Processing of Seismic Events Collected by the New Mexico Tech Seismic Network Using a Matseis-based Review Tool and a Postgres Database

by
Xiaobing Zhang

Submitted in Partial Fulfillment of the Requirements for the Degree of Master of Science in Geophysics from the Department of Earth and Enviromental Science, New Mexico Institute of Mining and Technology

August 10, 1999

Abstract

A seismic event analysis tool based on the interface of MatSeis and a Postgres database is recently created. Storing the earthquake events collected by New Mexico Tech network in Postgres databases is also a new approachment. These works aim at improving the previous event analysis system which uses flatfile database to store the events and DBPick to analyze.

Experiment with the new system shows that it is usually capable of fulfilling our objective of reviewing earthquake events: reading data from a Postgres database, relocating the event and then writing the reclassified event into a reviewed Postgres database. Reviewed events can be reviewed again.

Matseis- NMT, the development of MatSeis, is believed to be a fairly useful and complete seismic event analysis tool.

Acknowledgment

Thanks go to my advisor Dr. Richard Aster for his direct help in every step of this project. Mark Harris and Christopher Young in Sandia National Laboratories provided invaluable assistance and suggestion in my understanding of the MatSeis software and therefore interfacing of MatSeis with Postgres. Thanks to Mitch Withers at Center for Earthquake Research and Information for his patient help in understanding LWCEDS project and his work of the scripts 'lwcedsauto' and 'lwcedsproc' which are the base of my new routine.

I thank Dr. Allan Sanford for his help in my analyzing events with Seismos and his seminars on earthquake identification on analog seismographs have been very helpful. Dr. John Schlue is proved valuable addition to my advisory committee

I appreciate the generous help from my fellow graduate students throughout this project, especially Charlotte Rowe who provided very useful advice in many problems and Kuowan Lin who helped a lot with the Socorro and New Mexico seismicity.

Finally, I wish to thank my parents for their support and encouragement that is always there.

This project is supported by the Sandia National Laboratories through contract No. 29P-9RA010-502.

Table of Contents

1. Introduction: Motivation/Brief History of NMT Network Monitoring

The state of New Mexico is seismically active because of the existence of the Socorro magma body (SMB) and the frequent mining activities. Seismicity monitoring has provided rich information since 1982 when the network was installed. Digital earthquake data are currently processed using **LWCEDS** method (*Withers,* 1997; *Withers et al.,* 1998; *Withers et al.,* 1999*)* and were reviewed using **JSPC-DBPick** (*Harvey* et al.,1994).

Retrospect: a brief history of NMT network monitoring

The New Mexico Tech (NMT) network was installed between August 1982 and February 1983 and the datalogger was added in 1991 (*Skov,*1994). Currently this network has 17 stations (Figure 1), including 11 short-period, vertical-component, analog-telemetry stations, digitized to 100 Hz; 1 short-period, 3-component, analog-telemetry station, digitized to 100 Hz; 5 short-period, vertical-component, analog-telemetry stations, not digitized (*Aster*, personal communication).

The digital data acquisition system was installed in 1991 and improvement and development have been carried out thereafter. At first NMT network used a digital seismic data acquisition and processing system which was originally applied for collection of data from both Rio Grande Rift and Mount Erebus, Antarctica (*Skov*, 1994). This system used **XDETECT** to detect earthquake events and **XPICK** and **Seismos** (*Hartse*, 1991) to determine the location of events. **XDETECT**, the principal data acquisition software, uses the ratio of the first differences of short term average (**STA**) and long term average (**LTA**) of the signal as the criteria to detect earthquake events

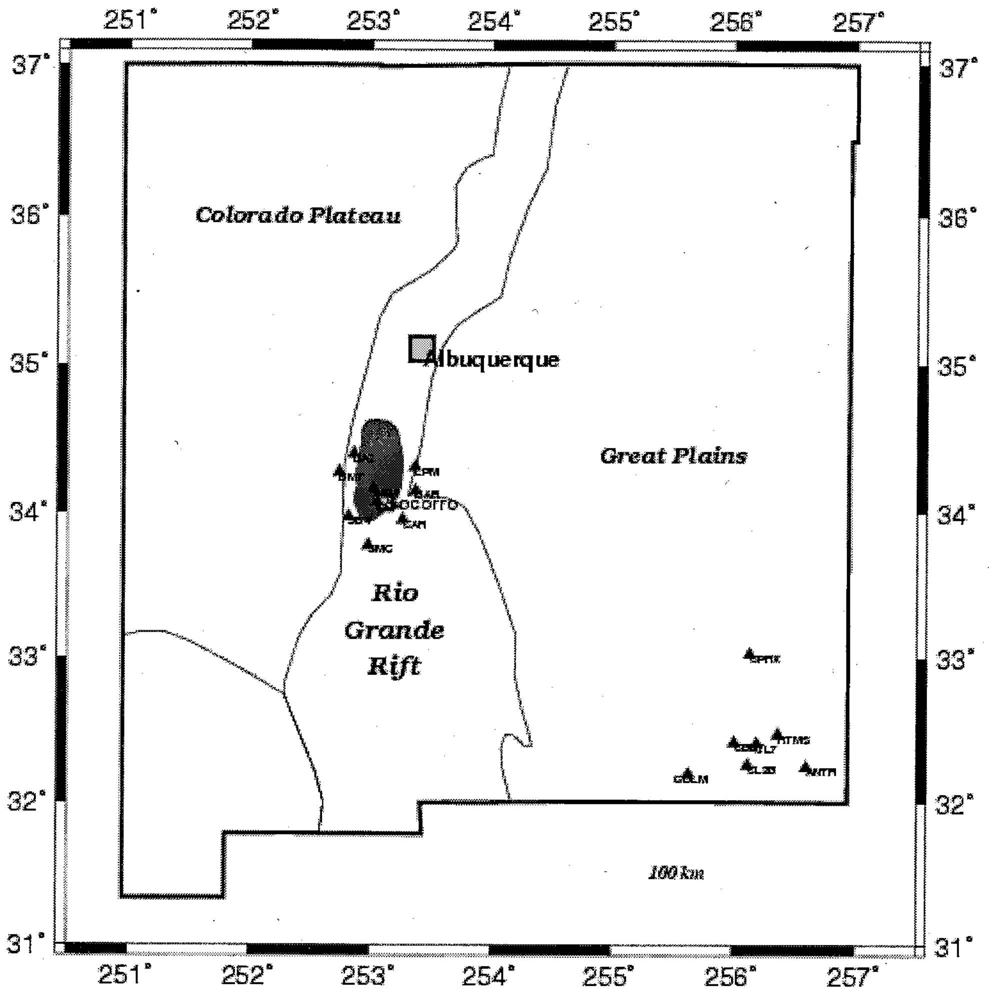


Fig. 1. Station map of New Mexico Tech network: Socorro stations and teleseismic stations. Geographical data are the same as *Hartse*[1991].

(*Skov*, 1994). Data acquired were temporarily stored in a PC datalogger and then transferred to a SUN work station for an analyst to review. E-mail messages were sent automatically to those who were concerned when earthquakes were detected.

Seismic data processing using the method of ***LWCEDS***

Improvement has been made recently on event detection and location algorithms for the NMT network. NMT network is using the **LWCEDS** method to monitor seismicity of New Mexico and adjacent states (Utah, Texas, Arizona and Colorado). The current version of **LWCEDS**, **LWCEDS-1.1**, is capable of monitoring and alerting operators to earthquakes in near-real-time.

The earthquake events that NMT network collects are classified into the following types: *local*, mainly reflected microearthquakes centered near willard, New Mexico from the local Socrro magma body (Figure 2);. *regional*, events that occur in the region of New Mexico out to a range of several hundred kilometers; *teleseisms*, events that occur at ranges greater than ~5 degrees.

LWCEDS, short for 'local waveform correlation event detection system', uses an adaptive **STA/LTA** (*Allen*, 1978; *Allen*, 1982) algorithm to detect, report and record an earthquake event and also gives coarse location, magnitude and time for the detected event based on processed waveform correlation (*Withers,* 1997; *Withers et al.,* 1998; *Withers et al.,* 1999). The information of the event is stored in CSS-3.0 (Center for Seismic Studies Version 3, *Anderson et al.*, 1990) flatfile tables.

Seismic data analysis using ***DBPick***

The seismic data analysis tool in New Mexico Tech was formerly **JSPC DBPick** and associated software (*Harvey et al.*,1994). This tool made use of the CSS-3.0 format

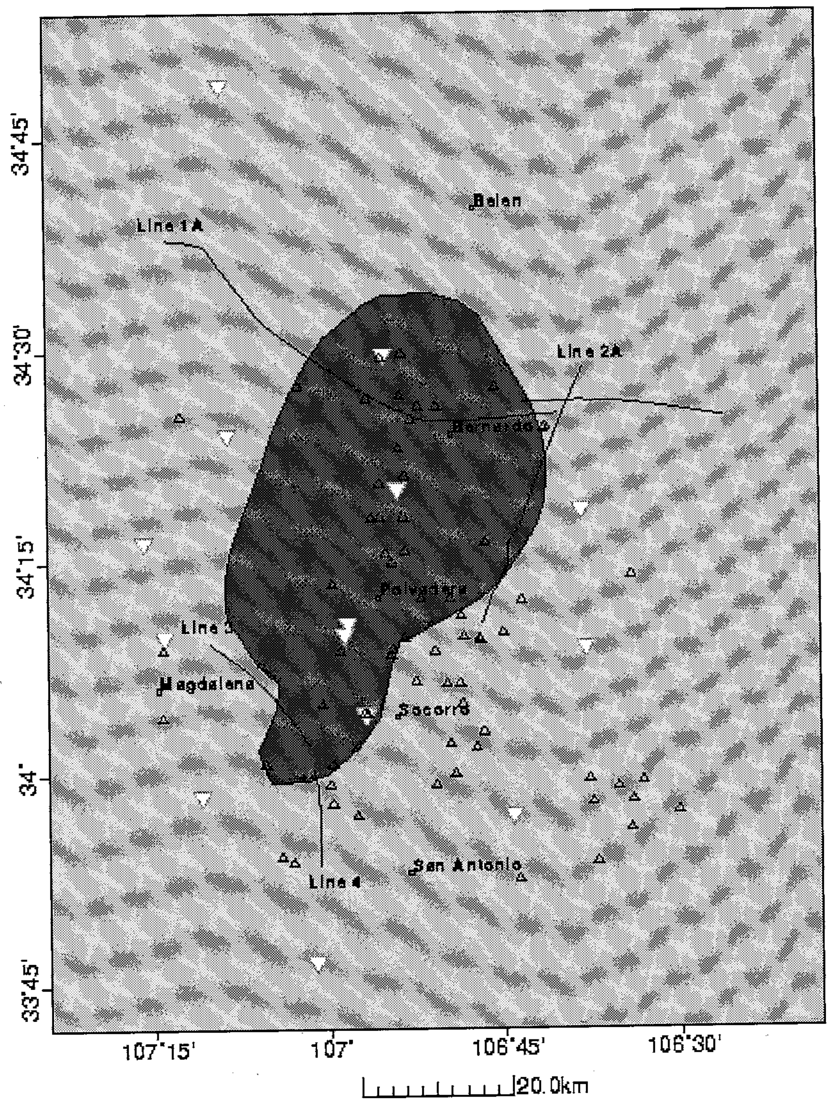


Fig.2. Magma sill map showing the geographic extent of Socorro magma body (SMB) (from *Balch et al.*, 1997).

flatfile database. **DBPick** is a fairly simple review tool which doesn't have the functionality of data processing. It reads data from flatfile tables and concatenates the modified tables (the result of analysis) to the output data directory therefore makes huge flatfile tables in that directory. **Seismos** was formerly run as a customized DBPick feature to relocate local events.

DBPick reads data from only flatfile database and writes to flatfile database, which makes it a limited tool for the common cases where data are stored in traditional relational databases (such as Oracle and Postgres database). The other disadvantage of **DBPick** is that **Seismos** as the location tool of **DBPick** is proved vulnerable while locating regional events as it doesn't always converge in the process of iteration for events with large azimuthal gaps (*Aster*, personal communication).

The above shortcomings of **DBPick** form part of the reasons that we want to use a totally different seismic event analysis tool-**MatSeis** (*Harris et al.*, 1997), and make modifications for our use.

Motivation of this project

The success of the data acquisition system and **LWCEDS** automatic location was most recently proved by locating the large main shocks which occurred on December 31, 1997 and January 4, 1998 with magnitude M_l=3.0 and M_l=3.8 respectively, centered near Willard, New Mexico.

The task of this project is to interface **LWCEDS** with **MatSeis**, a seismic event analysis and data processing tool created by Sandia National Laboratories, to create the version of **MatSeis** in New Mexico Tech, **Matseis-NMT**.

We wish to use **MatSeis** as the earthquake event review tool because based on open code which interfaces with the versatile Matlab (*The Mathworks*, 1992) package, **MatSeis** provides a large library of signal processing routines and supplies a convenient algorithm development environment (*Harris et al.*, 1997), which makes it easier to interface with a Postgres database than otherwise. Besides, the relocation tool '**LocSAT**' (*Bratt and Bache*, 1998) in **MatSeis** is more robust in relocating regional events than **Seismos**. At this point, **DBPick** and associated software appear to be obsolete as its development and support has ended.

The desire to use **MatSeis** instead of **DBPick** to review events and the fact that **MatSeis** is very slow in reading from the flatfile database initiated the development of a **MatSeis** interface with the Postgres database. Another advantages of using database is that it is easier to manipulate and administrate than flat files.

Experiments with **Matseis-NMT** show that it is generally able to fulfill our goal of monitoring NM seismicity and give reliable location and magnitude estimation.

2.Previous Work

The project of **Matseis-NMT** development is based on the data processing system of New Mexico Tech network, the previous review tool-**DBPick**, understanding of how Postgres database works and study of the **MatSeis** package.

2.1 Data Processing System of NMT Network

Introduction of STA/LTA algorithm and its application in LWCEDS

The **STA/LTA** algorithm is the signal processing algorithm used by **LWCEDS** to enhance seismic phase arrivals (*Withers,* 1997; *Withers et al.,* 1998; *Withers et al.,* 1999).

This algorithm is a common method used in automatic earthquake detection (*Allen*,1978). 'STA' stands for the short term average and 'LTA' stands for long term average of some characteristic functions, such as absolute value, square, first derivative, of the signal. To detect an earthquake, when 'STA' is greater than some multiples of 'LTA' and some other logical conditions are satisfied an event is declared. Depending on the feature of the signal and the characteristics of the signal that need to be highlighted, such as frequency variation or amplitude changes, **STA/LTA** algorithm can be operated on many kinds of characteristic functions of the signal (*Allen*, 1982).

To avoid prohibitively complex models in correlation, **LWCEDS** requires a processed envelope which has smooth, low amplitude response to background noise and seismic coda, and provides peaks at a maximum number of phase arrivals (*Withers,* 1997; *Withers et al.,* 1998; *Withers et al.,* 1999). **STA/LTA** is performed on seismic signal for this objective because it smooths out the effect of the low frequency components in the signal and thus amplifies the contrast between the signal and the background noise which makes the peak arrivals significant.

In **LWCEDS**, the **STA/LTA** processed envelope correlates with theoretical or empirical travel time curves (Figure 3). High correlation indicates that signal is present in the data and that traces are properly aligned in both space and time. Poor correlation indicates misalignment of traces in space and time or lack of an event (*Withers*, 1997). The basic idea of LWCEDS method is shown in Figure 4.

***Seismos**-Event Relocation Tool*

Seismos, a software package with a group of Fortran programs to invert for event locations, was used to perform a fine relocation for local earthquakes. It is used as the event fine location tool after **LWCEDS** coarsely locates the events.

Once an event is detected within the range of a local fine grid, a local velocity model is then used to perform a fine search. **Seismos** was designed by Hanse Hartse [1991] to calculate the location of microearthquakes, mainly reflections from the Socorro magma body, New Mexico. **Seismos** uses the P, S, SzS, PzP and SzP arrivals and the local velocity model generated by the ray tracing method to invert for the location of local earthquakes. It is an improvement over traditional generalized least square inverse method and emphasizes the focal depth constraint. Reflection phases PzP, SzS and SzP are used instead of just P and S phases. The method is designed for the NMT network because the station spacing is large relative to earthquake depth which makes it difficult to constrain focal depth using only direct P and S arrivals (*Hartse*, 1991).

Seismos successfully calculated the location of local seismic events. However relocation of regional events using **Seismos** is not robust because the iteration process doesn't converge well.

