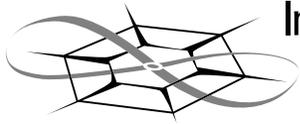


**The University of Kansas**



**Information and  
Telecommunication  
Technology Center**

Technical Report

**Database and Web Application to View  
Echograms and Ice Sheet Thickness Plots of  
Greenland Ice Sheet Data**

Subhajyoti Paul

ITTC-FY2005-TR-27640-06

January 2005

Project Sponsors:  
National Science Foundation  
Grant #OPP-0122520  
NASA Grants #NAG5-12659 and #NAG5-12980

Copyright © 2005:  
The University of Kansas  
2335 Irving Hill Road, Lawrence, KS 66045-7612  
All rights reserved.

## ACKNOWLEDGEMENTS

This master's project would not have materialized without the support and assistance of several people. First and foremost, I express my deep gratitude to Dr. Prasad Gogineni, my thesis advisor and committee chair, for giving me the opportunity to work on this project. Thanks to Dr. David Andrews and Dr. John Gauch for serving on my thesis committee. I am deeply indebted to Mr. Torry Akins, my project supervisor, for conceiving the idea behind this project. Thanks also go to Mr. John Paden, whose insightful solutions bailed me out whenever I was in a quandary. Thanks to my friend Atul for reviewing the initial draft of this document.

Last, but not least, I thank my parents and my siblings for their support and encouragement throughout the course of my stay at KU.

This work was supported by the National Science Foundation (grant OPP-0122520) and NASA (grants NAG5-12659 and NAG5-12980).

## ABSTRACT

Radar depth sounder data collected as part of the Program in Arctic Regional Climate Assessment (PARCA) are used to analyze bedrock and ice sheet conditions in the polar regions. As the size of the total data collected is unmanageable as a single file, the data are broken down into smaller files. MATLAB programs and tools are developed to extract information from these files one at a time and present them in the form of images, thickness plots and contour maps. The problem with this method is that it is difficult to isolate information pertaining to a certain search criterion across multiple files.

This project explores a new approach for viewing the radar data files. The first part of this project aims at loading selective data from data files into database tables. Only enough information from each data file is stored in the database as would be required to conduct searches on the data files later; the raw data are still stored in the original files because it is impractical to store all of the raw data in the database due to its size. The second part of this project is the development of a Mex interface to MATLAB (written in C) to conduct queries on the database. Results from the database queries are used to create dynamic web pages for the target audience – scientists, geologists and educators. The web interface designed as a part of this project seeks input from the user from a web browser. The user input is passed to a background MATLAB process for further processing. The final output appears in the web browser in the form of MATLAB plots and images. A plot of flight lines corresponding to the data is provided to the user to aid in visualization. Buttons are provided to generate echograms and thickness plots corresponding to the area covered by the flight lines.

## TABLE OF CONTENTS

1. Introduction.....	1
1.1 Motivation.....	3
1.2 Goals.....	3
1.2.1 Database loader.....	3
1.2.2 MATLAB mex interface.....	4
1.2.3 Web Interface.....	4
1.3 Software Requirements.....	4
1.3.1 Software for the developer.....	4
1.3.2 Software for the end user.....	5
1.4 Approach.....	5
2. Related Work.....	6
2.1 Format of coherent radar depth sounder data file.....	6
2.2 Data types in files.....	9
3. Overall System Design.....	10
3.1 System architecture.....	10
3.1.1 Database loader.....	11
3.1.2 Mex – interface.....	12
3.1.3 Web interfaces.....	12
4. Database Design and Implementation.....	13
4.1 Directory structure of data files.....	13
4.1 Entity-Relationship (ER) diagram.....	14
4.2 Database contents.....	15
4.2.1 Description of FileInfo table entries.....	15
4.2.2 Description of Header table entries.....	15
4.2.3 Description of GPS table entries.....	16
4.2.4 Description of TopBottomData table entries.....	17
4.3 Design issues – Transaction safe or non-transaction safe tables.....	17
4.4 Loader program.....	19
5. Mex interface.....	21
6. Web Interface.....	26
6.1 MATLAB operation on the web.....	28
6.2 Layout of the html form.....	29
6.3 Files used for web interfaces.....	30
6.4 Signal Processing.....	31
7. Experiment results and validation.....	33
7.1 Test Case 1:.....	33
7.2 Test Case 2:.....	36
7.3 Test case 3:.....	39
CONCLUSIONS AND FUTURE WORK.....	40
Recommendations.....	41
APPENDIX A.....	43
A.1 Header.....	43

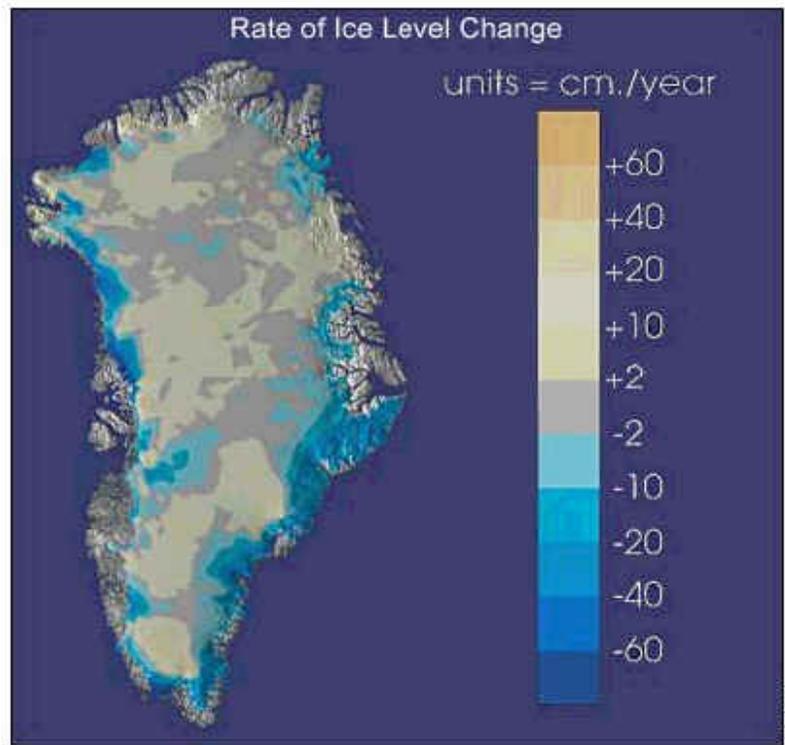
A.2 FileInfo.....	43
A.3 GPS .....	44
A.4 TopBottom .....	45
APPENDIX B .....	46
Configuration and Setup Notes.....	46
B.1 Configuring Matlab Web Server .....	46
B.2 Software Setup .....	47
REFERENCES .....	49

## TABLE OF FIGURES

Fig 1 – Rapid thinning of Greenland’s Coastal ice ([3]) .....	2
Fig 2 – Overall system design.....	11
Fig 3 – Data file archive structure.....	13
Fig 4 – ER diagram of CoRDS file database .....	14
Fig 5 – Contents of table FileInfo.....	15
Fig 6 – Partial contents of table Header.....	16
Fig 7 – Partial contents of table GPS .....	16
Fig 8.– Partial contents of table TopBottomData .....	17
Fig 9 – Flowchart of operation of the loader application.....	19
Fig 10 – Operation of mex file.....	22
Fig 11 – result of select query within mex function .....	24
Fig 12 –MATLAB webserver and httpd running on the same machine.....	27
Fig 13 - MATLAB web server and httpd running on different machines .....	27
Fig 14 – Functioning of MATLAB web server [7].....	28
Fig 15 – Output of submit button.....	29
Fig 16 – Output of search triggered by user .....	33
Fig 17 – Echogram for surface covered by flight path 2 .....	34
Fig 18 – Echogram generated manually for comparison (has more records).....	34
Fig 19 – Thickness plot for surface covered by flight path 2 .....	35
Fig 20 – Thickness plot generated manually (has more records) .....	36
Fig 21 – Output on main page.....	37
Fig 22 – Echogram generated by the application.....	38
Fig 23 – Echogram generated manually for comparison .....	38
Fig 24 – Echogram generated using first 7 files only .....	39
Fig 25 – Header table structure.....	43
Fig 26 – FileInfo table structure.....	44
Fig 27 – GPS table structure .....	44
Fig 28 – TopBottom table structure .....	45
Fig 29 – Options in appl.conf .....	47

# 1. Introduction

Average temperatures in the Arctic region are rising twice as fast as the global average [1]. Arctic ice is getting thinner, melting and rupturing [1]. If this trend continues, then summers in the Arctic could become ice-free in a few more decades. A warmer Arctic will affect weather patterns and food production around the world. For example, Kansas would be 4 degrees warmer in the winter without Arctic ice. This would have a profound effect on the wheat production in Kansas [1]. Apart from local flora and fauna, the effect of climbing global temperatures and melting land mass ice has far-reaching effects beyond the polar regions. A vast majority of the world's population live in low-lying coastal areas. With increasing global temperatures and the consequent rise in sea level, the lives of people living in the coastal regions will be affected. As such it is important for scientists to precisely identify the factors contributing to sea level rise. At present, there is a large uncertainty in the role of polar ice sheets in the global sea level rise [2]. To assess the role of the polar ice sheets in sea level rise, we will need to determine the mass balance of these ice sheets. One of the crucial measurements required to determine the mass balance of an ice sheet is the ice thickness.



