the **LimsNims/PerlL** directory and saved in ASCII files *FIR_29.dat*, *FIR_23.dat* and *FIR_58.dat*. Corresponding FIR filter responses are shown in the figure below:



Fir coefficients files *FIR*.dat* are read by script **LimsNims/PerlL/***import_blkt61.pl* (called by the control scripts *arc_lims*) and these coefficients then placed in fir_coeff_61 table. The column ref_id in this table refers to the column id in the fir_symmetry_61 table. Only first 29 coefficients for ref_id=1 are shown in the table below.

mysql> select * from fir_coeff_61 where ref_id=1;

id	ref_id	seq	coefficient
1	1	0	8.334796e-06
2	1	1	2.208931e-05
3	1	2	4.066573e-05
4	1	3	4.465071e-05
5	1	4	-7.91818e-08
6	1	5	-0.0001311461
7	1	6	-0.0003608211
8	1	7	-0.0006376099
9	1	8	-0.0008187073
10	1	9	-0.000681857
11	1	10	7.250504e-07
12	1	11	0.001328088
13	1	12	0.003115611
14	1	13	0.00479482
15	1	14	0.005455172
16	1	15	0.004085216
17	1	16	-2.636826e-06
18	1	17	-0.006675313
19	1	18	-0.01460059
20	1	19	-0.02119498
21	1	20	-0.02303564
22	1	21	-0.01672284
23	1	22	5.3251e-06
24	1	23	0.02725022
25	1	24	0.06240374
26	1	25	0.1003492
27	1	26	0.1344599
28	1	27	0.1581684
29	1	28	0.1666601

Filter blocketts 53 and 61 content corresponds to the content of two other blocketts, describing decimation and sensitivity/gain.

Decimated and non-decimated filter stages are both included in the blockette 57 tables, as recommended by SEED manual. For non-decimated stages the decimation factor is 1, and the offset value is 0.

mysql> select * from decimation_57

id	chan_id	proto_id	stage_ number	input_ sample_rate	Decimation _factor	Decimation _offset	est_ delay	corr_ applied
1	1	NULL	1	24	1	0	0	0
2	1	NULL	2	24	6	0	5	-5
3	1	NULL	3	4	5	0	10	-10
4	1	NULL	4	0.8	4	0	90	-90
5	2	NULL	1	24	1	0	0	0
6	2	NULL	2	24	6	0	5	-5
7	2	NULL	3	4	5	0	10	-10
8	2	NULL	4	0.8	4	0	90	-90

9	3	NULL	1	24	1	0	0	0
10	3	NULL	2	24	6	0	5	-5
11	3	NULL	3	4	5	0	10	-10
12	3	NULL	4	0.8	4	0	90	-90
13	4	NULL	1	24	1	0	0	0
14	4	NULL	2	24	1	0	0	0
15	4	NULL	3	24	6	0	5	-5
16	4	NULL	4	4	5	0	10	-10
17	4	NULL	5	0.8	4	0	90	-90
18	5	NULL	1	24	1	0	0	0
19	5	NULL	2	24	1	0	0	0
20	5	NULL	3	24	6	0	5	-5
21	5	NULL	4	4	5	0	10	-10
22	5	NULL	5	0.8	4	0	90	-90

Blockette 58 is used both as a gain at a given frequency (stage>0) and as sensitivity (stage=0) for the entire channel at given frequency. The **sensitivity_gain_58** table is filled out simultaneously with the **decimation_57** table. Number of rows in **sensitivity_gain_58** table is equal to the number of rows in **decimation_57** table plus number of channels (sensitivity specification for each channel).

Gain for stage 1 for magnetic channels (low-pass) is calculated as $1/h_{fac}*1e9$, where h_{fac} is taken from *sp file, at zero frequency (conversion from Tesla to digital counts). For example, if h_{fac}=3.05e-2, the gain will be 1/3.05e-2*1e9=3.2787e10, as shown in the table below. FIR stage gains are units for the magnetic channels, and sensitivity is a product of gains over entire channel at zero frequency, that is also $1/h_{fac}*10^9$.