Improvement in relocation of regional events in this project is achieved by using

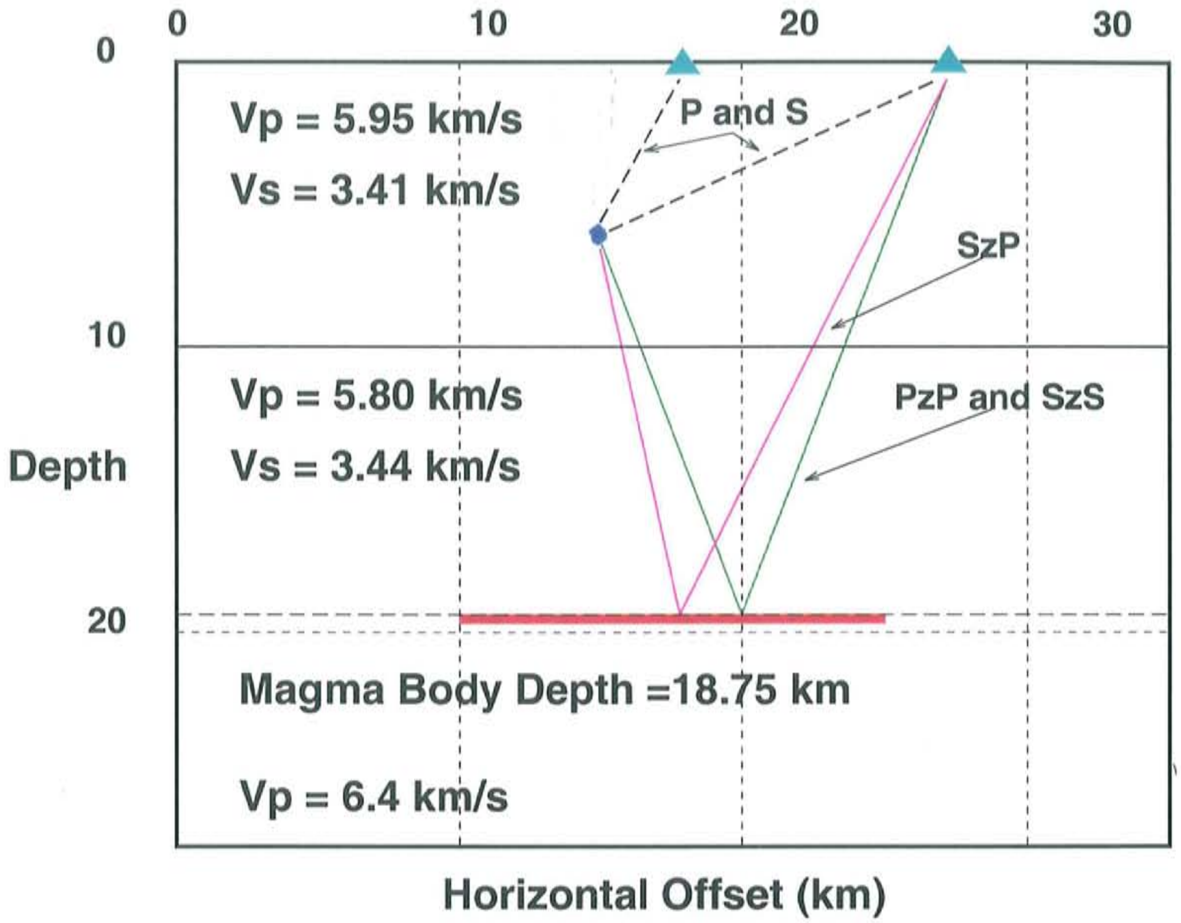


Fig. 3. Velocity model used by both Seismos and LocSAT in this study. Reflected phases PzP, SzS and SzP from the Socorro Magma Body are observed (*Withers*, 1997)

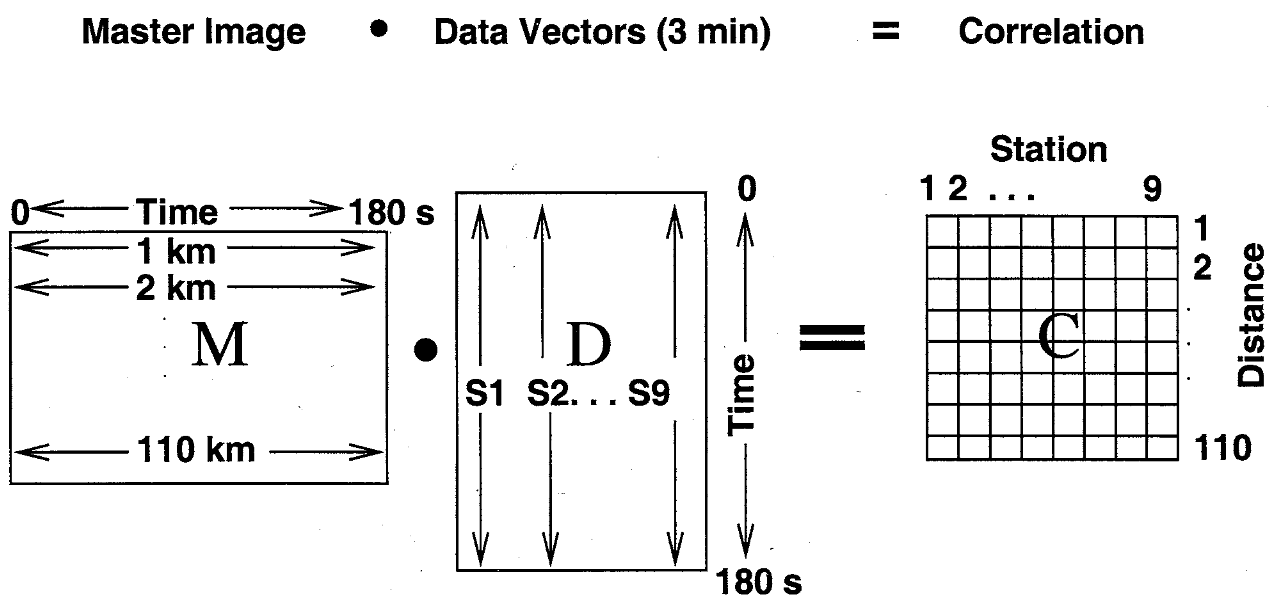


Figure 5

Fig. 4. Basic idea of the method of LWCEDS. The model matrix correlates with the data matrix to generate the correlation matrix. Each element of the correlation matrix contains a value representative of how well the data correlate with the model for each station at each distance. A grid is imposed on the monitoring region and a correlation value for each grid point is determined by summing a single contribution from each station with consideration of the travel distance between the grid point and the station. The grid point with the maximum sum is stored as the hypocenter if this sum is above a detection threshold (*Withers*, 1995).

LocSAT, a software used by permission of Walter Nagy, SAIC, algorithm described in *Bratt and Bache* (1988). **LocSAT** is also proved an effective tool in locating local events therefore it is used in our new system as the location tool.

2.2 DBPICK- The Review Tool

DBPick was used by the NMT network as the review tool to analyze earthquake events. The following jobs are performed: waveform review, picking arrivals, and editing a CSS relational database.

DBPick is an X-window based interactive graphical program for displaying seismic waveforms, analyst picked phase arrivals and temporary predicted arrivals. **DBPick** also provides functionality for creating and/or editing phase arrivals. The data must be presented in a CSS relational database implemented as a set of ASCII flat files (e.g., datascope), each file corresponding to one of the CSS relations (tables).

DBPick allows relocation of an event by calling **Seismos**. **Seismos** relocates an event once this event falls in the range of a Socorro seismic anomaly area. In **DBPick**, **Seismos** is performed when clicking on the '**seismos**' button.

DBPick is called by a C program **review.c** which allows analyst to review earthquake events. It accesses an unreviewed database, writes to two temporary databases, calls **DBPick**, and then updates the reviewed database. '**Review**' is called by the script '**lwcedsproc**' which is run to process events. After an event is reviewed, the flatfile tables of the event are appended respectively to the reviewed tables under the directory where the output database lives.

To summarize, the above review process can be explained as the following schema.

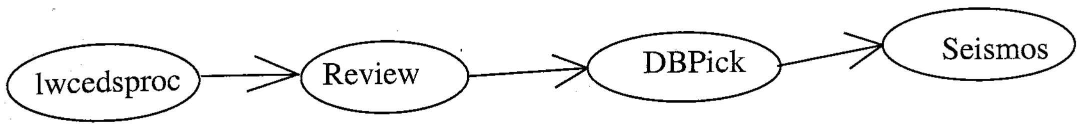


Fig 5. A simple schema illustrating relations between the four routines: lwcedsproc calls review, review calls DBPick and DBPick calls Seismos when processing a local event.

2.3 Introduction to Postgres Database

The above description gives a general idea about the NMT network data acquisition system on which **Matseis-NMT** is based. As is mentioned in introduction, we want to improve the event review process by interfacing **MatSeis** with Postgres databases. We use Postgres database to store the information of earthquake events which come from **LWCEDS**. Postgres is a public domain, open source database management system developed at the University of California at Berkeley (*Yu et al.*, 1993). Postgres is a relational database, composed of tables (or relations), which are made up of columns (vertical) and rows (horizontal). We use CSS3.0 format Postgres database (*Anderson et al.*, 1990; *Anderson et al.*, 1990).

The flatfile tables from **LWCEDS** are already CSS3.0 formatted. To populate the Postgres database, we only need to copy these tables to the Postgres database.

The tables in the CSS3.0 format database are: affiliation, arrival, assoc, gregion, instrument, lastid, netmag, network, origerr, origin, remark, sensor, site, sitechan, wfdisc.

Some basic and frequently used Postgre queries are:

a. to create database:

```
createdb DATABASE;
```

a. to query a database:

```
psql DATABASE;
```

b. to select from a table:

```
select COLUMN from TABLE where WHERE;
```

c. to drop table:

```
drop table TABLE;
```

d. to quit:

```
\q;
```

e. to destroy database:

```
destroydb DATABASE;
```

where DATABASE is the name of the database, TABLE is the name of the table in the database, COLUMN is the field of a table, WHERE is the condition for the query (*Yu et al.*, 1995; *Elmasri et al.*, 1994). Programs are written in Matlab to interface Postgres database with **MatSeis**.

2.4. MatSeis: Overview and Analysis of its Functionality

a. Overview of MatSeis

MatSeis is a seismic data analysis tool written in Matlab which has a large library of signal processing routines and a convenient algorithm development environment (*Harris et al.*, 1997). A series of signal processing tools such as filtering, beaming, spectral analysis, FK analysis, vespagram creation, database editing and mapping are provided and waveform correlation, origin-origin correlation, waveform-travel time correlation are also performed in **MatSeis** (*Harris et al.*, 1997).

MatSeis configures itself and finds environment variables when it starts up. The progra- ms **ms_start.m**, **ms_config.m** set environmental variables. **MatSeis** reads data from databases and writes result of analysis to databases. **MatSeis** reads **LocSAT** format data or ASCII matrix files as travel-time curves.

MatSeis has a series of graphical user interface (GUI) to control the visualization, processing and data analysis routines. There are menu GUIs for file, view, origin, waveform, travel-time, arrival, signal processing, GUIs for reading, writing, showing, deleting or editing such visible objects as waveform, origin, arrival, travel-time and GUIs for listing travel-time curves in **LocSAT** format, ASCII matrix format and master image format and for reading of above format travel-time curves. Through these GUIs, the user can select or set input parameters for different routines.

MatSeis has a set of functions for arrival editing, **LocSAT** initiation, locator parameter initiation and **LocSAT** tool.

On the 'View' menu, a series of mapping functions are used to display the events on the map of the globe and with different projecting methods and zooming functions, together with coastal lines or topography as background.

Matlab functions arrival.m, origin.m, travel-time.m and waveform.m stores and accesses these objects respectively.

b. Analysis of MatSeis Functionality

Matseis reads earthquake data from the following databases: **CSS3.0 flatfile database**, **CSS3.0 Oracle database**, **CSS3.0 local database**, each of which has the following relations (or tables): **arrival, assoc, lastid, netmag, origerr, origin, site, sitechan, wfdisc**.

Data in flatfile database and Oracle database are read in different ways. For the flatfile database, the program '**readrect.m**' reads the whole table as text format rectangular matrix, then searches the matrices for the lines that meet the required conditions (e.g. starting or ending time, orid, arid etc.), and then returns the matched lines and parses the resulted lines or arrays into fields (or attributes) of an object and assign the object such field values thus the origin, arrival and waveform objects can be shown by clicking on the 'show' button on corresponding object menu. Therefore an analyst can start reviewing the event. However, to connect with Oracle database, program '**sql_sel.m**' is used to query the database and return the query result. Using sql is much more convenient because only the query results are returned, thus it takes less memory and time.

To locate an earthquake, **MatSeis** uses **LocSAT**, which uses a generalized least square inverse method to find the location and origin time of an earthquake event. The velocity model is read by clicking on the 'travel-time read' button. After appropriate phases are selected, clicking on 'location' button gives the **LocSAT** tool GUI. For parameters in the this GUI, an analyst needs to update the origin ID, select wanted arrivals, usually Pg (or P) and Sg (or S) are enough for a local or regional event.

The regional area velocity models which we use in locating mining explosions are obtained by ray tracing method, generated from previous velocity models designed by Hans Hartses (1991), which is also used in **LWCEDS** and '**Seismos**' (*Aster*, personal communication).

The reviewing process starts after the data of an event are read. The analyst repicks the arrivals (adding arrivals or moving the original arrivals), reestimates the durations, and then runs **LocSAT**, which gives the location and origin time and then clicks the Magnitude menu to get the magnitude for this event.

After the above review process, there will be changes on the values of fields in different tables. The fields of 'location' and 'origin time' in origin table are changed, 'arrival time' may have different values if new arrivals are added or picks are moved, finally, the original tables in the original database are replaced by the new tables after operating writing origin, arrival, assoc. 'Wfdisc' is not changed because of the way it is read. A time segment of the waveform from the continuous series is read and 'wfdisc' table is not written in the reviewed database. Figure 6 shows the configure of **MatSeis**.

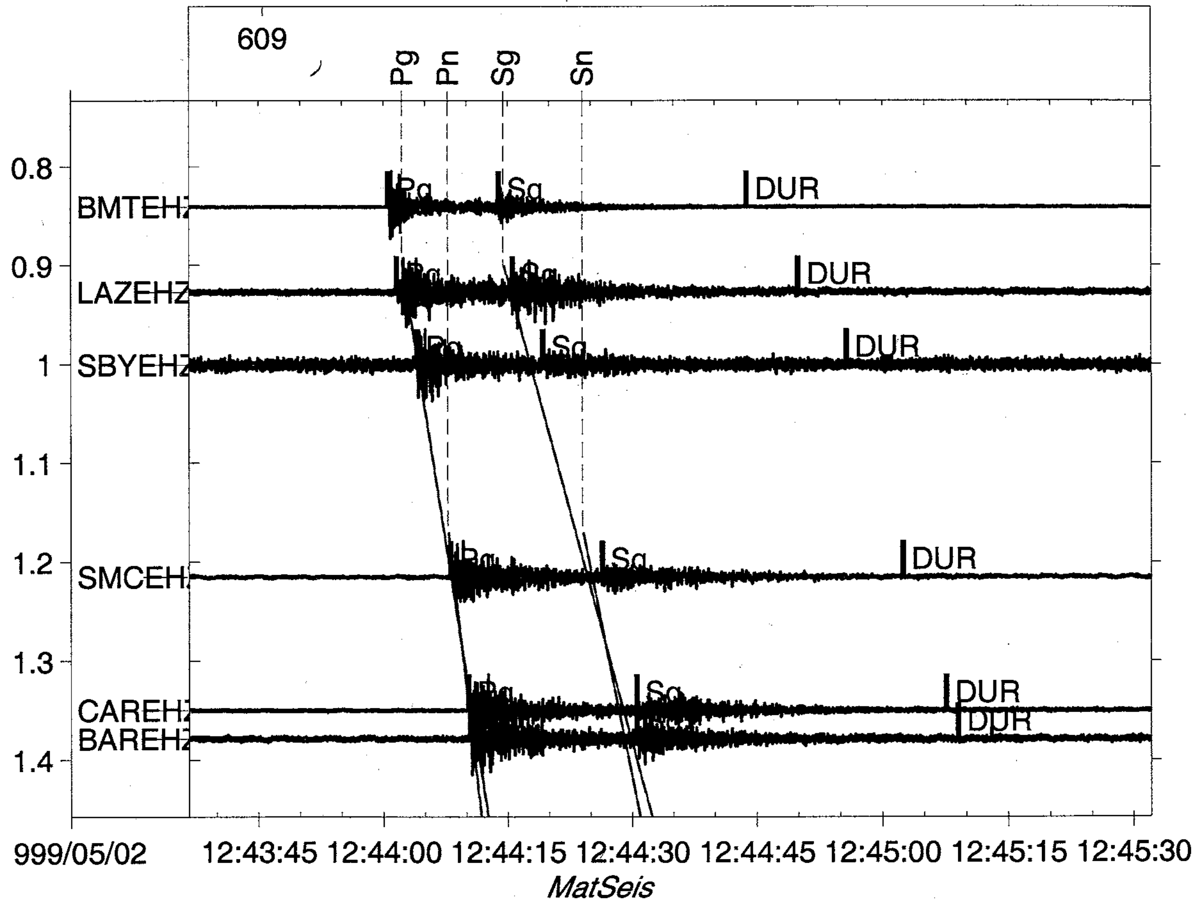


Fig. 6. Sample of the main MatSeis graphical window showing the time-distance waveform display. Function menus controlling the display are not shown.

3. Development of Matseis-NMT

To interface **MatSeis** with Postgres database and create our earthquake data processing tool, **Matseis-NMT**, there are several steps to carry out. First an unreviewed Postgres database should be constructed and populated. Second modifications are needed to the review process so the new review tool can read data from the Postgres database. Finally, we need to store the reviewed event in a 'post review' Postgres database.

3.1. Construction and Population of an Unreviewed Postgres Database

Creation of the unreviewed Postgres database is composed of creating a Postgres database with script 'createdb' and copying earthquake event data into the database with program 'review99' and script **lwceds99** (appendix 1).