**Fig 1 – Rapid thinning of Greenland’s Coastal ice ([3])**

The Radar Systems and Remote Sensing Lab (RSL) at the University of Kansas has been involved in the measurement of ice sheet thickness in Greenland since 1991. The data used for this project were collected by two coherent radar depth sounders operated in Greenland, each with a center frequency of 150 MHz. They are Improved Coherent Antarctic Radar Depth Sounder (ICARDS) and Next Generation Coherent Radar Depth Sounder (NGCoRDS) [4]. The data obtained by airborne radar measurements are processed to produce more comprehensible output such as radio echograms and thickness plots. The processed echograms are used by glaciologists all over the world to investigate possible reasons behind the alarming sea level rise in the past century.

Apart from data collection and analysis, there is also a need for a strong public outreach program that aims at disseminating information to the general public and

scientific community outside KU. The current work aims at making a contribution to this area.

## **1.1 Motivation**

One of the objectives of the Polar Radar for Ice Sheet Measurements (PRISM) project, funded by the National Science Foundation and NASA, is to make data collected from the polar regions available to the general public. As such, alliances have been formed with Haskell Indian Nations University and the Advanced Learning Technologies in Education Consortia (ALTEC) to make educational resources available to both students and educators alike. Data provided by PRISM field teams have to be analyzed and transformed into maps. Currently, interested parties can download data in the form of MATLAB data files (mat files). Portable Document Format (PDF) documents containing processed radio echograms are available for download too. This project explores a way to generate echograms and thickness plots on the fly and make the whole process more interactive.

## **1.2 Goals**

Following is a summary of the goals to be accomplished in this project. Radar data files are referred to as CoRDS files.

### **1.2.1 Database loader**

The first goal of this project is to test the feasibility of storing a subset of data from CoRDS files collected by airborne radar flights conducted by the NASA Wallops flight facility. This involves the design of a database and loading selected data from data files

into respective database tables. Information stored in the database should be self-sufficient to perform search operations at a later stage.

### **1.2.2 MATLAB mex interface**

The second stage is the development of a mex interface to query the data repository. MATLAB programs have already been written to read and extract data from binary data files. The mex file builds upon these programs. The central database, mex interface, and MATLAB programs work in tandem to return results corresponding to a search generated by users.

### **1.2.3 Web Interface**

The last and most tangible part of the project is the development of web interfaces to provide an online search facility. Users are prompted to specify search parameters (latitude and longitude). The final output visible to the users consists of thickness plots and echograms representing the bedrock conditions along the flight lines flown in the area of interest.

## **1.3 Software Requirements**

The following is a list of software required for this project.

### **1.3.1 Software for the developer**

All parts of the software application have been developed on a GNU/Linux system.

Following is an exhaustive list of software components used in this project.

1. C
2. Mysql (4.0.18 Standard)

3. HTML
4. JavaScript
5. MATLAB 7.0 (R14)
6. MATLAB Web Server
7. Apache Web Server
8. gcc

### **1.3.2 Software for the end user**

1. A graphical user interface (GUI) based web browser
2. JavaScript capability
3. Frames capability

## **1.4 Approach**

The approach taken is a combination of database, MATLAB, and web programming. MATLAB library functions are used for matrix operations. As the file sizes are enormous, it is not feasible to store the entire data set collected in the database. Instead, only the information necessary for searching (e.g., global positioning system tag, thickness, etc.) is stored in the database. Each radar data file contains many records (A-scopes). Only the necessary information from each record will have an entry in the database. The database allows quick retrieval of the record indices and corresponding file paths that satisfy a certain query operation. Using these record indices and file paths we can extract the corresponding records for further processing.

## 2. Related Work

This document would be incomplete without explaining the current framework that exists to view data from radar data files. What follows is a brief summary of the format and data types available in CoRDS files and the access mechanism. Much of the background content in this section is reprinted from the online CoRDS resource [5].

### 2.1 Format of Coherent Radar Depth Sounder Data File

These are the files generated by the data system of the airborne radar depth sounder. The radar depth sounder transmits a sequence of pulses from an aircraft flying over the ice sheet and coherently processes the received signal. The received signal is demodulated into its inphase (I) and quadrature (Q) components, which are sampled at 18.75 MHz. The operator sets the delay after which the system begins to sample. This delay is called the Sampling Window delay and is determined by the time taken by the transmitted pulse to make the trip from the transmitter to the ice surface and back. The number of samples is also set by the operator and is determined by the anticipated thickness of the ice sheets. Once the received signal is sampled, it is represented by the digital system as two vectors, I and Q. The I vector corresponds to the inphase channel, and the Q vector corresponds to the quadrature channel. Several pulses are averaged together to reduce the data rates. The number of samples averaged together is called the number of coherent integrations. The figure below illustrates how the coherent averaging is done for the two channels.

<b>I Channel Data</b>				
<b>I Channel Pulse 1</b>	<b>I Channel Pulse 2</b>	.....	<b>I Channel Pulse N</b>	<b>I channel averaged <math>I_{avg}</math></b>
Sample 1	Sample 1	.....	Sample 1	$\frac{1}{N^*} \sum_{p=1}^N I$
Sample 2	Sample 2	.....	Sample 2	
⋮	⋮	⋮	⋮	
Sample M*	Sample M	.....	Sample M	
<b>Q Channel Data</b>				
<b>Q Channel Pulse 1</b>	<b>Q Channel Pulse 2</b>	.....	<b>Q Channel Pulse N</b>	<b>Q channel averaged <math>Q_{avg}</math></b>
Sample 1	Sample 1	.....	Sample 1	$\sum_{i=1}^N I / N$
Sample 2	Sample 2	.....	Sample 2	
⋮	⋮	⋮	⋮	
Sample M	Sample M	.....	Sample M	

**Table 1 – Coherent averaging of samples**

**N\*** - Number of coherent integrations

**M\*** -Number of samples

**p** – Pulse no

The radar may operate in two modes – coherent and incoherent. When operating in coherent mode, the data system saves the averaged I and Q channel data to the hard disk.



(int32) datatype in position 3	(int32) datasize in position 3	(int32) Number of records in position 3
Position 3 data		
....		
(int32) datatype in position n	(int32) datasize in position n	(int32) Number of records in position n
Position n data		

**Table 2 – Data file format with repeated tagged data elements**

The first 64 bytes of every file contain header information. This header describes the radar settings during the time the data were collected. This is followed by a sequence of tagged records. Each record starts with a 12-byte tag followed immediately by the radar data (a single A-scope). The 12-byte tag is composed of three 4-byte fields:

- Data type
- Data size
- Number of records

## 2.2 Data Types in files

The figure below lists the possible data types in the binary files.

Datatype	Description	Format
1	Incoherent raw data	Uint16
2	I channel raw data (coherent)	Uint16
3	Q channel raw data (coherent)	Uint16
4	GPS string	Uint8
5	Time	Uint8
7..19	Reserved	
20	Top Curve	Float32
21	Bottom curve	Float32

**Table 3 – Data types in use**

## 3. Overall System Design

There are three main components in this project. They are:

1. **Database loader application:** The main purpose of this piece of software is to load selective information from CoRDS files to a central database. The Entity-Relationship and the data model diagrams are given later in this document.
2. **Mex interface:** This program acts as the intermediary between the web interface and the central database system. This takes as input the search parameters passed from the web search page. Subsequently, it forms a query based on the passed parameters and sends the query to the database, which responds with the corresponding result set. The results returned are further processed to isolate separate flight paths and their corresponding latitude and longitude values.
3. **Web Interface:** The primary purpose of the web interface is to interact with the end user. Latitude and longitude co-ordinates specified by the user are passed to the mex interface described above. Based on the information passed back by the mex file, buttons to create plots are created on the fly.