Gain for stage 1 for electric channels is equal to the dipole length (m), coming from *sp files (100m in the example below) and corrected for normalization frequency 0.01 (i.e. multiplied by A_0/A_1, where A_1 calculated at frequency=0.01Hz). Gain for stage 2 for magnetic channels (low-pass) is calculated as $1/e_{fac}*10^3$, where e_{fac} is taken from *sp file and multiplied by A_0/A_1, where A_1 calculated at frequency=0.01Hz (conversion from Volts to digital counts). FIR stage gains are units for the electric channels, and sensitivity is a product of gains over entire channel at 0.01Hz frequency.

mysql> select * from sensitivity_gain_58;

id	chan_id	proto_id	stage_number	sensitivity_gain	frequency	number_history
1	1	NULL	1	32786884923.4077	0	0
2	1	NULL	2	1	0	0
3	1	NULL	3	1	0	0
4	1	NULL	4	1	0	0
5	1	NULL	0	32786884923.4077	0	0
6	2	NULL	1	32786884923.4077	0	0
7	2	NULL	2	1	0	0
8	2	NULL	3	1	0	0
9	2	NULL	4	1	0	0
10	2	NULL	0	32786884923.4077	0	0
11	3	NULL	1	32786884923.4077	0	0
12	3	NULL	2	1	0	0
13	3	NULL	3	1	0	0
14	3	NULL	4	1	0	0
15	3	NULL	0	32786884923.4077	0	0
16	4	NULL	1	117.999344449908	0.01	0
17	4	NULL	2	655737.698468154	0.01	0
18	4	NULL	3	1	0.01	0
19	4	NULL	4	1	0.01	0
20	4	NULL	5	1	0.01	0
21	4	NULL	0	77376618.5503335	0.01	0
22	5	NULL	1	139.999222228704	0.01	0
23	5	NULL	2	655737.698468154	0.01	0

24	5	NULL	3	1	0.01	0
25	5	NULL	4	1	0.01	0
26	5	NULL	5	1	0.01	0
27	5	NULL	0	91802767.7715821	0.01	0

NIMS Filter/Decimation blocketts

Analogue 3 poles Butterworth low pass filter is applied for NIMS magnetic channels. Electric channels are filtered with 1 pole Butterworth high pass filter on stage 1 and 5 poles Butterworth low pass filter on stage 2. Parameters for these filters (T0 and delays) are coming from the **.hed* files produced by *nimsreadz*.

mysql>select * from poles_zeroes_53;

id	Chan_id	Proto_id	Transfer_ fn_type	stage_ number	signal_in_ units_refid	signal_out_ units_refid	Normalization _factor	Normalization _freq	number_ complex_ zeroes	number_ complex_ poles
1	1	0	A	1	1	4	1984.31439386405	0	0	3
2	2	0	A	1	1	4	1984.31439386405	0	0	3
3	3	0	A	1	1	4	1984.31439386405	0	0	3
4	4	0	A	1	2	3	1.00000351811134	0.01	1	1
5	4	0	A	2	3	4	313383.601119191	0.01	0	5
6	5	0	A	1	2	3	1.00000351811134	0.01	1	1
7	5	0	A	2	3	4	313383.601119191	0.01	0	5
8	6	0	A	1	1	4	1984.31439386405	0	0	3
9	7	0	A	1	1	4	1984.31439386405	0	0	3
10	8	0	A	1	1	4	1984.31439386405	0	0	3
11	9	0	A	1	2	3	1.00000351811134	0.01	1	1
12	9	0	A	2	3	4	313383.601119191	0.01	0	5
13	10	0	A	1	2	3	1.00000351811134	0.01	1	1
14	10	0	A	2	3	4	313383.601119191	0.01	0	5

Exactly like for LIMS case magnetic channels normalization frequency is taken zero, and for electric channels normalization frequency is set to $f_n=0.01$; then normalization factors are calculated from complex poles and zeroes using the same formulas, i.e.:

$$A_1 = \left| \frac{\prod_j (2\pi f_n - p_j)}{\prod_k (2\pi f_n - z_k)} \right|, \quad A_0 = \left| \prod_j p_j \right| \quad (no\ zeros,\ \ f_n = 0)$$

The poles and zeroes values for T0=0.5 (low-pass) and T0=37699 (high-pass) are shown in the tables below ("*ref_id*" column corresponds to the "*id*" column in the table *poles_zeroes_53*). For example for *chan_id=1*, *ref_id=1*, normalization factor for 3 poles Butterworth low pass filter is calculated as (here precision is limited to 2 digits unlike in the table above):