Creating a Postgres Database

Seismic events are stored in Postgres databases month by month. Before populating the Postgres database, a database with empty tables for a month should be created. 'Digqks', user of '**LWCEDS**', writes to this database. The original data come from LWCEDS so privilege of writing is granted to 'digqks'. The following script 'createdb' is run for the above objectives. **Site** and **sitechan** tables are not changed for different month because the data are constant parameters of the NMT Network.

```
#!/ bin/tcsh -f

echo \\i runfile |psql POSTGRESDATABASE;

the runfile is as following:

origin30_cre.sql;

origerr30_cre.sql;

lastid30_cre.sql;
```

```
netmag30_cre.sql;
wfdisc30_cre.sql;
arrival30_cre.sql;
assoc30_cre.sql;
site30_cre.sql;
sitechan30_cre.sql;
grant all on arrival, assoc, lastid, netmag, wfdisc, origin, origerr to digqks;
script. createdb
```

In the above runfile, the TABLE30_cre.sql (*Aster*, personal communication) where TABLE is the name of table is just a 'create table' query. For example, the origin30_cre.sql is following,

---

```
create table origin (
lat    float4 NOT NULL,
lon    float4 NOT NULL,
depth  float4 NOT NULL,
time   float8 NOT NULL,
orid   int4   NOT NULL,
evid   int4,
jdate  int4,
nass   int4,
ndef   int4,
ndp    int4,
grn    int4,
srn    int4,
etype  VARCHAR(7),
depdp  float4,
dtype  VARCHAR(1),
mb     float4,
mbid   int4,
ms     float4,
msid   int4,
ml     float4,
mlid   int4,
algorithm    VARCHAR(15),
auth   VARCHAR(15),
commid int4,
lddate DATE
 );
```

Postgres database query: create origin table

Population of a Postgres database

The tables to fill in the Postgres database are origin, origerr, arrival, assoc, lastid, netmag and wfdisc. Population of the Postgres database is carried out by two routines, program **review99.c** and script **lwceds99**.

*Script '**lwceds99**'*

The script **lwceds99** is modified from the combination of 2 C-shell scripts, **lwcedsauto** and **lwcedsproc** written by Mitch Withers (1995). A small part of **lwcedsproc** is used while the whole **lwcedsauto** is moved into **lwceds99**.

Lwceds99 used **lwcedsauto** to generate the data that the unreviewed Postgres database needs and then populate the Postgres database with such data using modified

lwcedsproc routines.

Lwcedsauto controls the operation of **LWCEDS-1.1**. It operates on the trigger data under the trigdir. The program finds how many days' triggers there are under trigdir, and then makes a directory for each trigger under which 3 directories are created. **CSS** is the database directory, where the flatfile tables are stored, **ahfiles** is the directory where the ah format waveform data are stored, **wvmfiles** is a directory where the compressed pcx format waveform data (the same as the trigger data) are stored.

Before the results are moved to the corresponding directories, preliminary processing on the triggered data is performed which includes 5 steps, **PreProc**, **Detector**, **Locator**, **GetMag**, and **PutDB** (*Withers*, 1995). Among these 4 steps, **PreProc** uses the adaptive **STA/LTA** method to filter the data. **Detector** performs frequency domain correlation to correlate processed data with master image and performs grid search for maximum correlation and outputs this grid point which is the coarse location of an event. If the coarse location falls within local grid which approximately encompasses the socorro magma body (SMB) **Locator** correlates processed data with master image in frequency domain and performs grid search for maximum correlation and outputs this grid point, otherwise it will just output the original coarse location.

GetMag estimates duration magnitude of the event. It finds average amplitude prior to estimated first arrival then estimates average amplitude on a moving window after estimated s-arrival. When E_{signal}/E_{noise} <cutoff, the duration is found from $dt=T_d-(T_p+Etime)$ (T_d is the critical point of time where the condition E_{signal}/E_{noise} <cutoff is satisfied, T_p is the P travel time and Etime is the event origin time) and the Socorro duration magnitude is calculated with $Md=2.79*\log10(dt)-3.63$.

PutDB is the last step of **lwcedsauto**. It populates the Origin, Arrival, Lastid, Wfdisc structures, fills the corresponding datascope structure and writes to the appropriate files to the **CSS** directory of each event.

Now that we already have the **LWCEDS**-preprocessed data under 3 directories for each event: **CSS** where the flatfile tables live, **wvmfiles** where the waveform data live and **ahfiles** where the **pcx** data live, the next step is to copy the flatfile tables to the unreviewed Postgres database and move the waveform data and **ah** format data to appropriate directories where such data are stored. This task is realized by the modified **lwcedsproc** which becomes part of **lwceds99**.

Lwceds99 sets the directory of the event as **InDBPath** and the directory where the waveform data and ah format data to be moved as **OutDBPath**. It populates the unreviewed Postgres database by calling a program **review99.c**. After the Postgres database is populated, the waveform data and ah format data are moved from **InDBPath** to **OutDBPath**.

Lwceds99 stores the flatfile tables in Postgres database by running the following command:

```
echo \\i $InDBPath/tmprunfile | pgsql POSTGRESDATABASE
```

where 'tmprunfile', a SQL command file is created by the program **review99.c**.

[redacted] works on the events of a month. Before running this script, a postgres database with CSS3.0 tables for the particular month should already exist. After running **lwceds99**, the origin, arrival, assoc, wfdisc tables are filled with the data of the month. **Site** and **sitechan** tables contains constant values describing the station parameters thus they can be filled in advance.

*program **review99.c***

Review99.c is called by the script **lwceds99**. **Review99.c** creates temporary tables in the unreviewed postgres database and populate them with the flatfile format data generated by **lwcedsauto**, then the previously generated tables copy data from these temporary tables and finally these temporary tables are deleted after the data of one event is populated. The following command in **lwceds99** is executed to access the Postgres database:

```
'echo \\i $InDBPath/tmprunfile | pgsql POSTGRESDATABASE',
```

where **InDBPath** is the **CSS** directory, the temporary database directory for one event, and POSTGRES DATABASE is a Postgres database for the month of this event. 'tmprunfile', generated by review99, is actually a command file that is run in the interface of database. In the Postgres database, it creates a temporary table, then copies from the filtered table generated by '**lwcedsauto**' for the event to the temporary table, then inserts to the 'permanent' table created by the above command the value of the temporary table. The following is an example 'tmprunfile' for the above process for 'origin' table.

```
\\i tmporigin30_cre.sql;
\\copy tmporigin from tmp1Path/tmp1filt.origin;
insert into origin select * from tmporigin;
drop table tmporigin;
```

example of a runfile- a Postgres database query- written by *review99.c*

In the above query, tmporigin30_cre.sql, the same as the previous query origin30_cre.sql is used to creates a temporary table. The tmp1filt.origin is the result of the original 'origin' table filtered by script **psqlprep** (*Aster*, personal communication). This script filters out bad records with more than 4 '0.00's and also change julian date in some tables into epoch date.

```
#!/bin/tcsh -f
#preps CSS flat files for loading into psql with the psql copy command
onintr CLEAN
#change the last field to a load date for each line
/usr/local/bin/gawk '{print $NF}' $1 | epoch | awk '{print $3}' | sed 's/\//-/g'
 > /tmp/psql1$$
/usr/local/bin/gawk '{ for (i=1;i<=NF-2;i++ ) printf("%s\t", $i); printf("%s\n",
$i) }' $1 > /tmp/psql2$$
#paste date field, remove bad lines, send filtered file to stdout
paste -d"      " /tmp/psql2$$ /tmp/psql1$$ | gawk \
'{if ($0 !~ /0.00      0.00    0.00    0.00/ && $1 != -1 && $1 != 0 ) print $0}
'\
| sed 's/Inf/0.0/g'
CLEAN:
/bin/rm -f /tmp/psql?$$ /tmp/psql$$
script psqlprep
```

After the above process, the unreviewed database of a particular month is populated once there is an event for this month.

3.2 Review Process in Matseis-NMT

The event review tool, **Matseis-NMT**, is modified from **MatSeis** to access Postgres database. The modifications of the review process include the following: database setting up, inputting data, relocation of events using **LocSAT** and estimation of event magnitude.

Database Setting-up

Matseis-NMT has different configuration from **MatSeis** because the Postgres database is introduced. Both how it reads from database and how it writes to the reviewed database are different.

Matseis-NMT has 5 database types including CSS3.0 local, CSS3.0 Oracle, CSS3.0 flatfile, CSS3.0 Postgres and CSS3.0 Postgres Rdb among which the 'Postgres Rdb' is the reviewed database where the reviewed events are stored and 'Postgres' is the database where the programs read data from. Currently only Postgres database is in use by New Mexico Tech.

MatSeis is very slow in reading data from a flatfile database, especially reading the large arrival table, thus flatfile database seems impractical for the data sets of New Mexico Tech.

Program db_setup.m sets up the above 5 database types and for each of them, there is a different database setup program where such parameters as database query command and name of tables are defined which are used in the associated programs to access these databases.

Once **Matseis-NMT** starts, the program ms_config.m runs to set up parameters for the routines '**LOCSAT**', 'databases' and 'maptool'. There are also parameter setup programs in these routines respectively and if the parameters can’t find their values in such programs their values will be defaulted as in program ms_config.m

There are different sets of data reading programs to read origin, arrival, waveform and other tables from a database, the following chart shows how origin table is read in **r_orig.m** for different databases (the programming structure for arrival reading and waveform reading are similar to this):

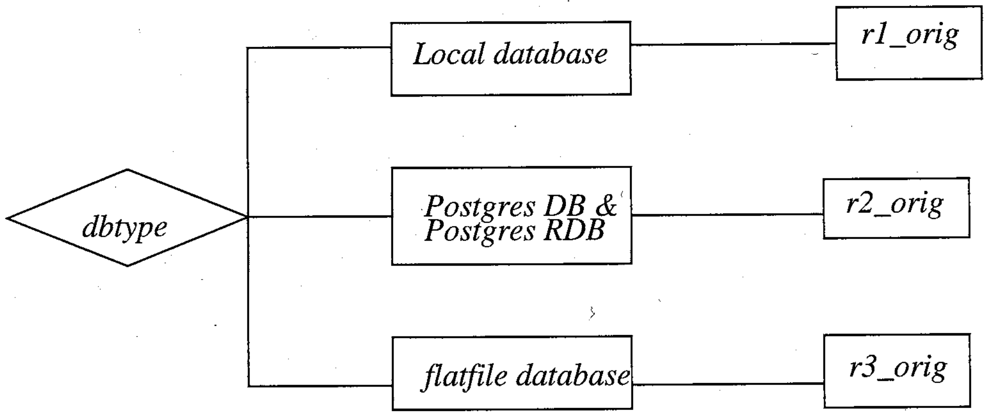


Fig. 7. Schema illustrating how Matseis-NMT reads data from origin table from different databases, it checks the current database type and calls the appropriate function to read origins. Origin data is read from the database and stored in origin objects.

Input of Data

Matseis-NMT accesses the earthquake event data from a Postgres database. It obtains the event data through database query program **sql_sel.m** and then parses the query results in the data reading programs (such as origin reading program r2_orig.m, arrival reading program r2_arriv.m, waveform reading program r2_wave.m) respectively. As a result each object (origin, arrival, waveform) is visible and analysis can be started.

1. Query Database: **sql_sel.m**

This is a frequently used program, adapted from the original **sql_sel.m** from

MatSeis. **MatSeis** uses Oracle database but **Matseis-NMT** uses Postgres database. The query languages for them are quite similar but not exactly the same (*Koch et al.*, 1995).

’**sql_sel.m**’ queries Postgres database using SQL, or standard query language (*Data et a*l., 1993; *Yu et al.*, 1995). For each table, there is a paragraph in the program to query the table because a filter (*Aster*, personal communication) is used to control the format of the returned result.

Basically, the following sentence is used for every query of the database,

```
select COLS from TABLE where WHERE;
```

where COLS are the fields (attributes) of the table, TABLE is the table name, WHERE is the condition for the query. To realize this query, the following runfile is run in the Postgres interface:

```
select COLS into tmp from TABLE where WHERE;
copy tmp to outfile;
drop table tmp;
quit;
```

The query result is returned to the main program which calls **sql_sel.m**.

Sql_sel.m deals with different tables, for each table, the database query is basically similar. However, before returning the query result data to the main program, the data are filtered first because the formats of data in each table are different and the unfiltered data won’t meet the format requirement of the main programs. In origin and arrival tables, the lddate field should be epoch data instead of julian data as in the original tables, thus scripts ’origin_lddate_fix’ and ’arriv_lddate_fix’ (*Aster*, personal communication) are operated on the respective query result data for this modification. The following program is an ’abstracted’ **sql_sel.m** and shows how it works on query the

origin table when it is called by r2_orig.m.

```
function out = sql_sel (TABLE, COLS, WHERE)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Check arguments.
%
if nargin ~= 3
  error ('Wrong number of input arguments.');
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Default output.
%
out = '';
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Get temporary file names.
%
filename = tempname;
filetodel = sprintf ( '%s.del', filename );
runfile = sprintf ( '%s.sql', filename );
outfile = sprintf ('%s.out', filename );
outfile1 = sprintf ( '%s.tmp1', filename);
outfile2 = sprintf ('%s.tmp2', filename);
tmpname = sprintf ('tmp%4.4d',gettid);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Copy sql start-up instructions and select call into run file.
%
```

```
fid = fopen (runfile, 'wt');
if (fid == -1)
  disp ( [ 'Could not open PSQL run file: "' runfile'"' ]);
  return;
end
fprintf ( fid, sprintf ( 'select %s into %s from %s where %s;\n',COLS,...
  tmpname,TABLE, WHERE ) );
fprintf ( fid, '\\copy %s to %s;\n',tmpname,outfile);
fprintf ( fid, 'drop table %s;\n',tmpname);
fprintf ( fid, '\\q;\n' );
fclose ( fid );
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Execute sql command.
%
switch par('MS_DATABASE')
   case 'CSS_3.0_POSTGRES'
      eval ( sprintf ( '! %s < %s', par ( 'MS_DB2_CMD' ), runfile ) );
   case 'CSS_3.0_PostgresRDB'
      eval ( sprintf ( '! %s < %s', par ( 'MS_DB4_CMD' ), runfile ) );
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Check which table to select and do it
     % Do an origin filter
     eval ( sprintf ( '!cat %s | /usr/local/src/Matseis/matseis-1.4/local.nmt/bin/originfilter > %s\n', ...
          outfile, outfile1));
     eval ( sprintf ( '!/usr/local/src/Matseis/matseis-1.4/local.nmt/bin/origin_lddate_fix %s > %s\n', ...
          outfile1, outfile2));
```

```
    eval ( sprintf ( '!/bin/rm a.tmp lddates edates \n'));
      end
  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   % Otherwise
   otherwise
    error ( ' UNKNOWN TABLE!' );
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Read output.
fid = fopen ( outfile2, 'rt' );
if ( fid == -1 )
  disp ( [ 'Could not open outfile file: "' outfile2'"' ] );
  return;
else
  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
  % Read output from file.
  %
 out = fread ( fid );
 out = char ( out );
 fclose ( fid );
 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Delete tmporary files.
eval(sprintf('!/bin/rm %s*',filename);
```

code 1. a compressed sql_sel.m, to illustrate how real sql_sel.m querries the database and return the desired results.

The tables to query are origin, arrival, assoc, network, site, sitechan and all the queries are done in similar way. The output from the query command may not be usable in the loading programs because of their format. Filters are used to control their formats. The filters (*Aster*, personal communication) are just the format of every field in the corresponding CSS3.0 format tables. For example, the origin filter is following,

```
#!/bin/tcsh -f
awk '{printf("%9.4f %9.4f %9.4f %17.5f %8d %8d %8d %4d %4d %4d %8d %8d %7s %9.4f
 %1s %7.2f %8d %7.2f %8d %7.2f %8d %15s %15s %8d %17s\n",$1,$2,$3,$4,$5,$6,$7,$8
,$9,$10,$11,$12,$13,$14,$15,$16,$17,$18,$19,$20,$21,$22,$23,$24,$25)}'
```

2. Parse the Result of Queries

sql_sel.m is called by data reading programs r2_TABLE.m for Postgres database. The result, as the content of a specific object (origin, arrival, assoc, waveform and so on), is reshaped into arrays with different lengths in different reading programs.

Following is one example of querying the origin table:

```
origins =
  33.1638 -109.1430   5.0000  873146364.48352    3245    3245  1997244  -1  -1  -1     -1     -
1    R    5.0000 - -999.99      -1 -999.99      -1    1.88       1          lwceds          LWCEDS       -1
873072000.00000
```

The arrays are then parsed into fields which are characters or integers or floats. These fields are then assigned to the corresponding new objects (origin, arrival, assoc, waveform etc.) and will be used as corresponding gui.data and thus will become objects.

Following is a part of the data parsing program,

```
origin ( 'orid',    index, sscanf ( origins(:,48:56)', '%f' ) );
origin ( 'loc',     index, sscanf ( origins(:,1:30)', '%f', [ 3 inf ] )');
origin ( 'time',    index, sscanf ( origins(:,30:47)', '%f' ) );
values = sscanf ( origins(:,75:107)', '%f', [ 5 , inf ] )';
origin ( 'n_ass',   index, values(:,1) );
origin ( 'n_def',   index, values(:,2) );
origin ( 'n_dp',    index, values(:,3) );
origin ( 'grn',     index, values(:,4) );
origin ( 'srn',     index, values(:,5) );
origin ( 'etype',   index, origins(:,109:115) );
origin ( 'dp_depth', index, sscanf ( origins(:,116:125)', '%f' ) );
origin ( 'dp_type', index, origins(:,127) );
origin ( 'mb',      index, sscanf ( origins(:,128:135)', '%f' ) );
origin ( 'ms',      index, sscanf ( origins(:,145:152)', '%f' ) );
origin ( 'ml',      index, sscanf ( origins(:,162:169)', '%f' ) );
origin ( 'alg',     index, origins(:,180:194) );
origin ( 'auth',    index, origins(:,196:210) );
```

example of program, parsing the origin object

After the origin, waveform and arrival data are read, the analyst can start the reviewing process.

Relocation of Events Using **LocSAT**

Relocation of events in Matseis_NMT is largely the same as that in **MatSeis** which is operated in the review process. The reviewing process contains relocation and magnitude estimation. **LocSAT** relocates the events, 'ms_duration.m' estimates the magnitude of an event.

The location button calls 'locsat_tool' which in turn calls **LocSAT**. **LocSAT** uses least square inverse method to locate earthquakes. **LocSAT** needs the information of origin ID, arrivals and velocity model which is the same used by **Seismos** (*Aster*, personal communication).