### 3.1 System Architecture

This section describes the functionality and interaction between the different components of the system designed for this project. Figure 2 diagrams the tasks performed by these components and how the flow of control and data takes place. The oval symbol represents a component and the rectangles represent an input/output to the system.

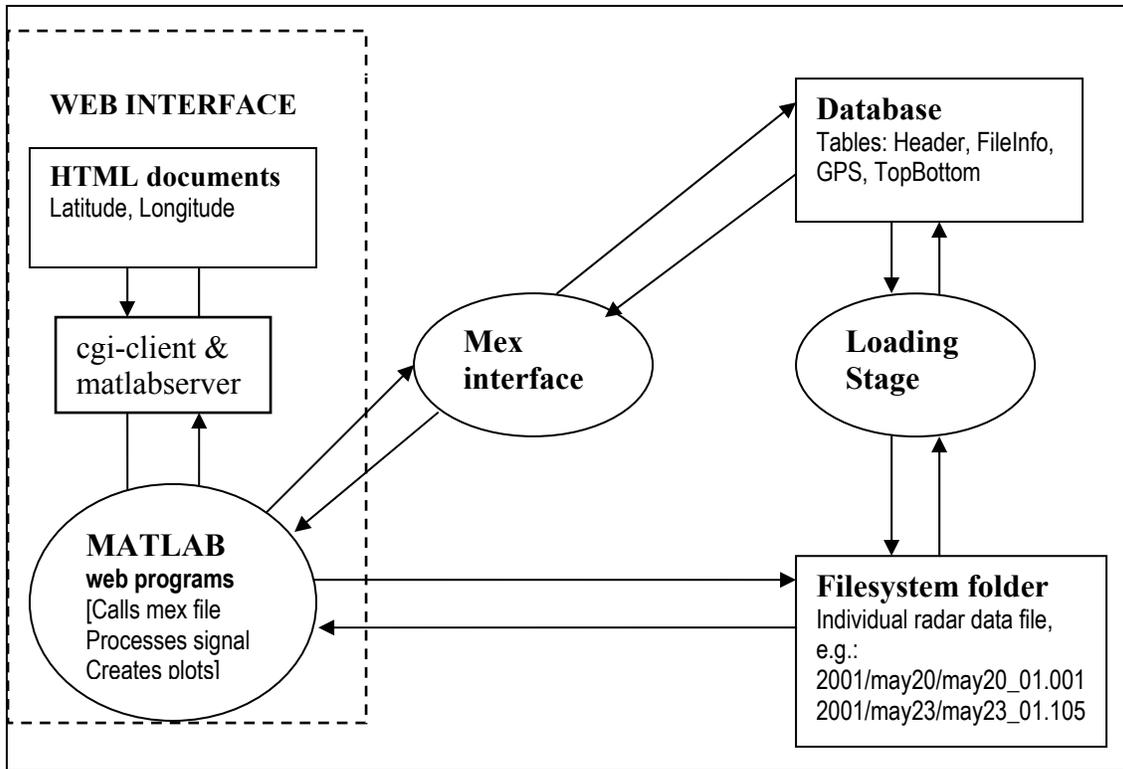


Fig 2 – Overall system design

### 3.1.1 Database loader

A database needs to be created before this stage is initiated. A detailed explanation of the database design follows in the next chapter. The database consists of four tables – Header, FileInfo, GPS, and TopBottomData – which hold the individual file header information, file indices, GPS data, and top and bottom curve information respectively. Data have not only to be consistent; they also have to be indexed in such a way that fast searching is made possible. The database loader is run from time to time to update the database with the latest CoRDS files.

### **3.1.2 Mex – interface**

This stage works in tandem with the web interfaces. Any time a user triggers a search, this stage comes into play. Once search information is received from the MATLAB cgi client, a MATLAB program calls this interface program. The purpose of this program is to query the database to find matching records, place some variables into the MATLAB's workspace, and then hand back control to the MATLAB process. The variables placed in the workspace are used to generate a flight line plot.

### **3.1.3 Web Interfaces**

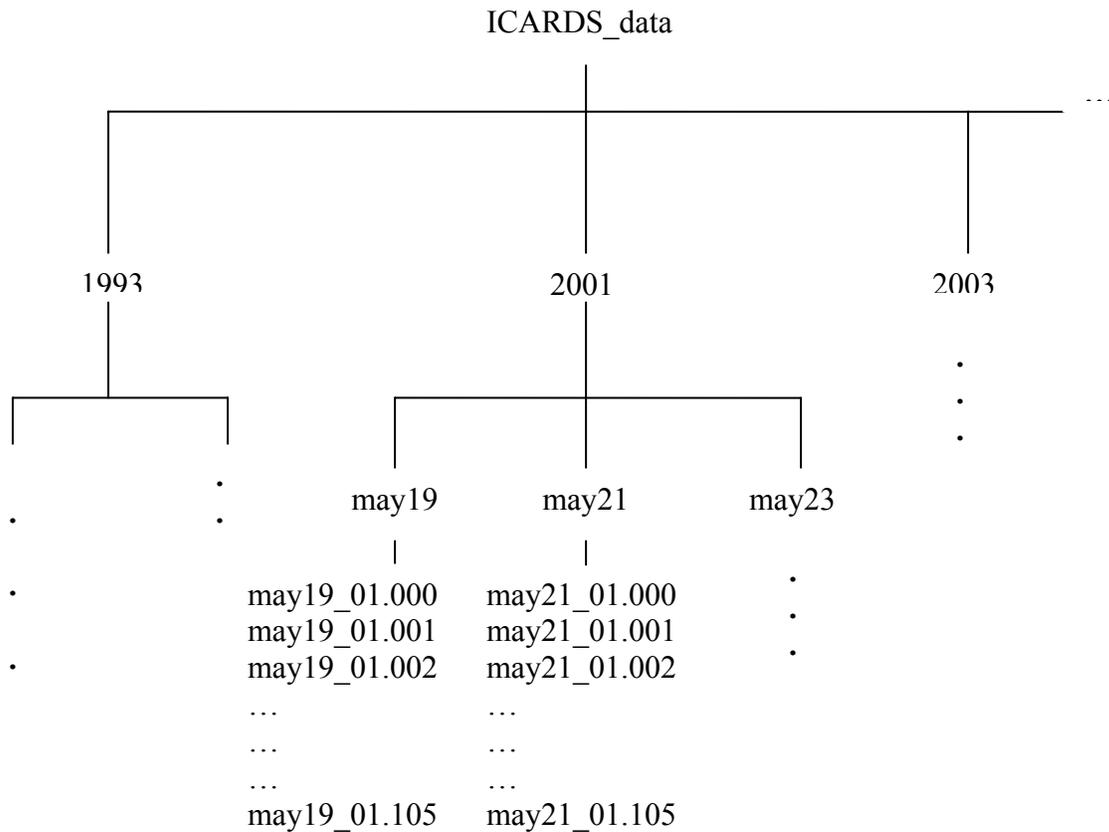
The mex interface passes a list of file names to the MATLAB workspace. This in turn is passed to the user's browser. This list is parsed to form a set of buttons. Clicking the buttons triggers the creation of echograms or thickness plots.

## 4.Database Design and Implementation

Selective information is extracted from data files and stored in the database. The data files are large in size. For example, the data for a single flight on May 23, 2001, over the polar regions generated 5.1 GB of data. These data are broken down into smaller files. The typical file size is between 31 – 40 MB.

### 4.1 Directory structure of data files

The directory structure where data files are stored is shown in the figure below.



**Fig 3 – Data file archive structure**

Files belonging to a particular year are stored relative to the directory called ICARDS\_data. Within each of the folders marked by years (1999, 2001, 2003, etc), is

another set of folders organized by dates (may23, may21, may19, etc). The data files are stored in these folders. The loader application should be run from the base folder ICARDS\_data. This ensures that the file paths stored in the database are always relative to the top-level directory, ICARDS\_data.