(-6.28+i10.88)*(-6.28-i10.88)*(-12.57)=1.984e3

For high pass filter (i.e. for *ref_id=4*) we have 1 pole, calculated as -2π/T0=-2π/37699=-1.67e-4, and normalization factor is calculated as $A_1 = \frac{(2\pi \cdot 0.01 + 1.67e-4)}{2\pi \cdot 0.01} = 1.0000004$

mysql>select * from complex_zeroes_53;

id	ref_id	seq	real_val	imaginary_val	real_error	imaginary_error
1	4	0	0	0	0	0
2	6	0	0	0	0	0
3	11	0	0	0	0	0
4	13	0	0	0	0	0

mysql>select * from complex_poles_53;

id	ref_id	seq	real_val	imaginary_val	real_error	imaginary_error
1	1	0	-6.28318530717958	10.882476952035	0	0
2	1	1	-6.28318530717958	-10.882476952035	0	0
3	1	2	-12.5663706143592	0	0	0
4	2	0	-6.28318530717958	10.882476952035	0	0
5	2	1	-6.28318530717958	-10.882476952035	0	0
6	2	2	-12.5663706143592	0	0	0
7	3	0	-6.28318530717958	10.882476952035	0	0
8	3	1	-6.28318530717958	-10.882476952035	0	0
9	3	2	-12.5663706143592	0	0	0
10	4	0	-0.000166667161123096	0	0	0
11	5	0	-3.88300851983698	11.951875091317	0	0
12	5	1	-3.88300851983698	-11.951875091317	0	0
13	5	2	-10.1661938270166	7.38651264712031	0	0
14	5	3	-10.1661938270166	-7.38651264712031	0	0
15	5	4	-12.5663706143592	0	0	0
16	6	0	-0.000166667161123096	0	0	0
17	7	0	-3.88300851983698	11.951875091317	0	0
18	7	1	-3.88300851983698	-11.951875091317	0	0
19	7	2	-10.1661938270166	7.38651264712031	0	0
20	7	3	-10.1661938270166	-7.38651264712031	0	0
21	7	4	-12.5663706143592	0	0	0
22	8	0	-6.28318530717958	10.882476952035	0	0
23	8	1	-6.28318530717958	-10.882476952035	0	0
24	8	2	-12.5663706143592	0	0	0
25	9	0	-6.28318530717958	10.882476952035	0	0
26	9	1	-6.28318530717958	-10.882476952035	0	0
27	9	2	-12.5663706143592	0	0	0
28	10	0	-6.28318530717958	10.882476952035	0	0
29	10	1	-6.28318530717958	-10.882476952035	0	0
30	10	2	-12.5663706143592	0	0	0
31	11	0	-0.000166667161123096	0	0	0
32	12	0	-3.88300851983698	11.951875091317	0	0
33	12	1	-3.88300851983698	-11.951875091317	0	0
34	12	2	-10.1661938270166	7.38651264712031	0	0
35	12	3	-10.1661938270166	-7.38651264712031	0	0
36	12	4	-12.5663706143592	0	0	0
37	13	0	-0.000166667161123096	0	0	0
38	14	0	-3.88300851983698	11.951875091317	0	0
39	14	1	-3.88300851983698	-11.951875091317	0	0
40	14	2	-10.1661938270166	7.38651264712031	0	0
41	14	3	-10.1661938270166	-7.38651264712031	0	0
42	14	4	-12.5663706143592	0	0	0

Since no FIR filters is used for NIMS, blockette 61 is not filled (unlike LIMS case).

Blockette 57 includes only non-decimated stages (thus decimation factor is unit, and decimation offset is zero for all channels). Typical part of the **.hed* file produced by *nimsreadz* for MT1 NIMS data used to fill out the *decimation_57* table is shown below:

```
# Magnetic field - 3 pole Butterworth LOWPASS
#  corner PERIOD    GROUP DELAY (sec):
#   Hx     0.5         0.159
#   Hy     0.5         0.159
#   Hz     0.5         0.159
# Electric field - 5 pole Butterworth LOWPASS
#  corner PERIOD    GROUP DELAY (sec):
#   Ex     0.5         0.2575
#   Ey     0.5         0.2575
# Electric field - 1 pole Butterworth HIGHPASS
#  corner PERIOD    TIME CONSTANT (sec):
#   Ex     37699       6000
#   Ey     37699       6000
#
# MT sample time offsets:
#    (>0 implies sample time earlier than clock time = FAST clock)
#          Absolute   Relative to Hx (including group delays)
#    Hx    0.2455       0.0000 seconds
#    Hy    0.2365      -0.0090 seconds
#    Hz    0.2275      -0.0180 seconds
#    Ex    0.1525      -0.0055 seconds
#    Ey    0.1525      -0.0055 seconds
```

For MT1 example shown above, absolute time offsets are taken as estimated delays in the table below.

mysql> select * from decimation_57;

id	chan_id	proto_id	stage_number	input_sample_rate	decimation_factor	decimation_offset	est_delay	corr_applied
1	1	NULL	1	8	1	0	0.2455	0
2	2	NULL	1	8	1	0	0.2365	0
3	3	NULL	1	8	1	0	0.2275	0
4	4	NULL	1	8	1	0	0	0
5	4	NULL	2	8	1	0	0.1525	0
6	5	NULL	1	8	1	0	0	0
7	5	NULL	2	8	1	0	0.1525	0
8	6	NULL	1	8	1	0	0.2455	0
9	7	NULL	1	8	1	0	0.2365	0
10	8	NULL	1	8	1	0	0.2275	0
11	9	NULL	1	8	1	0	0	0
12	9	NULL	2	8	1	0	0.1525	0
13	10	NULL	1	8	1	0	0	0
14	10	NULL	2	8	1	0	0.1525	0