A very useful tool in analyzing an event is the 'Measure_Tool' from **MatSeis**. The greatest advantage of this tool is that it allows the analyst to give standard deviation to the picks while picking the arrivals.

There are some factors that affect the quality of the collected data. For some period of time some stations might record noise instead of signal. The noise may be caused by traffic, wind, thunderstorm etc. For example in the summer, a lot of thunderstorms are recorded as triggered events in New Mexico. Such phenomena may last a relatively long time, e.g., 1 or 2 months. Program sel_ar.m filters out stations with noisy records and only traces or stations with 'good' records are used. Stations to filter vary with time so they should not be constants.

```
function [sta_index, sel_sta] =sel_ar(station)

%function sel_ar.m selects stations that is not 'SMC','SBY','GDL','CPR','LEM',

% and returns the station index of such stations. Stations can be changed in the program depending which

one should be deleted.

sel_sta='';

count=0;

for i=1:length(station)
```

```
if (strcmp(station(i,:),'SMC')~=1 ...
  & strcmp(station(i,:),'WTX')~=1 ...
  &strcmp(station(i,:),'GDL')~=1 ...
  & strcmp(station(i,:),'CPR')~=1 ...
  & strcmp(station(i,:),'BAR')~=1 ...
  & strcmp(station(i,:),'LEM')~=1)
   sel_sta=cat(1,sel_sta,station(i,:));
   count=count+1;
 end
end
sta_index='';
for i=1:count
 row=strfind(sel_sta(i,:), station);
 sta_index=unique(cat(1,sta_index,row));
end
code  sel_ar.m
```

To relocate a regional event, **LocSAT** is preferred over **Seismos** because relocation using **LocSAT** is more robust for regional events, in another word, the program converges more easily than '**Seismos**' does. Besides, the regional master image used by **Seismos** to locate a regional event didn't incorporate accurate lateral heterogeneities which makes the location not accurate either.

Estimation of Event Magnitude

Duration in **MatSeis** is obtained by clicking on the 'Magnitude' button. Duration of each trace is obtained by measuring the time distance from first arrival, 'P' for local events or 'Pn' for regional events, to 'DUR' then use the equation

$$mvec = 2.79*\log10(dvec)-3.63$$

The magnitude of the event is given by the mean value of the magnitudes from all the traces.

3.3. Storing the Results in a Reviewed Postgres Database

The unreviewed events are stored in an unreviewed database which is named 'socorro_YYMM', where 'YY' is 2 digit year and'MM' is a 2 digit month. For example, all the events of January 1999 are stored in the database 'socorro_9901'. All the reviewed events are stored in the reviewed database which is named 'rdb_YYMM', month by month.

An event is classified into the following categories: *a.* replace, *b.* deleted, *c.* explosion, *d.* local, *e.* regional, *f.* teleseism.

For the case of 'replace', the event must be read into **Matseis-NMT** from the reviewed database, i.e. 'rdb_YYMM'. After reviewing, the event in the reviewed database is deleted and the result of the new review process is written in the reviewed database. For an event that is read into Matsei-NMT from the unreviewed database, protection is made so that this event is not replacable.

Noise is classified as 'deleted' and is deleted from the database from which they are read (noise event is not written in the reviewed database so they can only be read in from unreviewed Postgres database). For a 'deleted' event, 'origin' table is deleted, associated 'assoc' table is deleted and the 'arrival' table which associates with the 'assoc' table is deleted. Based on the starting time and end time of the event, wfdisc in the corresponding time duration is also deleted.

In the origin table, 'explosion', 'local', 'regional' events are different in the 'etype' field. 'explosion' has etype of 'ex', 'local' has etype of 'l', 'regional' has etype of 'r'. Four tables, origin, arrival, assoc and wfdisc, for one event are written to the reviewd database.

Teleseisms are not relocated and their magnitudes are not estimated and the original tables are copied into the reviewed database (they don't have arrival tables).

The events are written to the 'reviewed Postgres database' and this database might be reviewed again. To avoid bad lines (containing more than 4 '0.00's) written and secure that the tables in the reviewed databases are readable by **MatSeis**, as we did in writing in the unreviewed database by the script **lwceds99**, we use the script '**psqlprep**' in the w2_TABLE.m programs where TABLE is the name of the table to write in. In addition, the lddate field in 'origin', 'arrival', 'assoc' tables are julian date and must be changed into epoch date to be CSS3.0 formatted, which is also realized in '**psqlprep**'.

To prevent the same events from unreviewed database from being written to the reviewed database more than once, Program duplc.m checks if the event to be written already exists in the reviewed database and if it does it is not written, otherwise it is. However, events read from a reviewed database can be replaced by a later review result.

Program duplc.m queries the reviewed database for the 'orid' and compares the 'orid' with the 'orid' to be written to avoid the duplicated events written, where 'orid' is one field in origin table standing for the event id.

```
function out=duplc(origin)

% DUPLC to check if one orid is already in the reviewed database

orgs='';

COLS='orid';
```

```
TABLE= 'origin';

WHERE = sprintf ( 'orid = %d', origin );

orgs = [ orgs; selct(TABLE,COLS,WHERE) ];

if size(orgs,1)==1

  out=0;

else

  out=1;

end

cdb=par('MS_DATABASE');

par('MS_DATABASE', cdb);
```

code duplc.m

4. Result Analysis

We are successful in storing the earthquake events collected by NMT network in Postgres databases. This process will be performed automatically (it can be run in a cronjob, for example) and then the events can be reviewed in near real-time. **Matseis-NMT** fulfills our objective of interfacing **MatSeis** with Postgres database.

Matlab is a versatile and reflexble language, which makes modification of **MatSeis** a fairly easy job. **MatSeis** is designed in a convenient way so that new functions can be added and tested in the command window then moved back to the **MatSeis** package. Functions are added to the control GUIs with ease(e.g., Postgres database setup is added to the original 'file' menu GUI, and a filter is used on the **LocSAT** 'arrival-edit' GUI so that phases 'DUR' are not shown).

Basically, we are adding one more database (Postgres database) access function to **MatSeis** and we made modifications on **MatSeis** so that **Matseis-NMT** writes events to a reviewed Postgres database. Matsei-NMT can write the tables for an events all in once while **MatSeis** writes to the output database table by table. The event classification menu is added and through operations in this menu an event is classified and will be written (if an event) to the reviewed database or deleted (if not an event) and the reviewed database can be reviewed again.

Analysis of the result from the reviewed Postgres database provides some interesting facts to discuss: the location of **LocSAT** and the similarity of the event groups from the same mine.

4.1 Comparison of LWCEDS Location with LocSAT Location

Location of the events can be seen from the 'View->Map Tool' in **MatSeis**.

Statistically, among the events of the late half year of 1997, we have:

Table 1: Distribution of Different Event Types among the Events from July, 1997 to December, 1997

DB name	No. of total events	No. of local events	No. of explosions	No. of teleseisms
rdb_9707	116	27	64	25
rdb_9708	122	22	88	11
rdb_9709	112	37	57	18
rdb_9710	96	12	67	16
rdb_9711	68	16	38	14
rdb_9712	59	22	23	14

table1. Distribution of different event types in the database

As a total, 23.7% of the events are local events, mainly reflections from the magma body, 58.8% of the events are explosions from the mines in the range of New Mexico and nearby states and teleseisms are 17.1%.

Comparisons are made between the **LWCEDS** locations (from unreviewed Postgres database) and **LocSAT** locations (from reviewed Postgres database) of mining blasts. Obviously, the relocated events are more converged to the minings (Figure 8 through Fig 25) compared to the original locations given by **LWCEDS**.

Three major groups of mining blasts located at Morenci, Taylor, Tyrone are discovered. Their geographic coordinates are shown in the following table:

Table 2: Mines and Their Locations

Name of Mines	Latitude	Longitude
Taylor	35°08	107°23
Tyrone	32°40	108°20
Morenci	33°05	109°18

Table 2. Location of Mines where the explosions are from.

4.2. Event Identification and Similarity Analysis

After the software of **Matseis-NMT** was preliminarily finished, we have been using it to review events. The results of the review process, i.e., the reviewed databases (RDB), are used as testing data in the project of 'cluster analysis' (*Carr et al.*, 1999) for the objective of separating events from different hypocenters.

We selected mining blasts from the RDBs for the test. This analysis is performed on the waveforms of the events. [redacted]

As the characteristics used to cluster, average Lg-P time, typical SNR are estimated for the selected 60 explosions from New Mexico Tech network, Typical SNR, shape of the waveform and energy of the signal in spectrogram are also analyzed.

The method of clustering is as follows: first, randomly divide events from each mine into a training group and a testing group, second, find good SNR events in the training set to run select test on, third, start with Lg-P time to separate events. Preprocessing of the waveforms includes trying different tapers, using Hilbert envelopes. Different methods are tried to group the events including flexible method, complete method, single method and group mean method (*Carr et al.*, 1999).

60 mining blasts from 3 mines (20 from x1, Tyrone, 20 from x2, Taylor, 20 from x3, Morenci) collected by New Mexico Tech are tested in this project. The 'dendro-gram' (Figure 26) show how well the events are separated. Clearly, the three mines x1, x2, and x3 are completely separated except that 3 x1s are grouped with x2s.

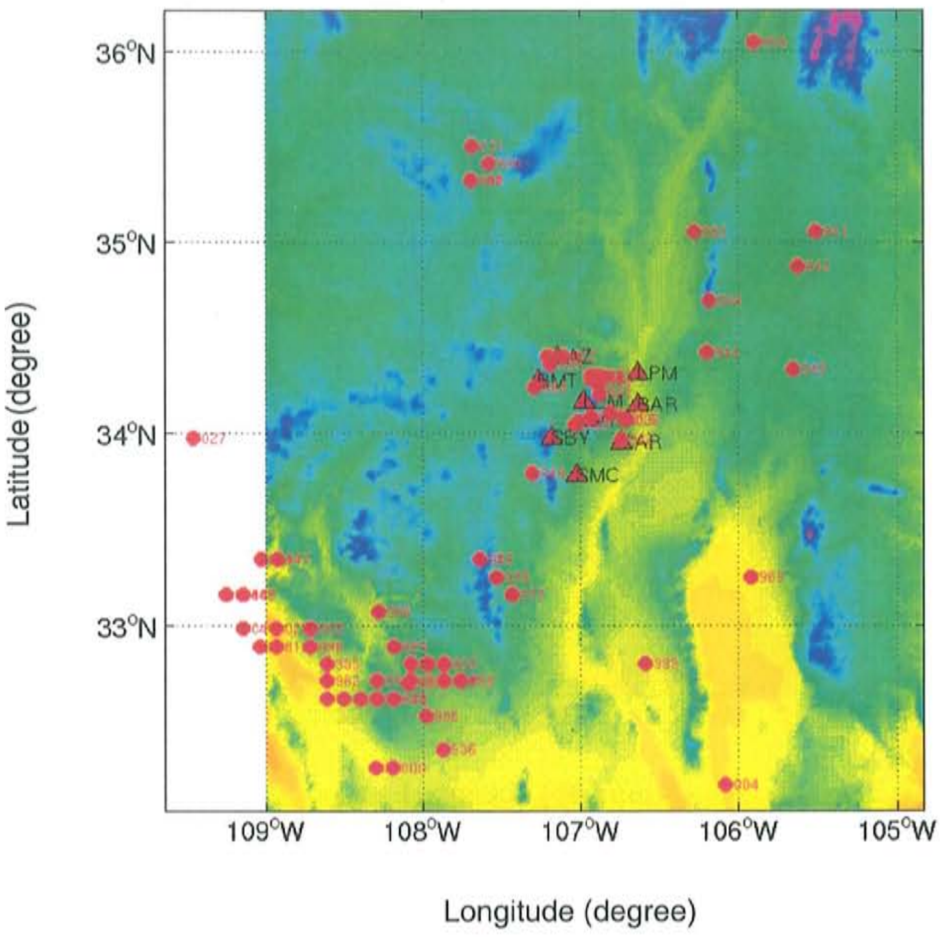


Fig.8. LWCEDS locations of local and explosion events of July, 1997. Data are from unreviewed database socorro_9707.

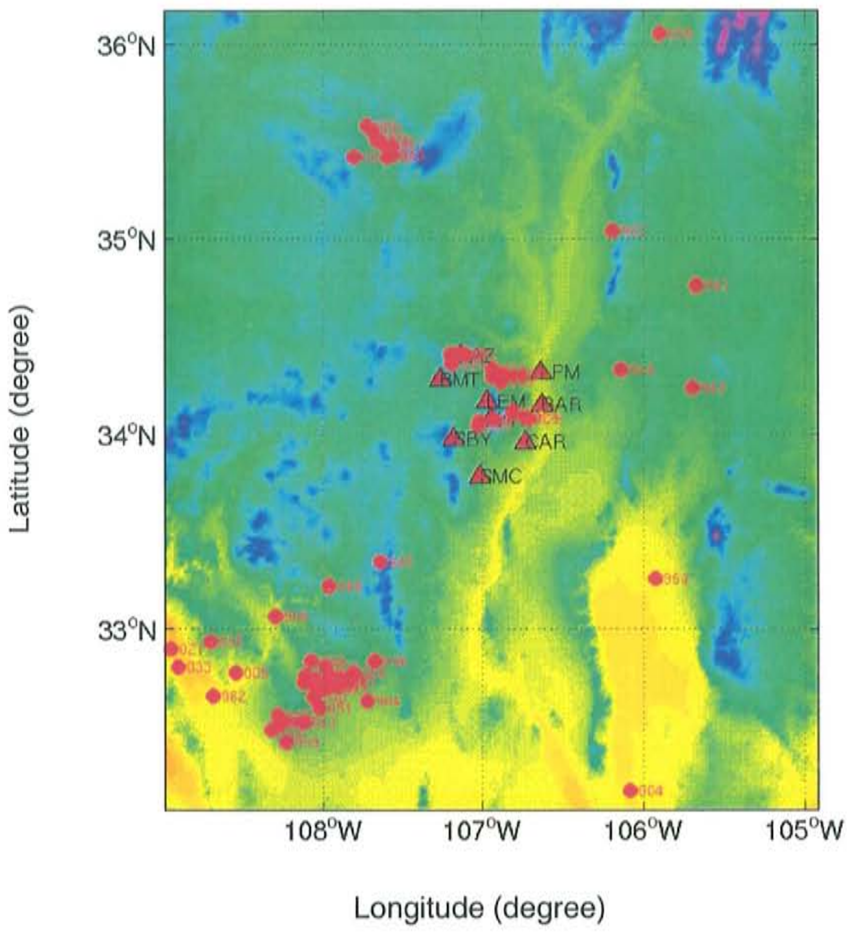


Fig.9. LocSAT locations of local and explosion events of July, 1997. Data are from reviewed Postgres database: rdb_9707.

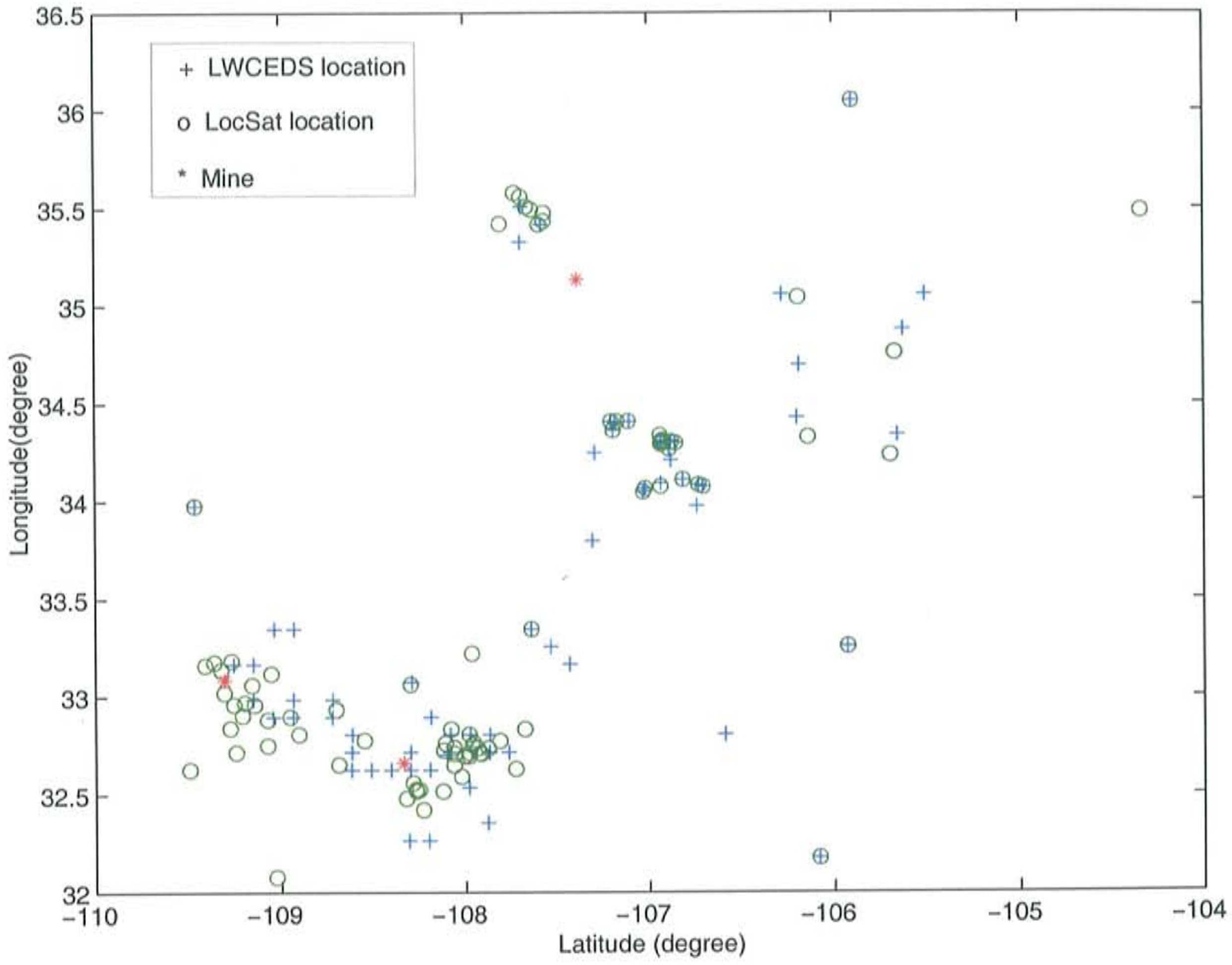


Fig. 10. LocSAT locations compared with LWCEDS locations of the events of July, 1997. This Figure shows that LocSAT locations are more converged to the mines.