## 4.1 Entity-Relationship (ER) diagram

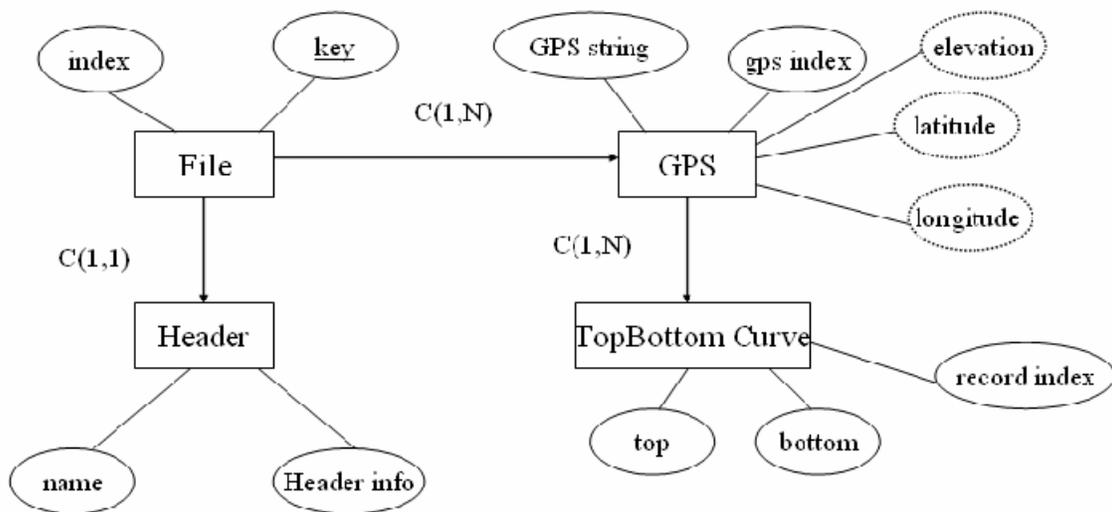


Fig 4 - ER diagram of CoRDS file database

The diagram above represents the overall logical structure of the database. Rectangles represent entity sets, ellipsis represent attributes and lines link attributes to entity sets or entity sets to relationships. For each entity and relationship set, there is a unique table, which is assigned the name of the corresponding set. Each table has a number of columns with unique names.

The entity set FILE has a one-to-one relationship with the HEADER entity set. FILE has a one-to-many relationship with GPS. GPS in turn has a one-to-many

relationship with TOP\_BOTTOM\_CURVE. Refer to Appendix A for a description of table fields and their corresponding data types.

## 4.2 Database contents

The following figures illustrate what the tables look like after data have been successfully loaded.

### 4.2.1 Description of FileInfo table entries

A snapshot of database entries for the table FileInfo is given below:

fileKey	fileName	fileIndex	nrecords
568	./2001/may20/may20_01.000	0	304
569	./2001/may20/may20_01.001	1	10000
570	./2001/may20/may20_01.002	2	10000
571	./2001/may20/may20_01.003	3	10000
572	./2001/may20/may20_01.004	4	10000
573	./2001/may20/may20_01.005	5	10000
574	./2001/may20/may20_01.006	6	10000
575	./2001/may20/may20_01.007	7	10000
576	./2001/may20/may20_01.008	8	10000
577	./2001/may20/may20_01.009	9	10000
578	./2001/may20/may20_01.010	10	3921
579	./2001/may20/may20_01.011	11	3267

**Fig 5 – Contents of table FileInfo**

The figure above displays the first several entries for CoRDS files recorded on May 20, 2001. fileKey is an auto increment column. fileIndex denotes the last three characters in the file name. The last column shows the number of ascopes/records in each of the individual files.

### 4.2.2 Description of Header table entries

The next figure displays the contents of the table Header.

fileKey	prf	swd	dspmode	nosamples	nocohintgr	noicohintgr	cards	dformat
568	9202	0	0	800	256	4	1	0
569	18409	0	0	800	256	4	1	0
570	18409	0	0	800	256	4	1	0
571	18409	0	0	800	256	4	1	0
572	18409	0	0	800	256	4	1	0
573	18409	0	0	800	256	4	1	0
574	18409	0	0	800	256	4	1	0
575	18409	0	0	800	256	4	1	0
576	18409	0	0	800	256	4	1	0
577	18409	0	0	800	256	4	1	0
578	18409	0	0	800	256	4	1	0
579	18409	0	0	800	256	16	1	0

Fig 6 – Partial contents of table Header

The above figure displays the information found in the header section of the files. Table Header is linked to table FileInfo through the column fileKey. Each row in FileInfo table will have exactly one corresponding row in table Header. This is due to the 1-1 correspondence between the two tables.

### 4.2.3 Description of GPS table entries

The figure below displays the contents of table GPS.

fileKey	gpsIndex	gps_str	UTC	lat	longd	elev
577	1	\$GPGGA,111711.50,6857.79727,N,03336.73175,W,1,11,00.8,+02168,M,,*61	111712	68.9633	33.6122	2168
577	2	\$GPGGA,111712.00,6857.83221,N,03336.78238,W,1,11,00.8,+02169,M,,*61	111712	68.9639	33.613	2169
577	3	\$GPGGA,111712.50,6857.86724,N,03336.83252,W,1,11,00.8,+02170,M,,*61	111712	68.9645	33.6139	2170
577	4	\$GPGGA,111713.00,6857.90236,N,03336.88218,W,1,11,00.8,+02171,M,,*60	111713	68.965	33.6147	2171
577	5	\$GPGGA,111713.50,6857.93757,N,03336.93140,W,1,11,00.8,+02173,M,,*62	111714	68.9656	33.6155	2173
577	6	\$GPGGA,111714.00,6857.97286,N,03336.98019,W,1,11,00.8,+02174,M,,*6C	111714	68.9662	33.6163	2174
577	7	\$GPGGA,111714.50,6858.00824,N,03337.02854,W,1,11,00.8,+02174,M,,*69	111714	68.9668	33.6171	2174
577	8	\$GPGGA,111715.00,6858.04372,N,03337.07639,W,1,11,00.8,+02175,M,,*60	111715	68.9674	33.6179	2175
577	9	\$GPGGA,111715.50,6858.07931,N,03337.12367,W,1,11,00.8,+02176,M,,*62	111716	68.968	33.6187	2176
577	10	\$GPGGA,111716.00,6858.11500,N,03337.17041,W,1,11,00.8,+02176,M,,*6F	111716	68.9686	33.6195	2176
577	11	\$GPGGA,111716.50,6858.15081,N,03337.21657,W,1,11,00.8,+02176,M,,*66	111716	68.9692	33.6203	2176
577	12	\$GPGGA,111717.00,6858.18672,N,03337.26214,W,1,11,00.8,+02177,M,,*60	111717	68.9698	33.621	2177

Fig 7 - Partial contents of table GPS

The above figure displays the contents of GPS table for the file may20\_01.009. Note that entries in this table are related to FileInfo table through the column fileKey.

Because of a one-to-many relationship, each row in the table `FileInfo` is associated with multiple entries in `GPS` table.

#### 4.2.4 Description of `TopBottomData` table entries

The next figure displays the contents of table `TopBottomData`.

<code>fileKey</code>	<code>gpsIndex</code>	<code>recordIndex</code>	<code>Top</code>	<code>Bottom</code>	<code>Thickness</code>
577	1	1	82.345734	302.731842	220.386108
577	1	2	82.347305	303.053864	220.706558
577	1	3	82.348885	303.372559	221.023674
577	1	4	82.350471	303.687927	221.337456
577	1	5	82.352058	304	221.647942
577	1	6	82.353645	304.308777	221.955132
577	1	7	82.35524	304.614319	222.259079
577	1	8	82.356842	304.916595	222.559753
577	1	9	82.358444	305.215668	222.857224
577	1	10	82.360054	305.511505	223.151451
577	1	11	82.361664	305.804169	223.442505
577	1	12	82.363281	306.093689	223.730408
577	1	13	82.364899	306.380035	224.015137

**Fig 8 – Partial contents of table `TopBottomData`**

`GPS` has a one-to-many relationship with `TopBottomData`. Hence, corresponding to the value pair (`fileKey=577`, `gpsIndex=1`) in `GPS`, you will find multiple rows in `TopBottom`. When data is collected, the `GPS` information remains the same over several pulses. Hence, the need for a one-to-many relationship between these two tables.

#### 4.3 Design Issues – Transaction safe or non-transaction safe tables

Mysql database server is the world’s most popular open source database. Mysql supports several storage engines to handle different table types. It includes both those that handle transaction safe tables and those that handle non-transaction safe tables.

1. MyISAM – handles non-transactional tables.
2. InnoDB – transaction safe tables.

The issue here is whether to choose a Transaction Safe Table (TST) or a Non-Transaction-Safe Table (NTST). TSTs have several advantages over NTSTs[6]:

1. They are safer, in the event of any power failure or any other unforeseen events that crash the server. Changes to NTSTs cannot be roll backed.
2. They use a Commit and Rollback feature to execute or ignore a transaction as per the user's needs.
3. They provide better concurrency for tables that get many updates.