Since HP200 NIMS **.hed* files are different and contain only delays realitive to Ex, the estimated delay values for these data are calculated as: -<*group_delay*>- <*relative_to_Ex_delay*>. For example, typical NIMS HP200 *.hed file part looks like:

```
# Magnetic field - 3 pole Butterworth LOWPASS
#  corner PERIOD    GROUP DELAY (sec):
#   Hx     0.5         0.159
#   Hy     0.5         0.159
#   Hz     0.5         0.159
```

```
# Electric field - 5 pole Butterworth LOWPASS
#  corner PERIOD    GROUP DELAY (sec):
#   Ex     0.5         0.2575
#   Ey     0.5         0.2575
# Electric field - 1 pole Butterworth HIGHPASS
#  corner PERIOD    TIME CONSTANT (sec):
#   Ex    188496      30000
#   Ey    188496      30000
#
# MT sample time offsets:
#    (>0 implies sample time earlier than clock time = FAST clock)
#       Relative to Ex (including group delays)
#    Hx  +0.2555 seconds
#    Hy  +0.2615 seconds
#    Hz  +0.2735 seconds
#    Ex  +0.0000 seconds
#    Ey  -0.1250 seconds
```

Then estimated delays for Hx, Hy, Hz, Ex, Ey will be -0.4145,-0.4245,-0.4325,-0.2575,-0.1325 correspondingly.

Like for LIMS case, the *sensitivity_gain_58* table is filled out simultaneously with the *decimation_57* table. Number of rows in *sensitivity_gain_58* table is equal to the number of rows in *decimation_57* table plus number of channels (sensitivity specification for each channel).

Gain for stage 1 for magnetic channels (Butterworth 3 poles low-pass) is calculated as $1/h_{fac}*1e9$, where h_{fac} is taken from **.hed* file, at zero frequency (conversion from Tesla to digital counts). Typical value of h_{fac} is 0.01, then gain will be 1e11, as shown in the table below. Sensitivity is a product of gains over entire channel at zero frequency, that is also $1/h_{fac}*10^9$.

Gain for stage 1 for electric channels is equal to the dipole length (m), coming from *.hed files (100m in the example below) and corrected for normalization frequency 0.01 (i.e. multiplied by A_0/A_1, where A_1 calculated at frequency=0.01Hz). Gain for stage 2 for magnetic channels (low-pass) is calculated as $1/e_{fac}*10^3$, where e_{fac} is taken from *.hed file, and multiplied by A_0/A_1, where A_1 calculated at frequency=0.01Hz (conversion from Volts to digital counts). For example, typical value for MT1 NIMS e_{fac} is 2.441406e-6, thus typical value for gain will be 4.0960e+08, multiplied by A_0/A_1 (~1). Sensitivity is again calculated as a product of gains over entire channel at 0.01Hz frequency.

mysql>select * from sensitivity_gain_58;

id	chan_id	proto_id	stage_number	sensitivity_gain	frequency	number_history
1	1	NULL	1	100000000000	0	0
2	1	NULL	0	100000000000	0	0
3	2	NULL	1	100000000000	0	0
4	2	NULL	0	100000000000	0	0
5	3	NULL	1	100000000000	0	0
6	3	NULL	0	100000000000	0	0
7	4	NULL	1	99.9996481901037	0.01	0
8	4	NULL	2	409600042.095954	0.01	0
9	4	NULL	0	40959860108.247	0.01	0
10	5	NULL	1	99.9996481901037	0.01	0
11	5	NULL	2	409600042.095954	0.01	0
12	5	NULL	0	40959860108.247	0.01	0
13	6	NULL	1	100000000000	0	0
14	6	NULL	0	100000000000	0	0
15	7	NULL	1	100000000000	0	0

16	7	NULL	0	100000000000	0	0
17	8	NULL	1	100000000000	0	0
18	8	NULL	0	100000000000	0	0
19	9	NULL	1	99.9996481901037	0.01	0
20	9	NULL	2	409600042.095954	0.01	0
21	9	NULL	0	40959860108.247	0.01	0
22	10	NULL	1	99.9996481901037	0.01	0
23	10	NULL	2	409600042.095954	0.01	0
24	10	NULL	0	40959860108.247	0.01	0