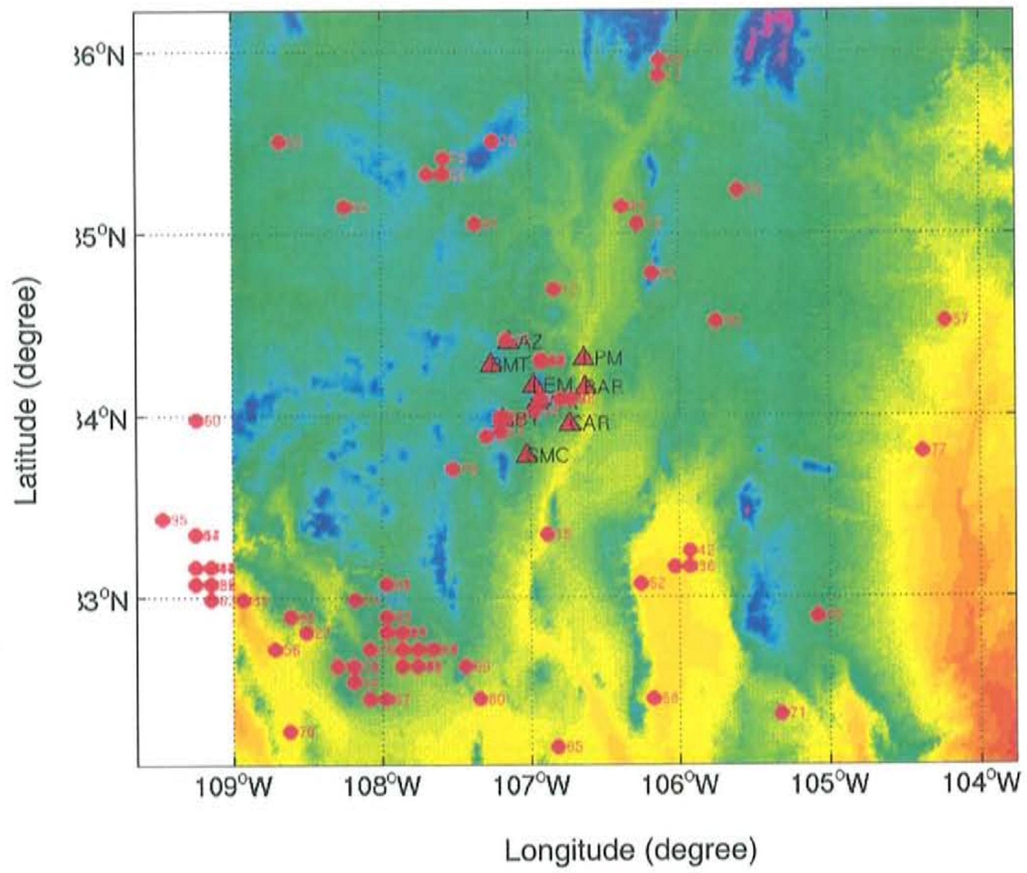


Fig.11. LWCEDS locations of local and explosion events of August,1997. Data are from unreviewed Postgres database: socorro_9708.

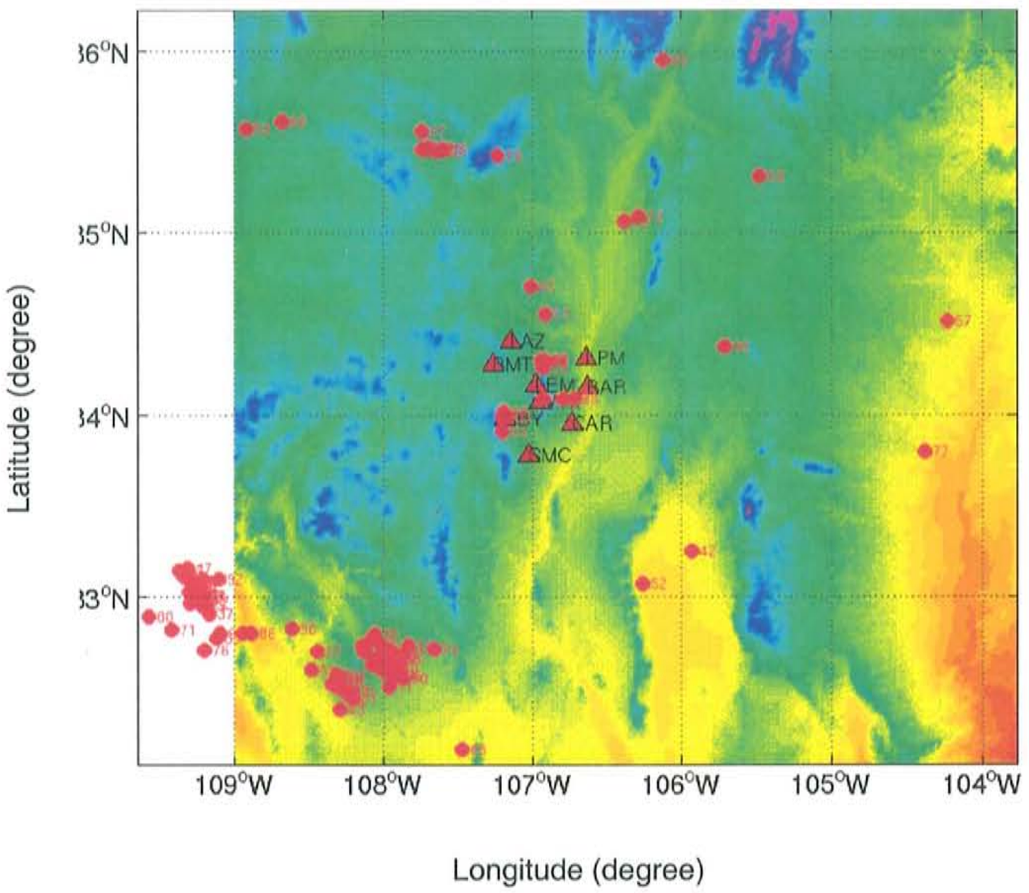


Fig.12. LocSAT locations of local and explosion events of August, 1997. Data are from reviewed Postgres database: rdb_9708.

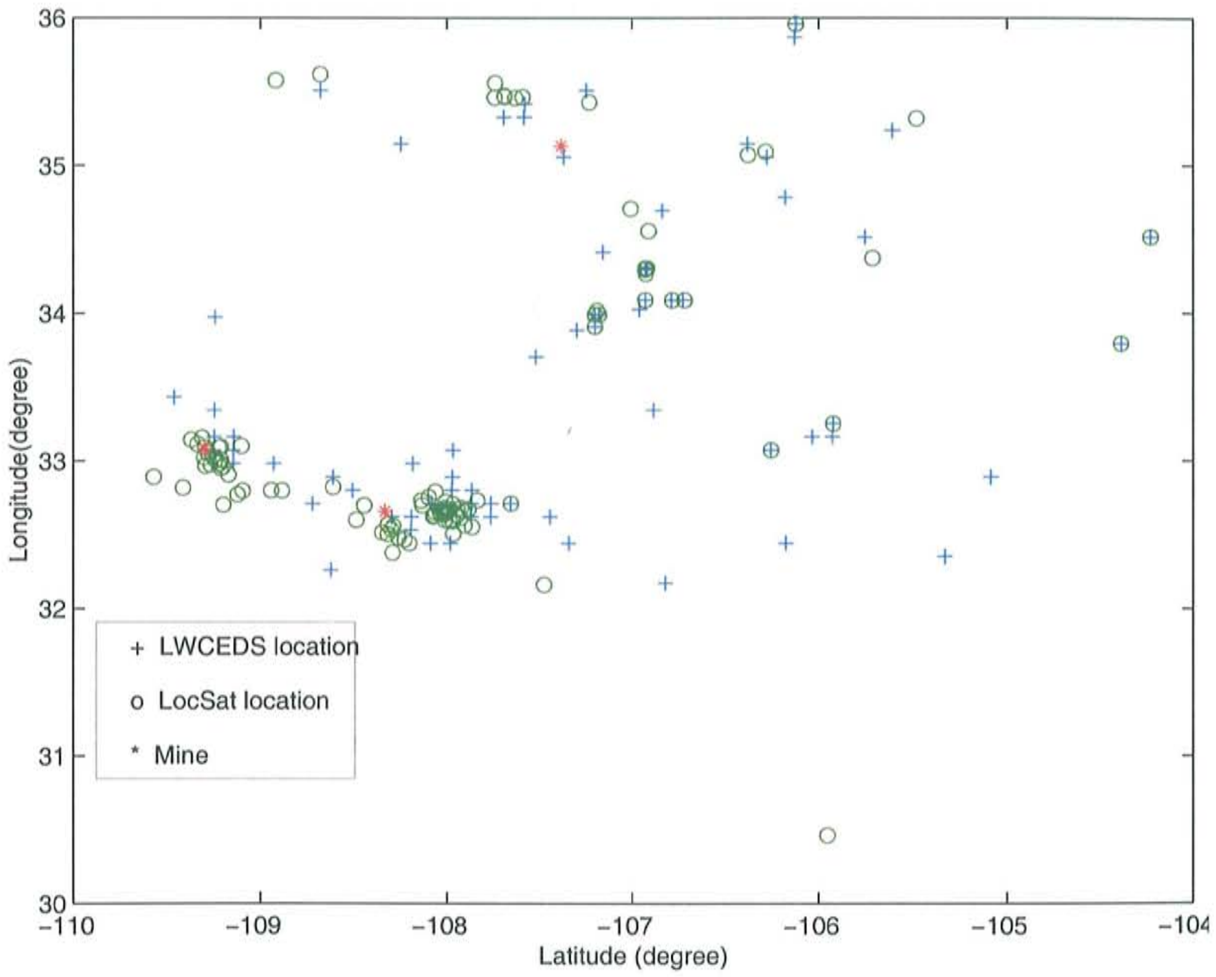


Fig. 13. LocSAT locations compared with LWCEDS locations of the events of August, 1997. This figure shows that the LocSAT locations are more converged to the mines.

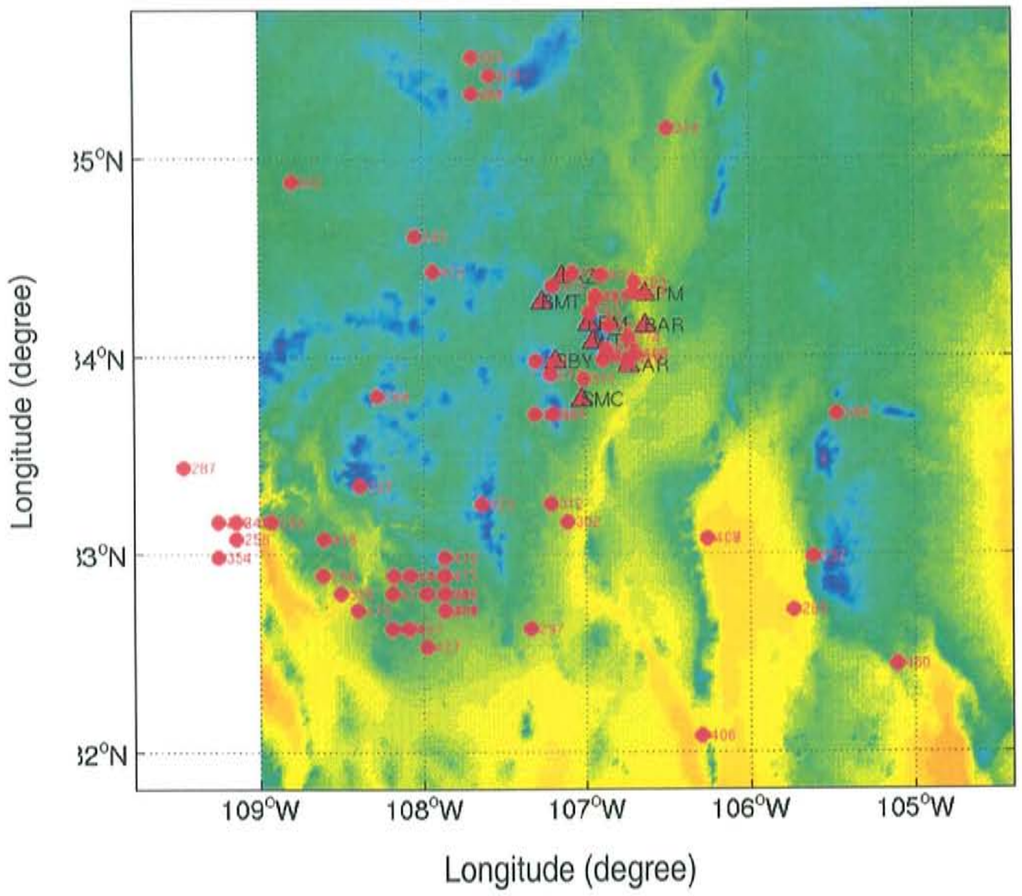


Fig.14. LWCEDS locations of local and explosion events of September, 1997. Data are from unreviewed Postgres database: socorro_9709.

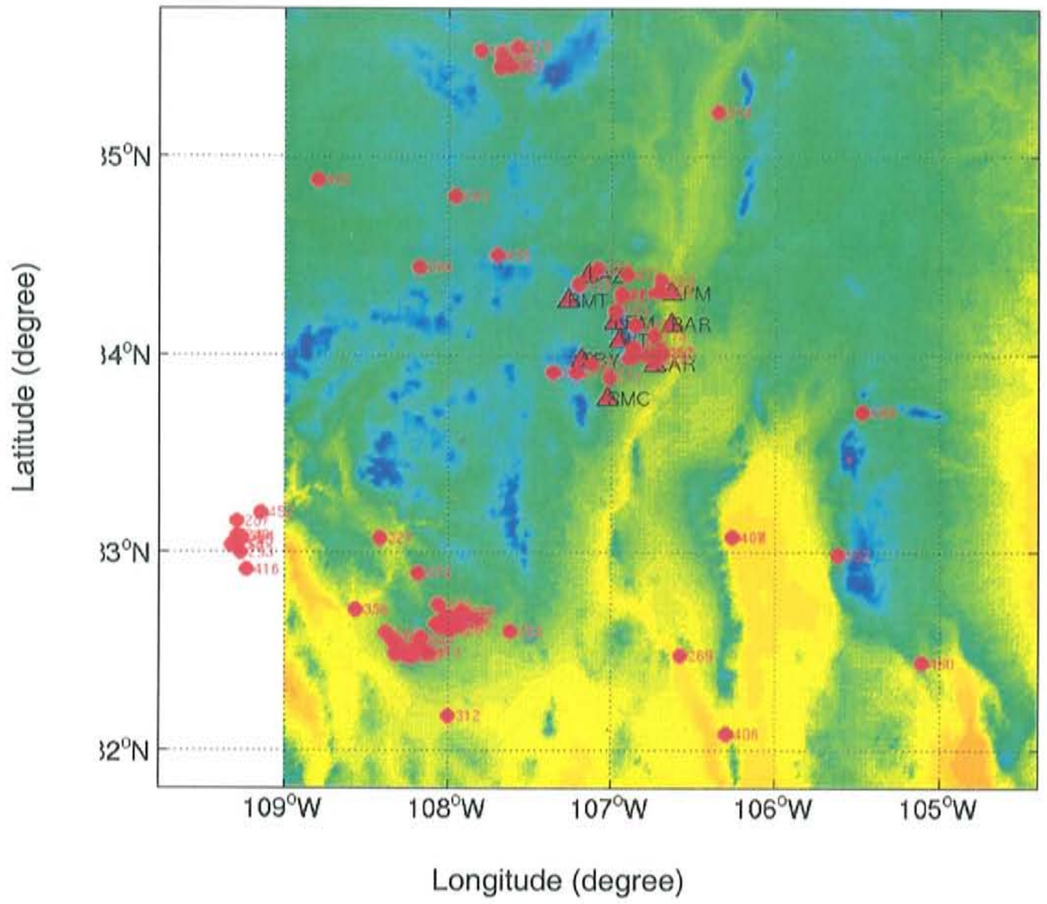


Fig.15. LocSAT locations of local and explosion events of September, 1997. Data are from reviewed Postgres database: rdb_9709.