NTSTs have advantages of their own, which are largely due to the absence of any transaction overhead:

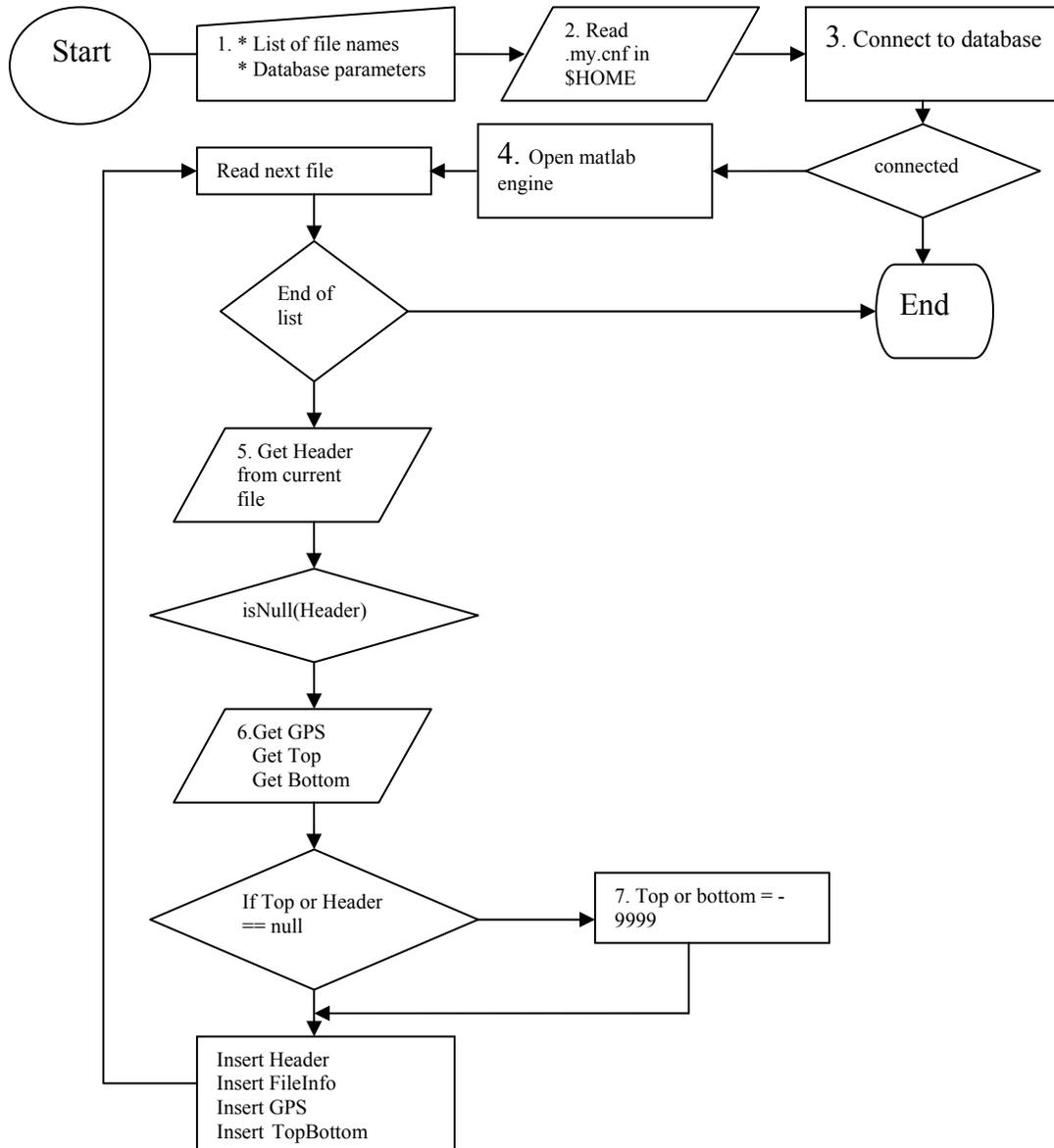
1. They are much faster than TSTs.
2. They consume less disk space.
3. They need less memory to perform updates.

In our application, data corresponding to 4 different data types is to be loaded into the database. What happens if the program successfully loads the first three data types but fails while loading the fourth data type because of some unhandled error or hardware failure? All the tables will contain a large volume of data. And our main motive is to conduct searches through these tables as fast as possible. Using InnoDB engine would hamper our goals. To make database search queries fast, MyISAM storage engine has been used to manage the tables. Utmost care is taken to handle all possible error conditions. Of course, in the event of power or hardware failure, we won't have a database transaction log to fall back on. All the update operations made up to the failure point will be permanent. An abrupt end of the application will leave the database in an

inconsistent state. The log file can be checked to see where the program termination took place and use that information to manually delete inconsistent data from the tables.

## 4.4 Loader Program

The diagram below illustrates the operation of the loader program.



**Fig 9 – Flowchart of operation of the loader application**

The various stages involved in this process are:

1. A list of file names and optional database parameters are passed as arguments to the program.

The usage is as follows.

```
CORDS_loader [-d database] [-u user] [-p[password]] [-h host] "list of file names"
```

The list of files names is specified using the “find” linux command. An actual example is given below. Note that the executable is to be run from the directory ICARDS\_data. If database parameters are not specified on the command line, then the program uses the database settings found in <home>/my.cnf file.

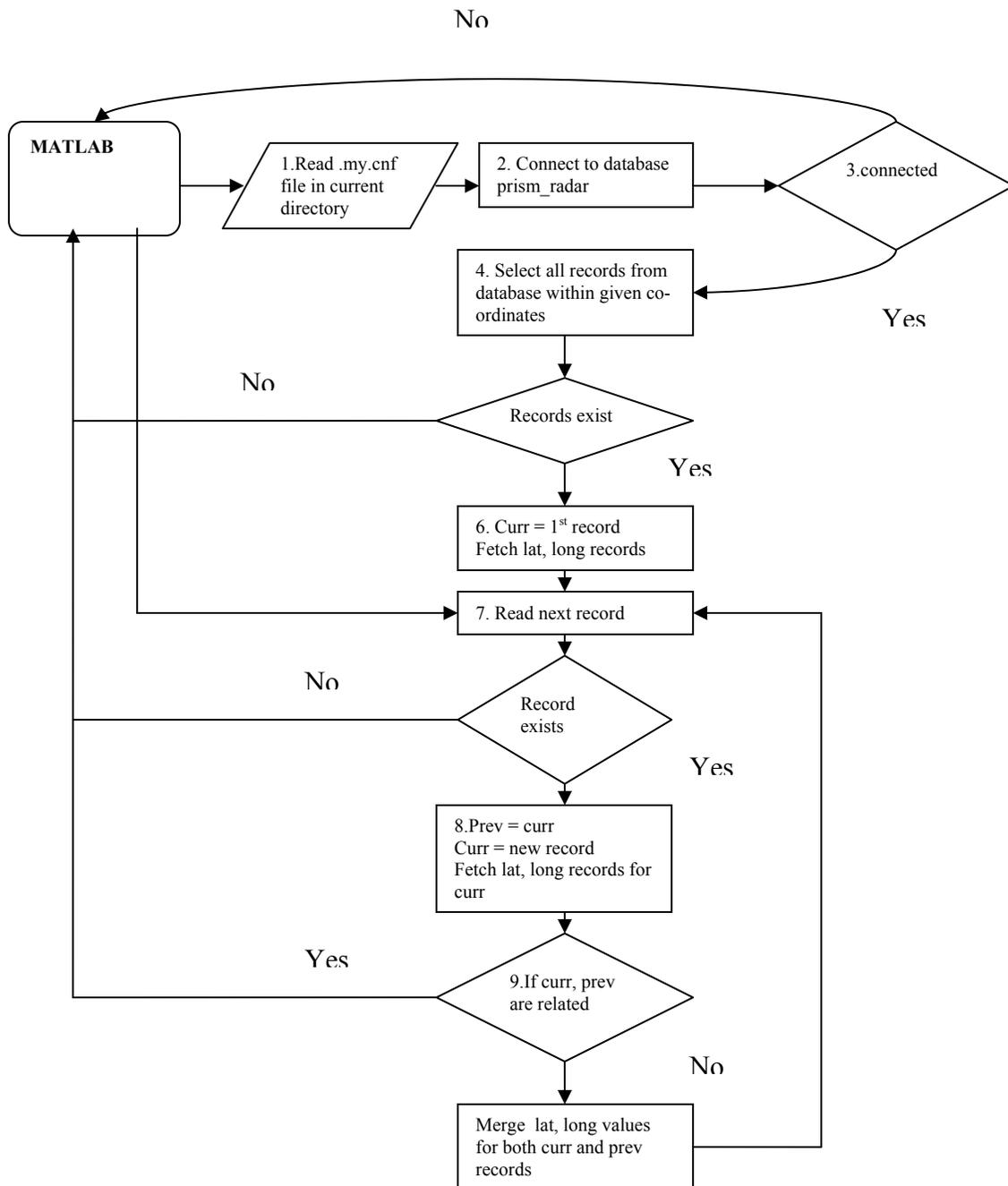
```
e.g. ncords_loader `find ./ -name `*.[0-9][0-9][0-9] -print |sort`
```

2. Read .my.cnf file in the current working directory. This file contains database parameters – server name, user name and database name. Password is to be passed from the command line for security reasons.
3. A connection to the backend database is made.
4. The MATLAB engine is opened. This helps us in calling MATLAB routines to be executed on the files passed as command line arguments.
5. File are read and processed sequentially. First the Header (datatype = 0) is read, followed by the GPS (datatype = 4), top (datatype = 20), and Bottom (datatype = 21).
6. A fileKey is generated at this point, which is used to index the files in the FileInfo, GPS, and TopBottomData tables.
7. If the Top or Bottom value is null, it is replaced by -9999 to represent invalid data.
8. Insert data into Header, FileInfo, GPS and TopBottomData tables.

## **5. Mex interface**

Mex files are subroutines produced from C or FORTRAN code. They can be called from within MATLAB just like any other M-file or built-in function. Mex files are dynamically linked subroutines that the MATLAB interpreter can directly load and execute. For our purpose, C language has been chosen to write the Mex files.

The main purpose of building a mex interface is to conduct queries on the database from within MATLAB. Based on the parameters passed to it, the program queries the database, loops through the records returned and generates a series of flight paths in the form of matrices. The output of this program is a structure of arrays which contain latitude and longitude values. The MATLAB instance, from within which the mex file is called, generates a plot of the flight lines based on the latitude and longitude values.



**Fig 10 – Operation of mex file**

The various stages involved are:

1. The mex file is called from within a MATLAB instance. The name of the mex file to be called is `form_paths`. The mex file first reads a file called `.my.cnf` in the

current user's home directory (\$HOME). This file contains data base parameters – database name, user name, database server and an optional password to connect to.

The mex file usage is shown below.

```
form_paths(min_lat, max_lat, min_longd, max_long, max_files)
```

“min\_lat” and “max\_lat” stand for the lower and upper limit of latitude values.

”min\_longd" and “max\_longd” stand for the lower and upper limit of longitude

values. The input argument “max\_files” is used to determine the maximum

number of samples available within the first N files for a flight line, where N

denotes the value of “max\_files.”

2. A connection to the database is established.
3. If a connection is not successfully established, return to MATLAB workspace with error message.
4. Select records from the database which fall under the given co-ordinates, i.e., latitude and longitude range. The following sql query is issued.

```
“select T.fileKey, F.fileName, F.fileIndex, F.nrecords, count (*), MIN(T.recordIndex),  
H.nosamples from TopBottomData T inner join FileInfo on F.fileKey = T.fileKey inner join  
GPS G on (G.gpsIndex = T.gpsIndex and G.fileKey = T.fileKey) inner join Header on  
(H.fileKey = F.fileKey)where (G.lat > x and G.lat < y) and (G.longd > a and G.longd < b)  
group by T.fileKey.”
```

where x, y , a and b form the latitude and longitude range.

Say the search values are:

Minimum latitude – 67°N      Maximum latitude – 69°N

Minimum longitude – 34°W      Maximum longitude – 37°W

The result of the search query at this stage is displayed below.

fileKey	fileName	fileIndex	nrecords	count(*)	MIN(T.recordIndex)	nosamples
611	./2001/may20/may20_01.023	23	10000	857	9144	800
612	./2001/may20/may20_01.024	24	9432	9432	1	800
613	./2001/may20/may20_01.025	25	369	369	1	800
614	./2001/may20/may20_01.026	26	10000	10000	1	800
615	./2001/may20/may20_01.027	27	10000	1094	1	800
645	./2001/may20/may20_01.057	57	10000	7649	2352	800
646	./2001/may20/may20_01.058	58	10000	10000	1	800
647	./2001/may20/may20_01.059	59	10000	1408	1	800
698	./2001/may21/may21_01.004	4	10000	2652	7349	800
699	./2001/may21/may21_01.005	5	10000	10000	1	800
700	./2001/may21/may21_01.006	6	10000	10000	1	800
701	./2001/may21/may21_01.007	7	10000	10000	1	800
702	./2001/may21/may21_01.008	8	10000	10000	1	800
703	./2001/may21/may21_01.009	9	10000	10000	1	800
704	./2001/may21/may21_01.010	10	10000	10000	1	800
705	./2001/may21/may21_01.011	11	10000	10000	1	800
706	./2001/may21/may21_01.012	12	10000	5591	1	800

**Fig 11 – Result of select query within mex function**

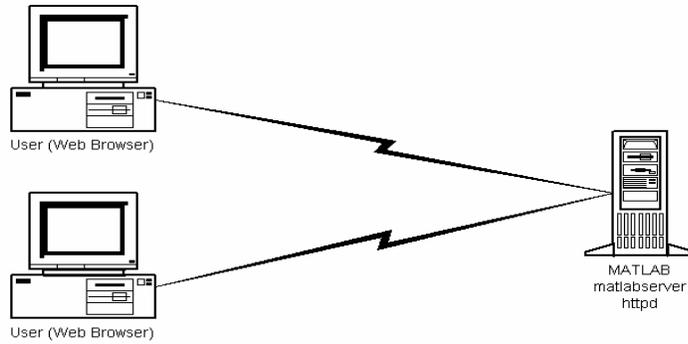
The rows can be processed to break the result set into three flight paths. Observe the fileKey column. There is a jump from fileKey 615 to 645. So, this is identified as the boundary between two logical flight paths even though both the files ./2001/may20\_01.027 and ./2001/may20/may20\_01.057 belong physically to the same flight. Another boundary is identified between the pair of fileKeys 647 and 698. The field count (\*) gives the total number of records within each file that satisfy the search query.

5. The latitude and longitude values for all files that fall within a flight path are concatenated and placed in the MATLAB workspace as a matrix variable.
6. A string variable is also placed in the MATLAB workspace. This string can be parsed to get the filenames belonging to a particular flight path. The string also holds the starting record index and the number of records selected per file. This string is used by another MATLAB program to create a set of buttons to be displayed to the user.

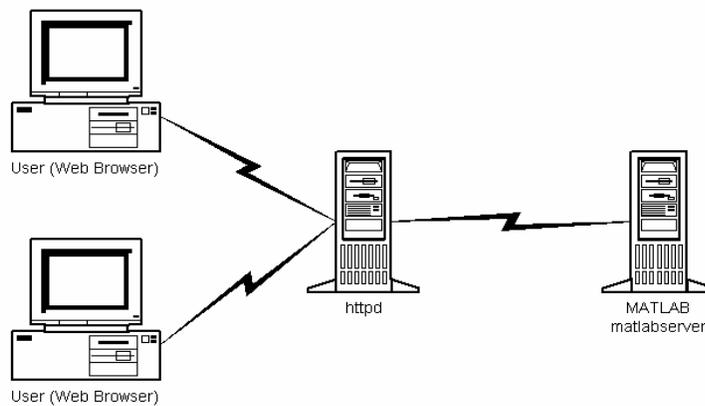
## 6. Web Interface

The web interface has been designed using a combination of HTML and JavaScript. The input form on the search page seeks four inputs from the user - minimum and maximum values of latitude and minimum and maximum values of longitude. These values are then passed to a MATLAB thread via a cgi client. The MATLAB instance processes the input and makes a call to the mex file meant to return flight paths. The mex file returns a structure of arrays from which a plot is generated. The mex file also passes information about the individual file names involved and the starting and ending record indices within the individual files.

An important component in the whole process is the MATLAB web server. The MATLAB Web Server is a toolbox, which is a web-front-end for MATLAB. The MATLAB web server allows us to create MATLAB applications that use the World Wide Web to send data to MATLAB for computation and to display the results in a web browser. A MATLAB web server component needs to be installed. The other two software requirements are the `matlab cgi client` on the web server and `httpd`. The MATLAB web server (`matlabserver`) and the web server daemon (`httpd`) can run on the same or separate machines. Both the configurations are shown below.



**Fig 12 – MATLAB webservice and httpd running on the same machine**



**Fig 13 – MATLAB web server and httpd running on different machines**

MATLAB web server applications are a combination of M-files, HTML and graphics.