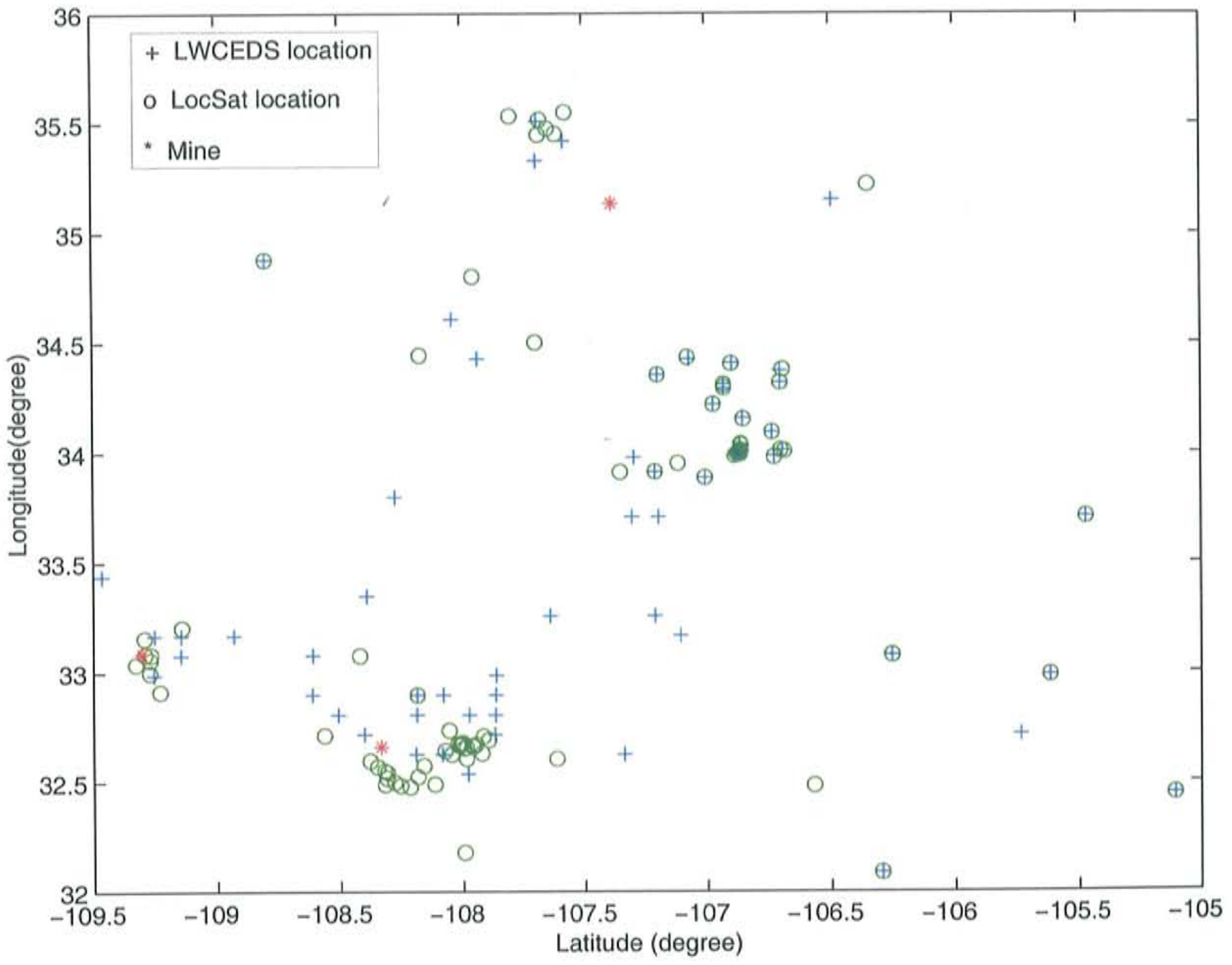


Fig.16. LocSAT locations compared with LWCEDS locations of the event of September, 1997. This figure shows that LocSAT locations are more converged to the mines.

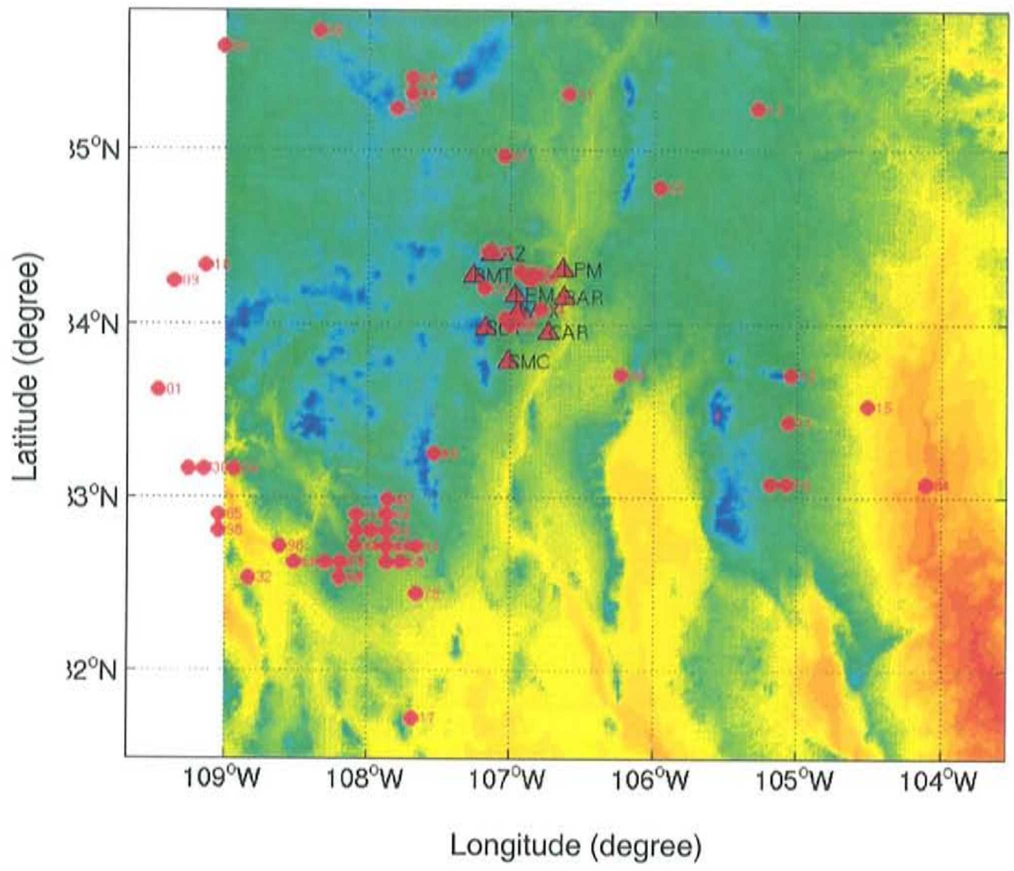


Fig.17. LWCEDS locations of local and explosion events of October, 1997. Data are from unreviewed Postgres database: socorro_9710.

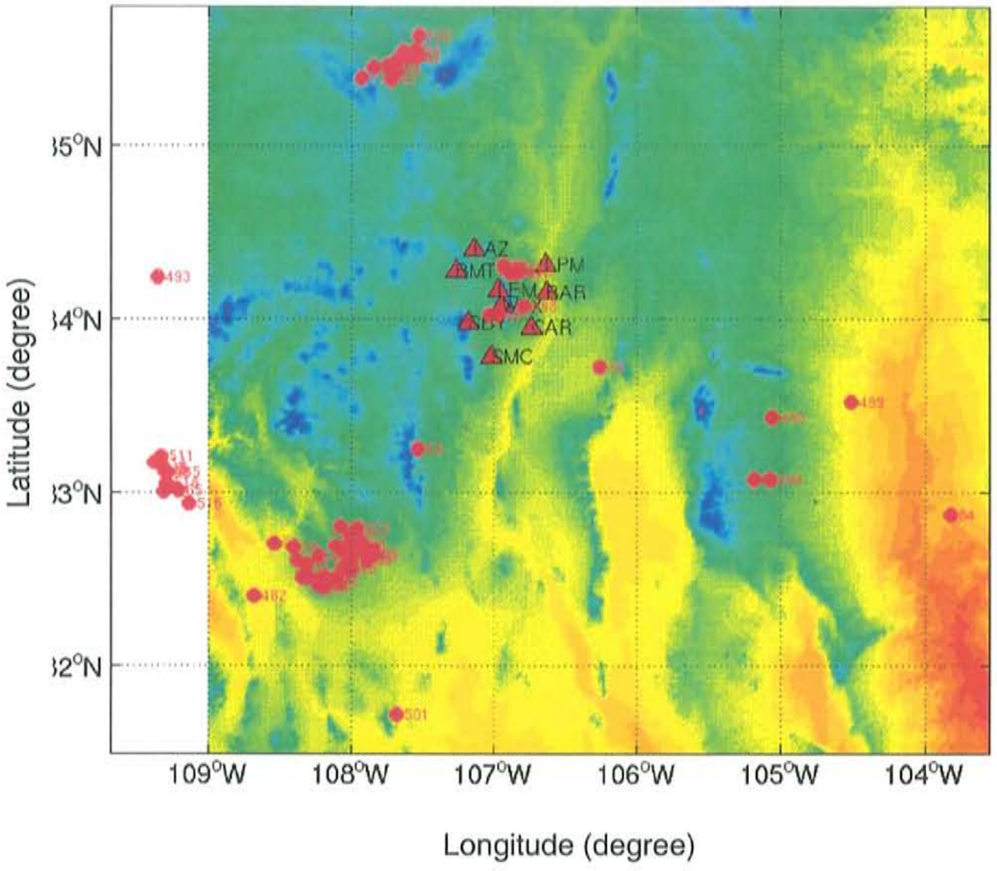


Fig.18. LocSAT locations of local and explosion events of October, 1997. Data are from reviewed Postgres database: rdb_9710.

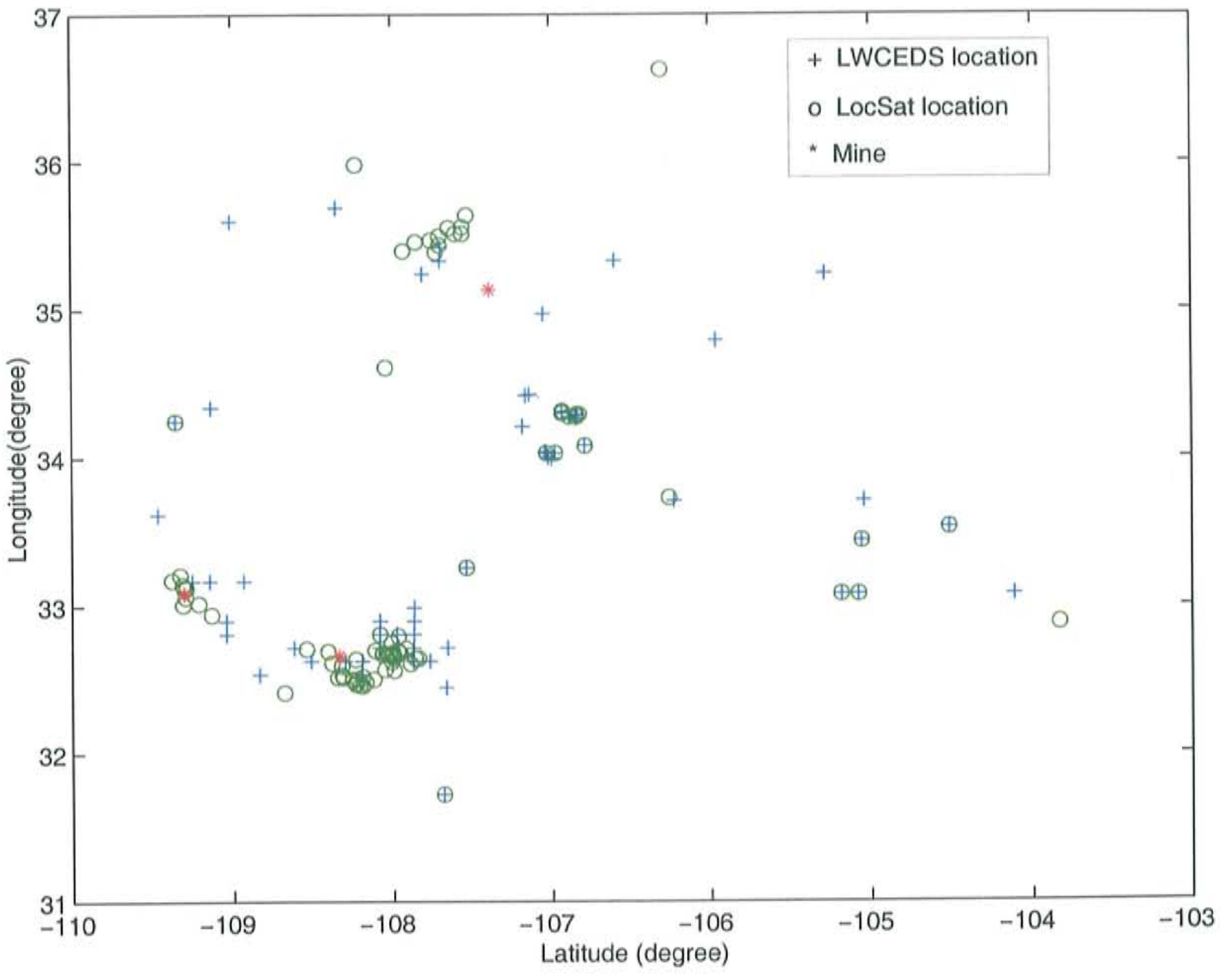


Fig. 19. LocSAT locations compared with LWCEDS locations of the events of October, 1997. This fiugre shows that LocSAT locations are more converged to the mines.

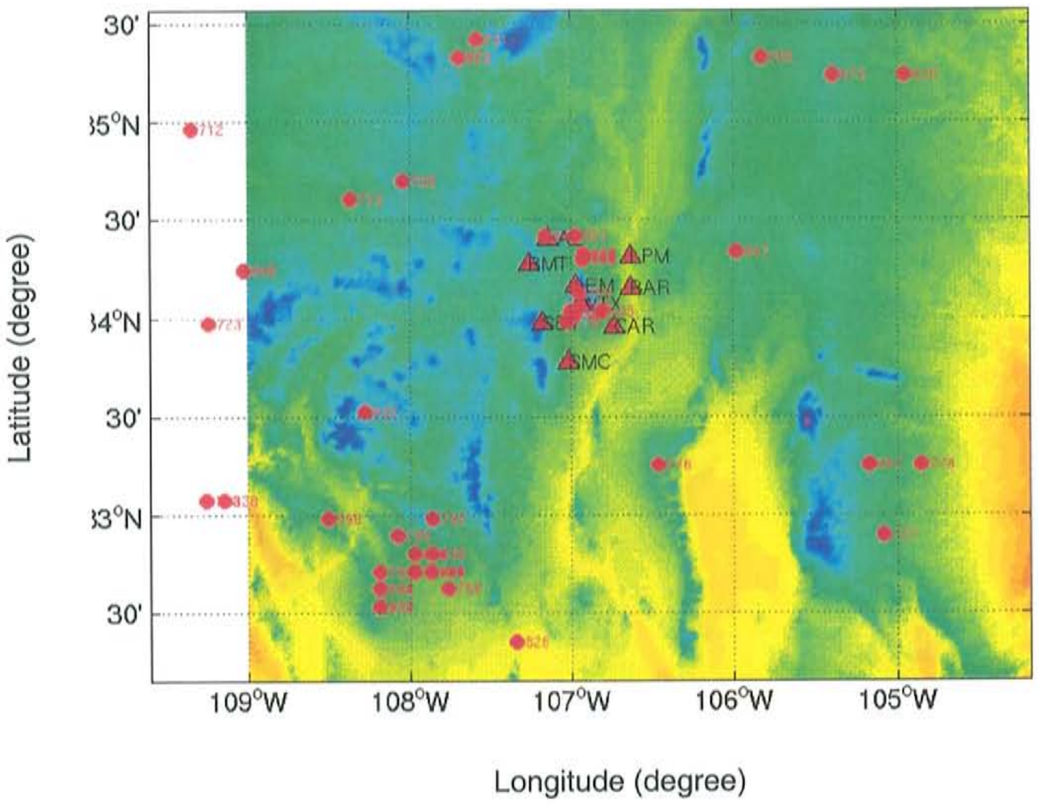


Fig.20. LWCEDS locations of local and explosion events of November, 1997. Data are from unreviewed Postgres: socorro_9711.

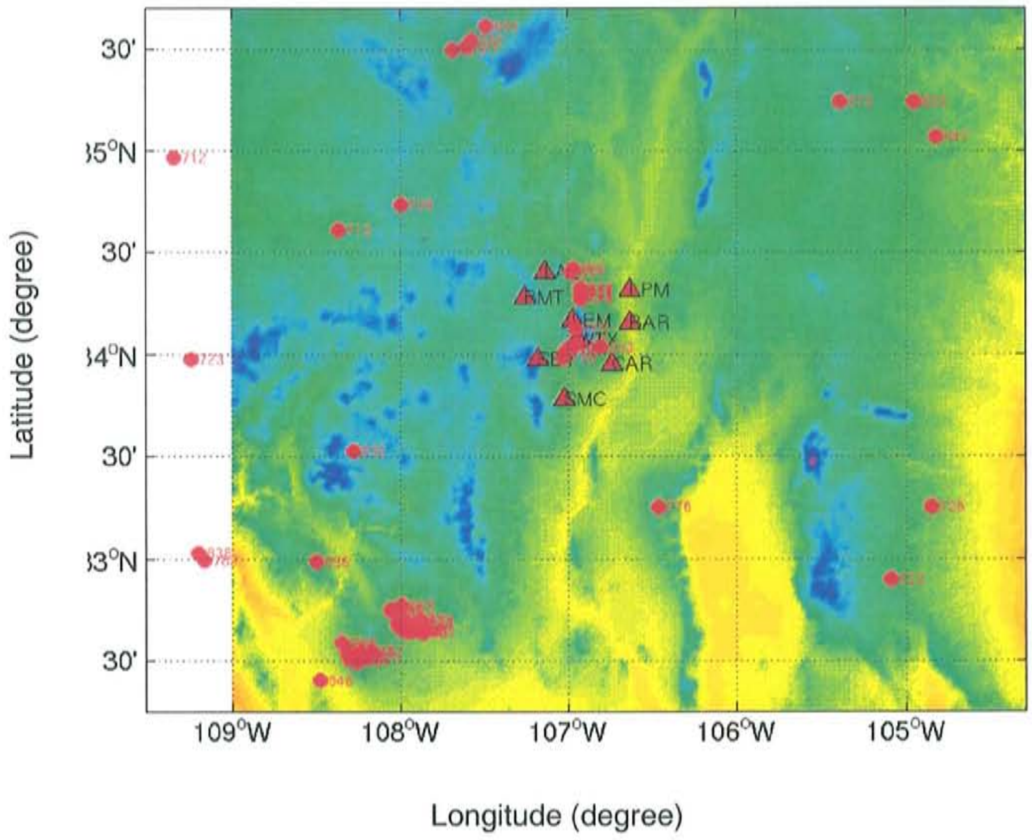


Fig.21. LocSAT locations of local and explosion events of November, 1997. Data are from reviewed Postgres database: rdb_9711.

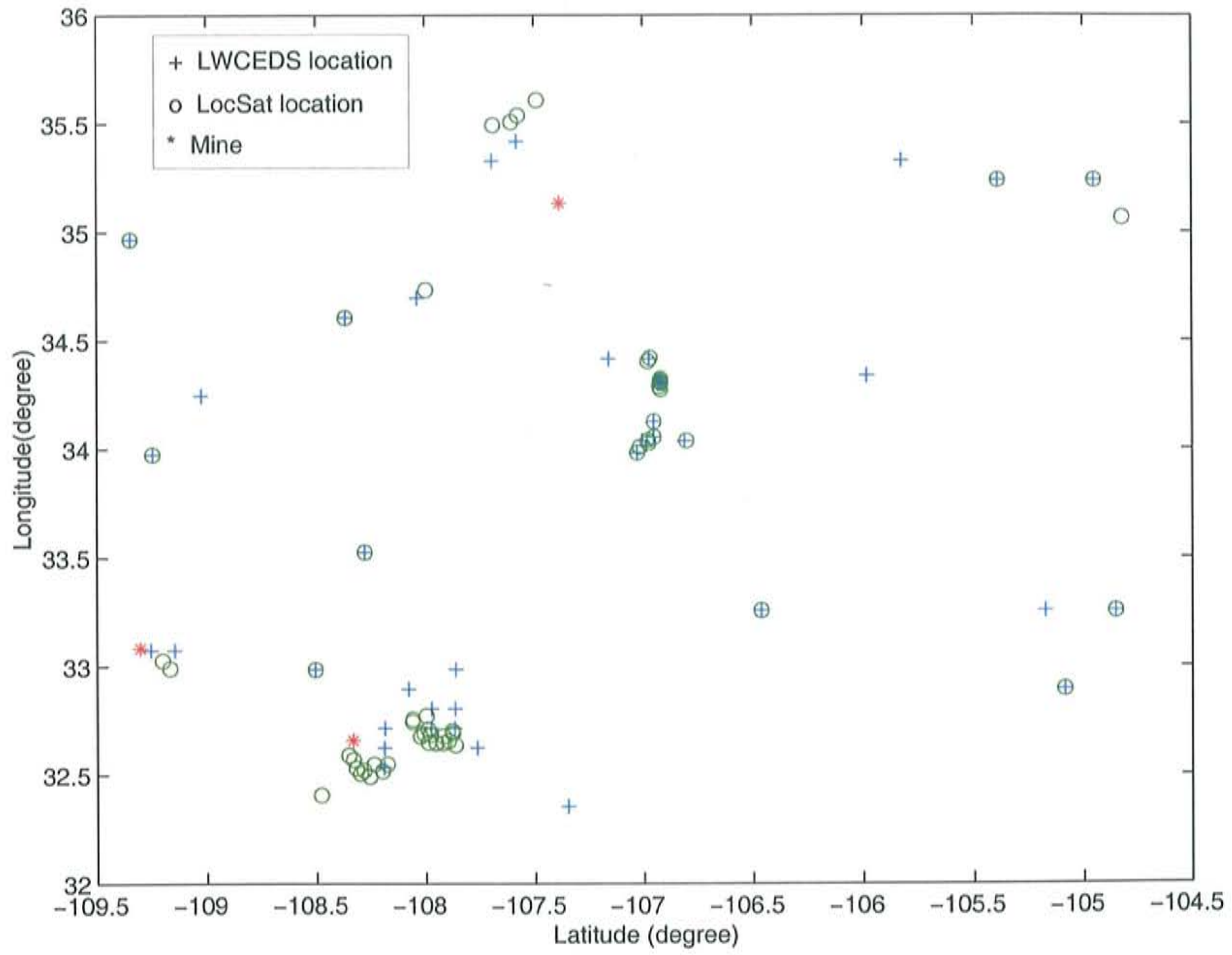


Fig. 22. LocSAT locations compared with LWCEDS locations of the events of November, 1997. This figure shows that the LocSAT locations are more converged to the mines.

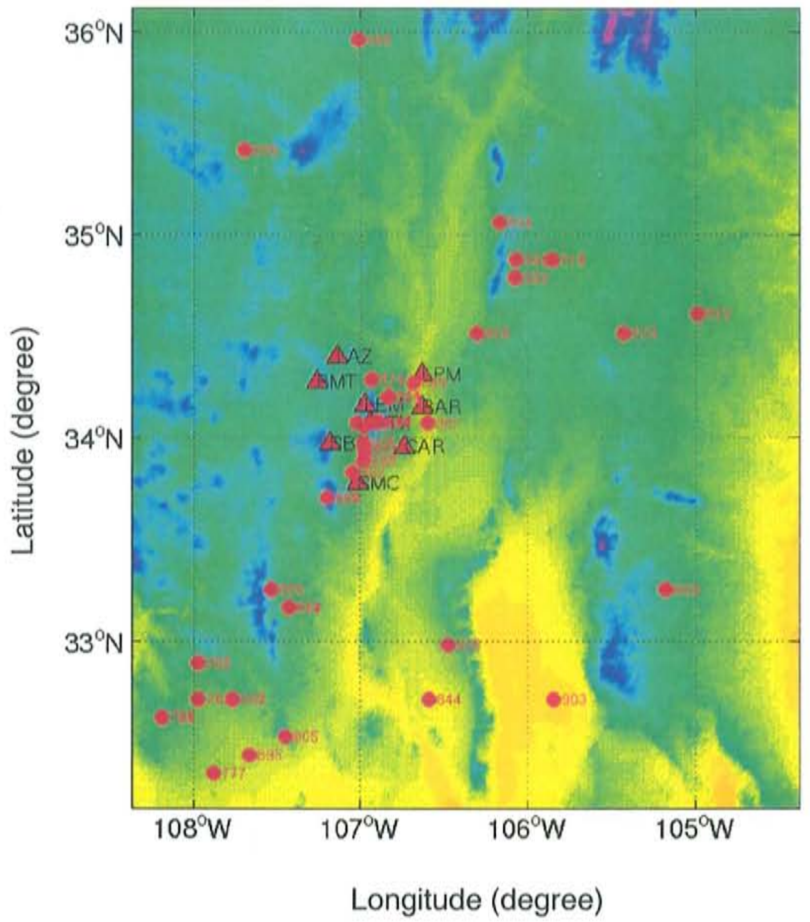


Fig.23. LWCEDS locations of local and explosion events of December,1997. Data are from unreviewed Postgres database: socorro_9712.

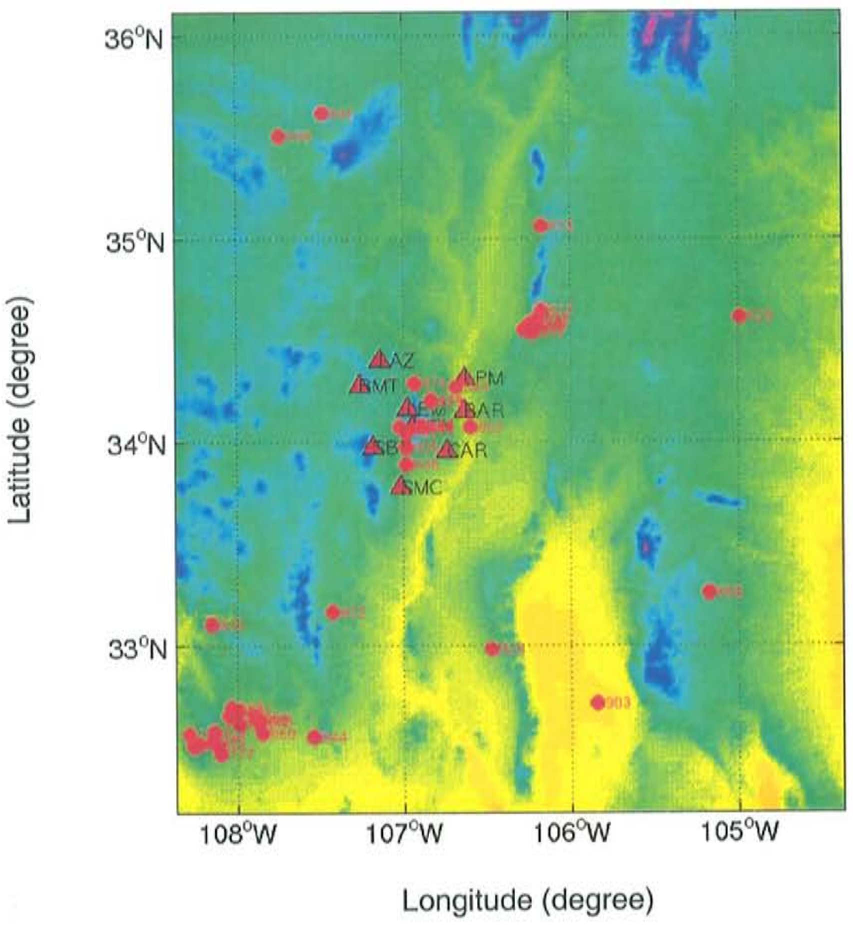


Fig.24. LocSAT locations of local and explosion events of December,1997. Data are from reviewed Postgres database: rdb_9712.

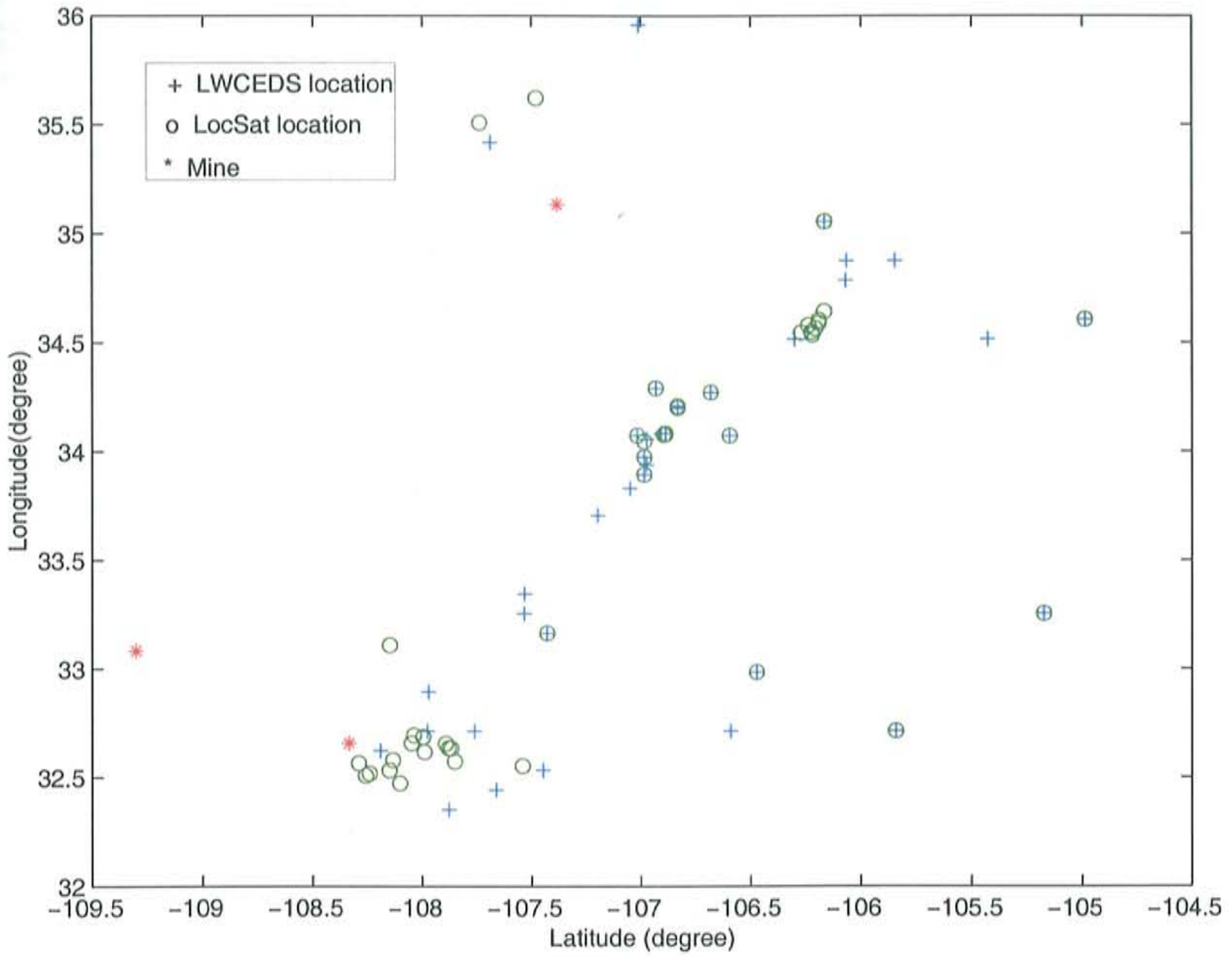


Fig.25. LocSAT locations compared with LWCEDS locations of the events of December, 1997. This figure shows that LocSAT locations are more converged to the mines.

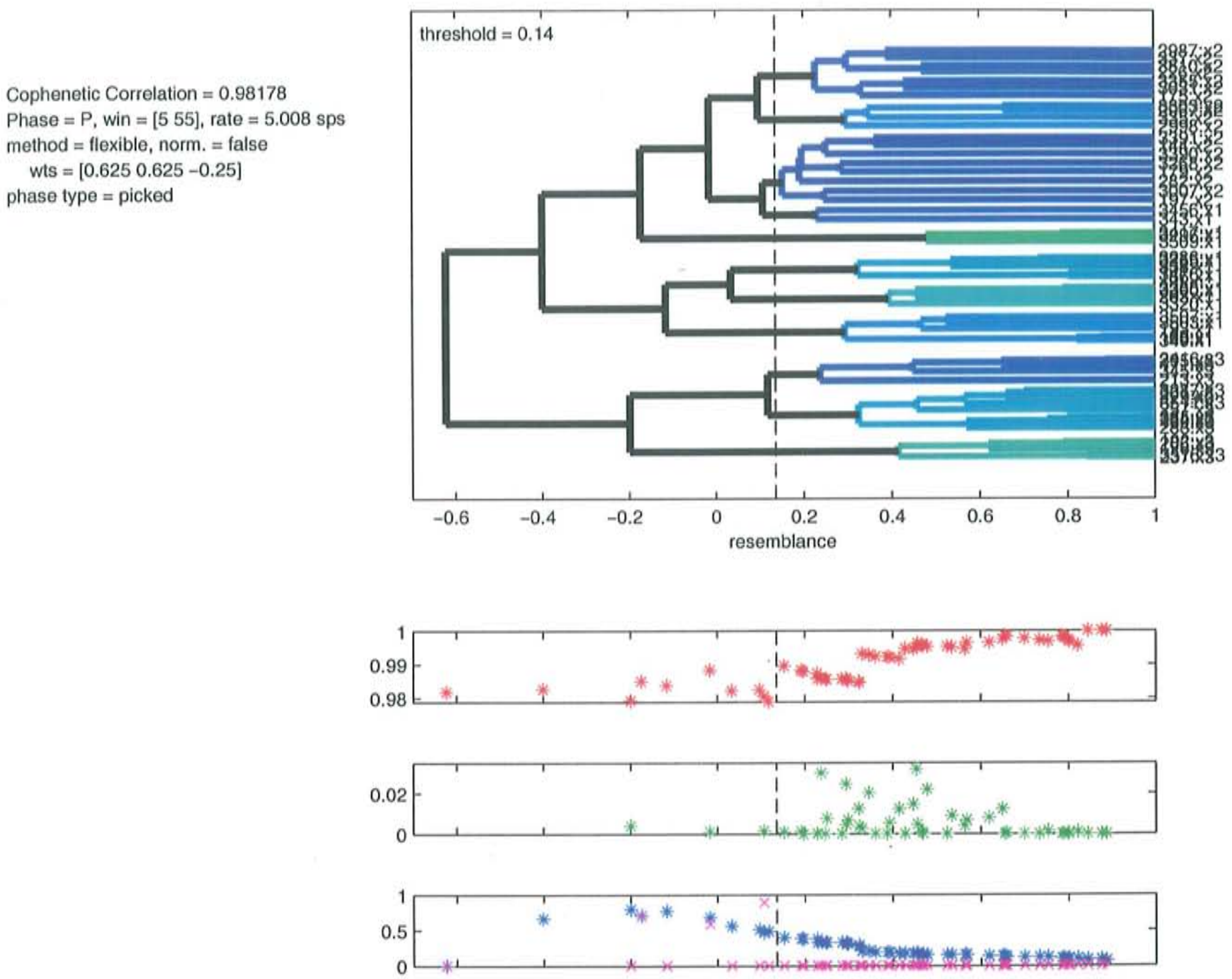

Cophenetic Correlation = 0.98178
Phase = P, win = [5 55], rate = 5.008 sps
method = flexible, norm. = false
wts = [0.625 0.625 -0.25]
phase type = picked

Fig.26. 'DendroPlot' of 60 mining explosion events in 1997. Similarity among the groups of x1, x2, x3 is obvious except that several x1's are grouped as x2's. x1 is Tyrone, x2 is Taylor and x3 is Morenci.