The steps involved are as follows:

1. Create HTML documents for collection of input data and display of the results.
2. List the application name and associated configuration data in a file `matweb.conf`, which resides in the `cgi-bin` directory of the web server.
3. Write a MATLAB M-file that:
  - i. Analyzes input data entered in the HTML form
  - ii. Calls MATLAB functions of mex files for further processing

- iii. Creates images and plots if required
- iv. Places output data in a MATLAB structure
- v. Calls htmrep (MATLAB function) to place the output data in an HTML output document template

### 6.1 Matlab operation on the web

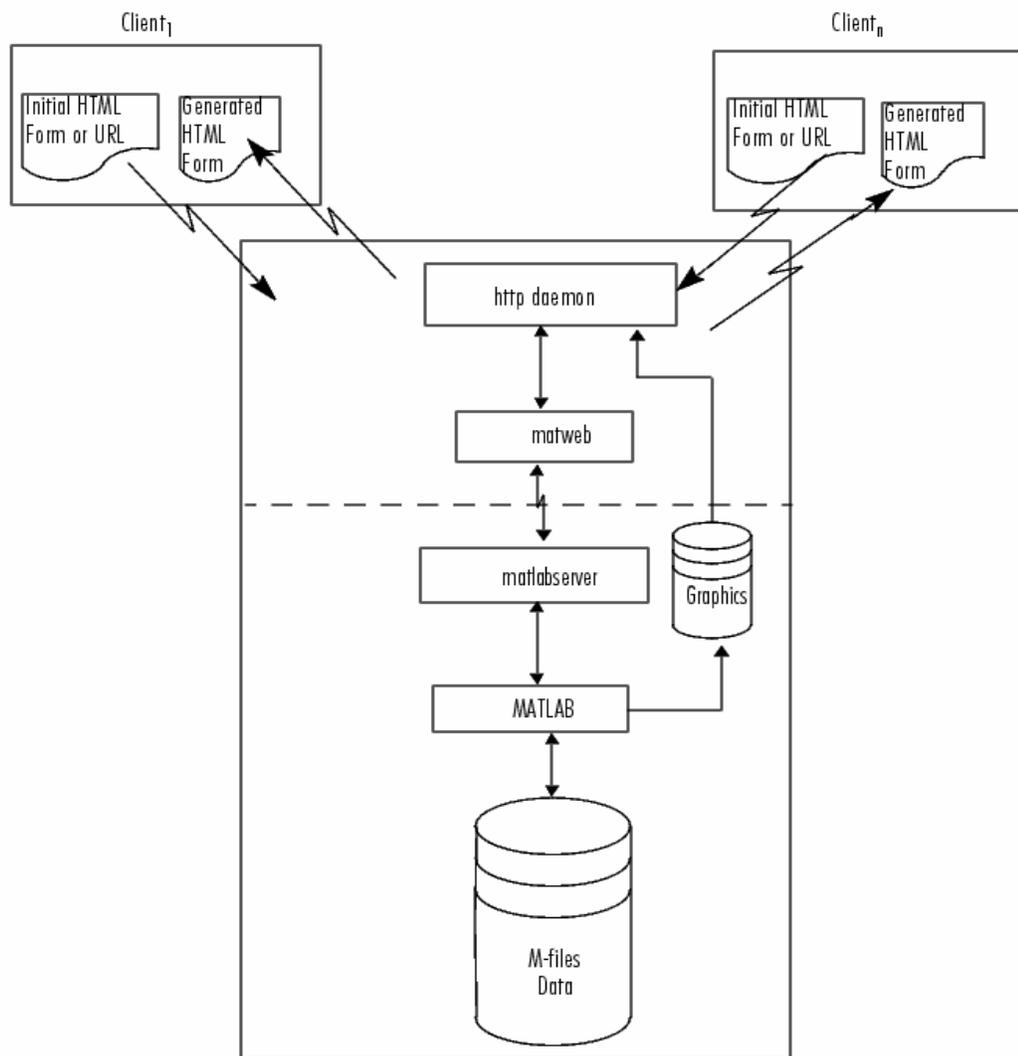
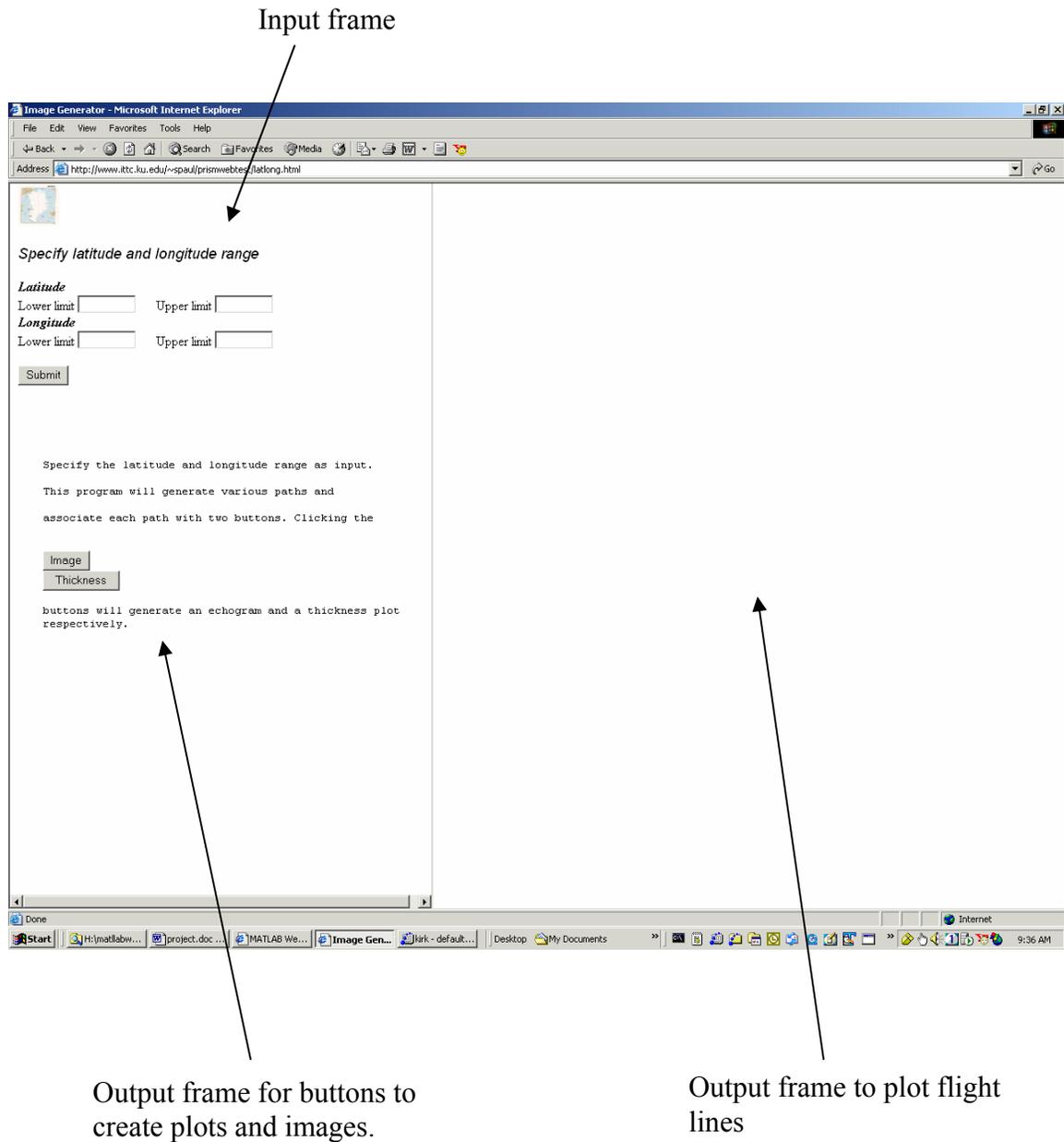


Fig 14 – Functioning of MATLAB web server [7]

## 6.2 Layout of the html form

The figure below shows the layout of the search html document.



**Fig 15 – Output of submit button**

The above snapshot of the browser window shows the main page. On the top left side lies the input frame. Here, the minimum and maximum values for latitude and longitude are

specified. Once the Submit button is clicked, a call to the MATLAB web server is made. The MATLAB web server runs on a remote machine where MATLAB is installed. The web server spawns several MATLAB threads based on configuration file entries. It hands over the input to one of the MATLAB threads, which starts the processing. The output html document is then sent back to the web browser to be displayed in the two output frames.

### **6.3 Files used for web interfaces**

What follows below is a discussion of the different MATLAB applications and html templates used in this project.