5. Possible Further Development

Development on **Matseis-NMT** in the future can be realized through improvement of both the procedure of populating the unreviewed Postgres database and the review process with **Matseis-NMT**.

In the database populating process, the program **review99.c** reads from the unreviewed database, writes the runfile and appends the tables under **InDBPath** to the tables under **OutDBPath**. This routine is not simplified enough. Among the above procedures, appending the tables to the **OutDBPath** tables should be emitted because all the tables are already populated in the Postgres database. However, if the flatfile tables are kept so as to keep a backup of the original flatfile Datascope, this process is still necessary.

Effort could be placed in the future if we want to use other database (say, Oracle, Sybase) than Postgres to store the event information. Only little modification over the current script **lwceds99** and program **review99.c** should be enough.

There are redundant procedures too in the review process using **Matseis-NMT** especially in the input/output of the databases. Associations among orid, start time, end time, arid etc. could be more completely used so that some data reading process can be simplified or even deleted and thus the reading data from unreviewed database process can be even less time consuming. The current version is still awkward somewhere.

In general, **Matseis-NMT**, based on **MatSeis**, is a fairly complete seismic data processing tool which is proved very useful in earthquake data analysis. It fulfills the objective of reading seismic data from different databases (oracle, postgres, flatfile), performing a series of signal processing, accurately relocating the event and get a good estimation of the magnitude, store the reviewed event in the reviewed Postgres database (into Oracle database can be done by writing the tables one by one).

As is expected by the developers of **MatSeis**, users can still add new functionality to **MatSeis** although the functionality in **MatSeis** seems more than adequate (*Harris et al.*, 1997). We are hoping the same with **Matseis-NMT**.

REFERENCES:

Allen, R.V. (1978). Automatic Earthquake Recognition and Timing from Single Traces, *Bull. Seism.Soc.Am.*, 68, 1521-1532.

Allen, R.V. (1982). Automatic Phase Pickers: Their Present Use and Future Prospects, *Bull. Seism.Soc.Am.*, 72, s225-s242.

Anderson, J. and H. Swanger (1990). Center for Seismic Studies Version 3 Database: Sql Tutorial.

Anderson, J., Farrell, W.E., Garcia, K., Given, J., H. Swanger (1990). Center. for Seismic Studies Version 3 Database: Schema Reference Manual.

Bratt, S.R. and T.C. Bache (1998). Locating events with a sparse network of regional arrays, *Bull. Seism.Soc.Am.*, 78, 780-798.

Carr, D., Young, C., Harris, J., Aster, R., X. Zhang (1999). Cluster Analysis for CTBT Seismic Event Monitoring, *Seism. Res. Lett.*

Data, C. J. and H. Darwen (1993). A Guide to the SQL Standard, 3rd Edition, Addition-Wesley Publishing Company.

Elmasri, R. and S. B. Navathe (1994). Foundmentals of Database Systems, 2nd Edition, Addison-Wesley Publishing Company.

Harris, J. M. and C. J. Young (1997). MatSeis: a Seismic GUI and Tool-box for MATLAB. *Seism. Res. Lett, Vol 68, No.2,* 267-269.

Hartses, H. E. (1991). Simultaneous Hypocenter and Velocity Model Estimation Using Direct and Reflected Phases from Microearthquakes Recorded within the Central Rio Grande Rift, New Mexico, *Ph.D Dissertation, New Mexico Tech,* 251 pages.

Harvey, D. J. and D. M. Quinlan (1994). JSPC Projects and Information Products. *Eos, Transactions, American Geophysical Union, v. 75, n.44 Suppl.,* p. 68.

Koch, G. and K. Loney (1995). Oracle: The Complete Reference,3rd Edition, Oracle Press.

Mathworks (1993). Matlab-High Performance Numeric Computation and Visualization Software, First Printing.

Skov, M. (1994). Digital Seismic Data Acquisition and Processing as Applied to Seismic Networks in the Rio Grande Rift and Mount Erebus, Antarctica, *M.S. Independent Study, New Mexico Tech,*. 97 pages.

Withers, M. (1995). LWCEDS user's manual.

Withers, M. (1997). An Automated Local/Regional Seismic Event Detection And Location System Using Waveform Correlation, *Ph.D Dissertation, New Mexico Tech,* 198 pages.

Withers, M., Aster, R., Young, C., Beiriger, J., Harris, M., Moore, S. and J. Trujillo (1997). A Comparison of Select Trigger Algorithms for Automated Global Seismic Phase and Event Detection, *Bull.Seism.Soc.Am.* 88, pp. 95-106.

Withers, M., Aster, R. and C.Young (1999). An Automated Local and Regional Seismic Event Detection and Location System Using Waveform Correlation, *Bull.Seism.Soc.Am.* 89, 3, pp. 657-669.

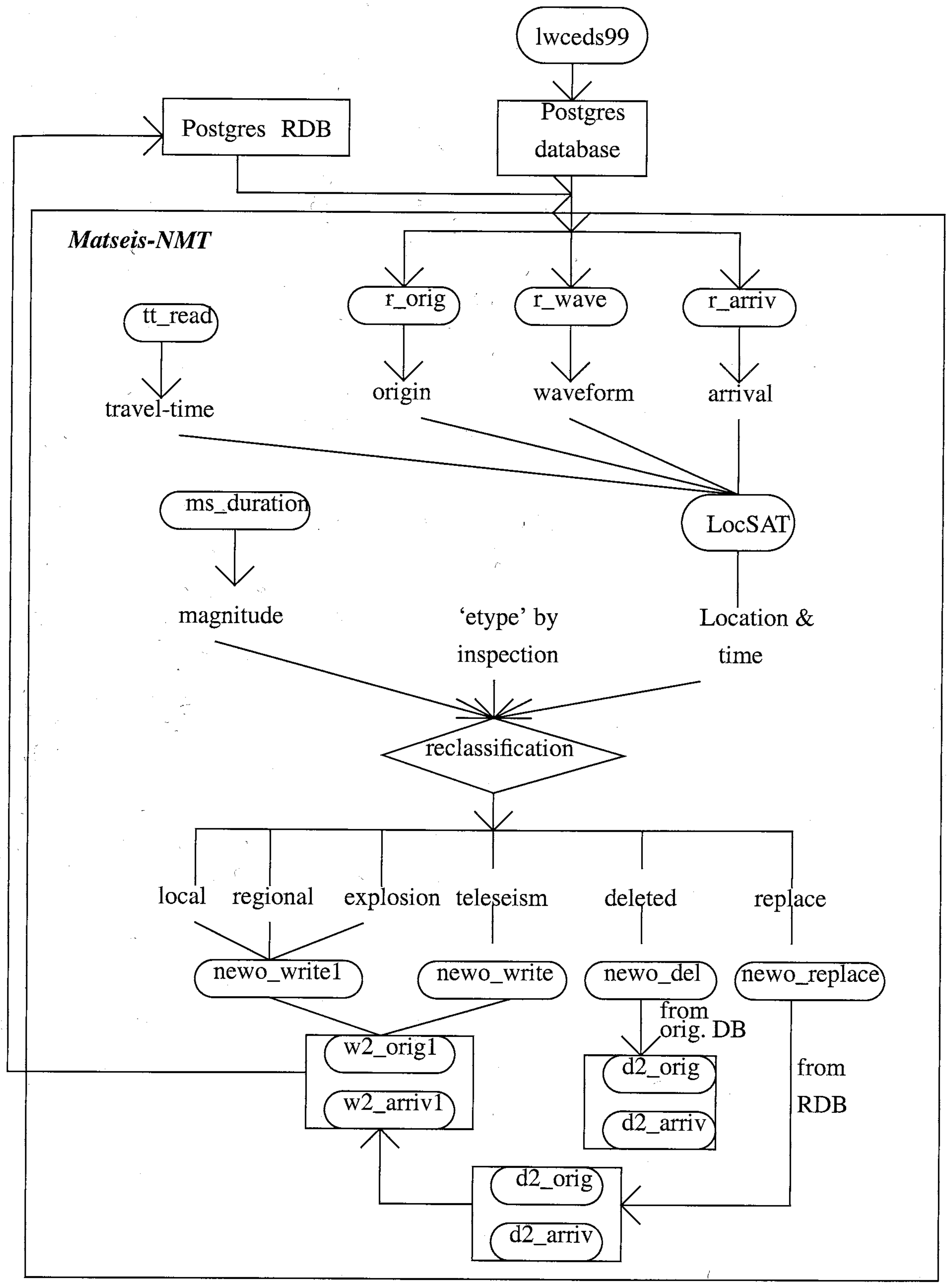
Yu, A. and J. Chen (1995). The Postgres95 User Manual, Version 1.0.

Appendix 1. Script 'lwceds99'

```
#!/bin/csh

setenv LWCEDSHOME /raid2/users/digqks/LWCEDS-1.1

set basedir = /data2/digqks

setenv LWCEDSPAR $basedir/LWCEDS.par

set trigdir = $basedir/data_incl/bak2

set parfile = $basedir/LWCEDS.par

set Template = $basedir/Template/template

setenv REVIEWHOME /raid2/users/digqks/Review

set reviewparfile=$basedir/review99.par

/bin/touch $basedir/digqks.logfile

set upcdir = $basedir/CSS3.0/1997/data

unalias cd
cd $trigdir

set nonomatch
if (`/bin/ls | wc -l` == 1) then
   @ numwvm = 0
else
   @ numwvm = `/bin/ls *.wvm | wc -l` >& /dev/null
endif

if ( $numwvm == 0 ) then
 exit 0
endif

set days = (`/bin/ls *.wvm | cut -c1-6 | sort | uniq`)
foreach i (`echo $days`)

 set UPCdir = $basedir/$i.UPc

 /bin/mkdir $UPCdir

 set files = (`ls $i??.wvm`)
 foreach j ($files)

  cd $trigdir

  set size1 = 0
  set size2 = 999999999999
  while ( $size1 != $size2)
  set size1 = `ls -l $j | awk '{print $4}'`
  sleep 10
  set size2 = `ls -l $j | awk '{print $4}'`
end

set eventdir = $UPCdir/`basename $j .wvm`.upc

/bin/mkdir $eventdir

set AhDir = $eventdir/ahfiles
/bin/mkdir $AhDir
set SUDSdir = $eventdir/wvmfiles
/bin/mkdir $SUDSdir
set DBdir = $eventdir/CSS
/bin/mkdir $DBdir

setenv AH_DIR $AhDir
/bin/mv $j $AhDir/$j

cd $AhDir
/raid1/local/bin/pcx2ah $j >& /dev/null
/usr/bin/compress $j > /dev/null
/bin/mv $j.Z $SUDSdir

cd $eventdir
touch lwcedslog
touch event.dat

cp $Template.lastid.0 $DBdir/upc.lastid
chmod u+w $DBdir/upc.lastid
touch $DBdir/upc.origin
touch $DBdir/upc.origerr
touch $DBdir/upc.netmag
touch $DBdir/upc.arrival
touch $DBdir/upc.assoc
touch $DBdir/upc.wfdisc
ln -s $Template $DBdir/upc
ln -s $Template.wfdisc.sensor $DBdir/upc.wfdisc.sensor
ln -s $Template.affiliation $DBdir/upc.affiliation
ln -s $Template.instrument $DBdir/upc.instrument
ln -s $Template.network $DBdir/upc.network
ln -s $Template.sensor $DBdir/upc.sensor
ln -s $Template.site $DBdir/upc.site
ln -s $Template.sitechan $DBdir/upc.sitechan

foreach trigger (`ls -d $AhDir/*`)

  @ dirsize = `echo $trigger | wc -c`
  if( $dirsize > 64 ) then
    exit -1
  endif
  set trigger = $trigger/

  $LWCEDSHOME/PreProc/PreProc par=$parfile WfdiscPath=$trigger \
  >> lwcedslog

 $LWCEDSHOME/Detector/Detector par=$parfile WfdiscPath=$trigger \
  >> lwcedslog

 $LWCEDSHOME/Locator/Locator par=$parfile WfdiscPath=$trigger \
  >> lwcedslog

 $LWCEDSHOME/Magnitude/GetMag par=$parfile WfdiscPath=$trigger \
  >> lwcedslog

 cat $LWCEDSHOME/fine.event.dat >> event.dat

  $LWCEDSHOME/DBLink/PutDB par=$parfile WfdiscPath=$trigger \
  DBPath=$DBdir/upc \
  >> lwcedslog

 sed -e 's/.Proc/"/g' $DBdir/upc.wfdisc >$DBdir/upc.wfdisc1
 sed -e 's/.......UPc..........upc/CSS3.0V1997Vdata/' \
 $DBdir/upc.wfdisc1 > $DBdir/upc.wfdisc
end

  setenv InDBPath $eventdir/CSS
  set DByear = 19`echo $1 | cut -c1-2 `
  setenv OutDBPath $basedir/CSS3.0/1997

  if (! -e $OutDBPath) then
    mkdir $OutDBPath
    /bin/cp $Template.lastid.0 $OutDBPath/socorro.lastid
    chmod u+w $OutDBPath/socorro.lastid
    touch $OutDBPath/socorro.origin
    touch $OutDBPath/socorro.origerr
    touch $OutDBPath/socorro.netmag
    touch $OutDBPath/socorro.arrival
    touch $OutDBPath/socorro.assoc
    touch $OutDBPath/socorro.wfdisc

    if ! -e $OutDBPath/socorro ln -s $Template $OutDBPath/socorro

      foreach tbl (wfdisc.sensor affiliation instrument network sensor site sitechan )
        ln -s $Template.$tbl $OutDBPath/socorro.$tbl
      end
    mkdir $OutDBPath/data
    mkdir $OutDBPath/data/wvmfiles
    mkdir $OutDBPath/data/ahfiles
  endif

  set Sudsdir = $OutDBPath/data/wvmfiles
   set AHdir = $OutDBPath/data/ahfiles

  /bin/cp $Template.lastid.0 $InDBPath/tmp1.lastid
  chmod u+w $InDBPath/tmp1.lastid
  touch $InDBPath/tmp1.origin
```

```
   touch $InDBPath/tmp1.origerr
   touch $InDBPath/tmp1.netmag
   touch $InDBPath/tmp1.arrival
   touch $InDBPath/tmp1.assoc
   touch $InDBPath/tmp1.wfdisc

   if ! -e $InDBPath/tmp1 ln -s $Template $InDBPath/tmp1

     foreach tbl (wfdisc.sensor affiliation instrument network sensor site
sitechan )
     if ( ! -e $InDBPath/tmp1.$tbl ) ln -s $Template.$tbl $InDBPath/tmp1.$tbl
     end

   $REVIEWHOME/review99 par=$reviewparfile
   /bin/cp $OutDBPath/socorro.lastid $InDBPath/tmp1.lastid

   foreach tbl (arrival assoc lastid netmag  origerr origin wfdisc)
    /raid2/users/digqks/bin/psqlprep $InDBPath/tmp1.$tbl > $InDBPath/tmp1filt.$tbl
   end
   echo \\i $InDBPath/tmprunfile | pgsql socorro_9708
    set numfromsuds = `ls $eventdir/wvmfiles/*.wvm.Z | wc -l`
    /bin/cp $eventdir/wvmfiles/* $Sudsdir
     /bin/cp -r $eventdir/ahfiles/* $AHdir
           else
           endif
         else
         endif
        end
        cd $basedir
       /bin/rm -r $UPCdir
        cd $trigdir
     end      # end loop over i events selected from day.upc
```

Appendix 2: Program Flow Diagram of Matseis-NMT



Appendix 3. Review process of an event.

1. From the 'File' menu, select 'Database Setup', in the 'setup' popup, select 'CSS 3.0 Postgres' and click on 'Apply'. In the 'PSQL Command Path' popup, key in the name of the database. and 'Apply'.

2. Click on the 'Origin' menu. This menu is for you to read data into Matseis. Key in 'Start time' and 'stop time' in the 'Origin Read' Popup, then 'Update'. After the origin ids show, select the orid you want. There are 5 choices for the 'Options' option, select what you want. 'Add' allows you to add new origins and 'Replace' allows you to clear out the previous origins and read in the new origins you want to use. 'Read Arrivals' allows you to read 'arrival' table of an event, 'Read Waveforms' reads from 'wfdisc' table. 'Zoom to Origins' make the event displayed at the origin time. Click on 'Apply' after you made your options.

3. If the travel time curves don't show, read in travel-time curves from the 'Travel-time' menu. Read in the phase that you want to use to locate the event. Select 'Regional' if the event looks like a regional event, 'local' if the event is a local event.

4. Move the arrivals in the window where the waveforms and phases are displayed if necessary. Use the 'Measure-Tool' menu to give standard deviation to the picks you decide.

5. Use 'Location' menu for relocating the event. In 'LocSAT Tool' popup, click on 'Arrivals' button, then in the 'LocSAT Arrival Edit' popup, select the phases you want to use, and 'Apply', check if every other parameter is given correctly in the 'LocSAT Tool' GUI. Then click on 'Apply'.

6. Click on the 'Magnitude->Duration' menu for the magnitude of the event. In the 'Duration Magnitude' popup, select the arrival you want to use and click on 'Apply'.

7. By now reviewing the event is finished. Click on the 'Reclassify' button, select what you decide the event is. If it is noise select 'Deleted'. If it is'local select 'local' and if it is regional, select 'regional' and if it is 'explosion', select 'Explosion', and 'Teleseism' is for a teleseismic event. 'Replace' is for an event that is read into Matseis-NMT from a reviewed database, after re-review, the event is replaced by the new result in the reviewed database. After this procedure the event is written to the Postgres reviewed database and the review process is finished.