#### **1. gen\_flight\_lines.m**

This is the first program that is invoked when the user clicks on the submit button on the search page. This program accomplishes two purposes. First, it calls the mex file `form_paths` with appropriate arguments. Second, it plots the flight lines based on latitude and longitude matrices returned by `form_paths`. It then calls the template html file `prismweb.html` to create a dynamic web page to be sent back to the web browser. Once data is transferred back to the browser, the flight plot appears on the right frame and the image and thickness buttons appear on the bottom left frame.

#### **2. gen\_flight\_lines.html**

This is a template html file used to create a dynamic web page with buttons to generate echograms and thickness plots corresponding to flight lines plotted in `gen_flight_lines.m`.

#### **3.gen\_echogram.m**

This file is invoked when a user clicks on the  button in the output frame. The MATLAB function defined in this file generates echograms based on the input values

present in the form of which the image button is a member. Various input values passed are data file names, starting record indices, and end record indices.

#### **4. image\_echogram.html**

This template html file is used by gen\_echogram to embed the echogram generated in part 3 above.

#### **5.gen\_thickness.m**

This file is invoked when a user clicks on the  button in the output frame. The MATLAB function defined in this file generates a thickness plot of the ice sheet based on the input values present in the form of which the image button is a member. Various input values passed are data file names, starting record indices, and end record indices.

#### **6. image\_thickness.html**

This template html file is used by gen\_thickness.m to embed the thickness plot generated in part 5 above.

Input validation is done in the web browser using JavaScript.

### **6.4 Signal Processing**

The collected data is further processed. This has a two fold purpose. The first is to reduce the data size and the second is to enable more accurate measurements of ice sheet thickness. The m-file gen\_echogram.m performs the following signal processing steps:

1. First the dc offset is eliminated. This is done by deducting the mean value of each A-scope from each of the sample values in the A-scope.

2. 10 coherent averages are done. This reduces the data size by a factor of 10, which leads to faster execution. This also serves the purpose of reducing the random noise.
3. Finally, five incoherent averages are done before generating the echogram. This reduces the data size by a factor of five and reduces the effects of fading.

## 7. Experiment results and validation

### 7.1 Test Case 1:

In this case, the number of samples is same across all files. The number of samples recorded is typically 800 or 1024. This parameter is set by the operator during the time of data collection.

Search values: Minimum latitude – 64

Maximum latitude – 69

Minimum longitude – 34

Maximum longitude – 35

The above search parameters produced the following output.

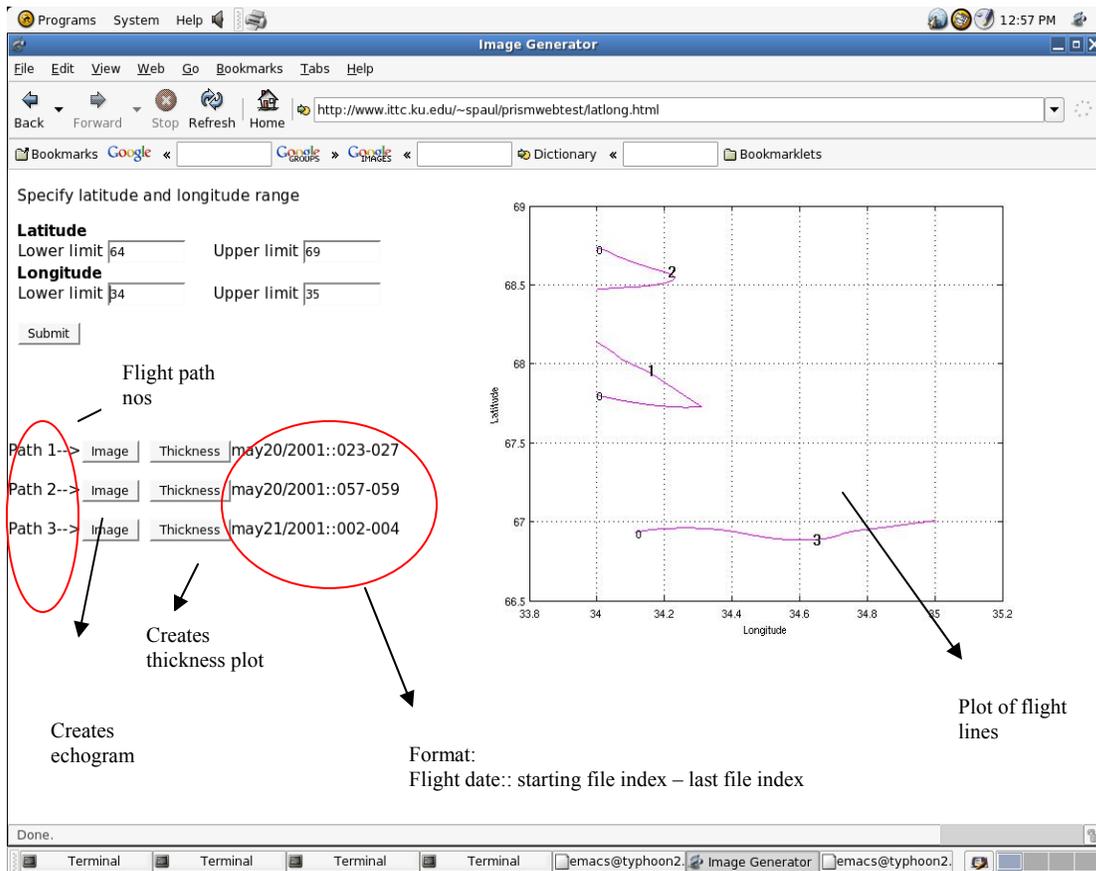
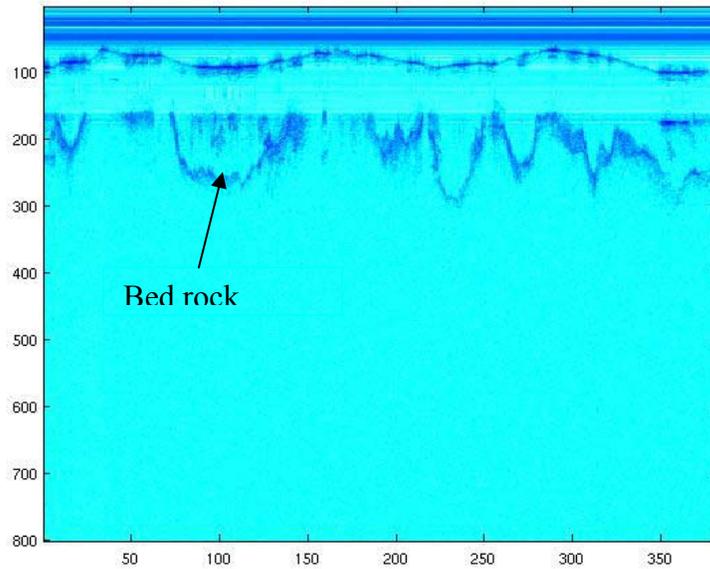


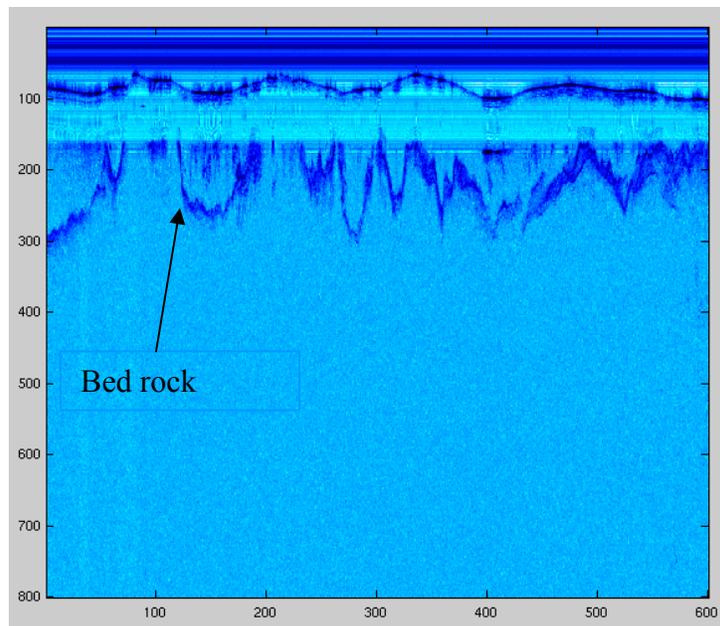
Fig 16 – Output of search triggered by user

Clicking on the image button corresponding to **Path 2** creates the radio echogram shown below.



**Fig 17 – Echogram for surface covered by flight path 2**

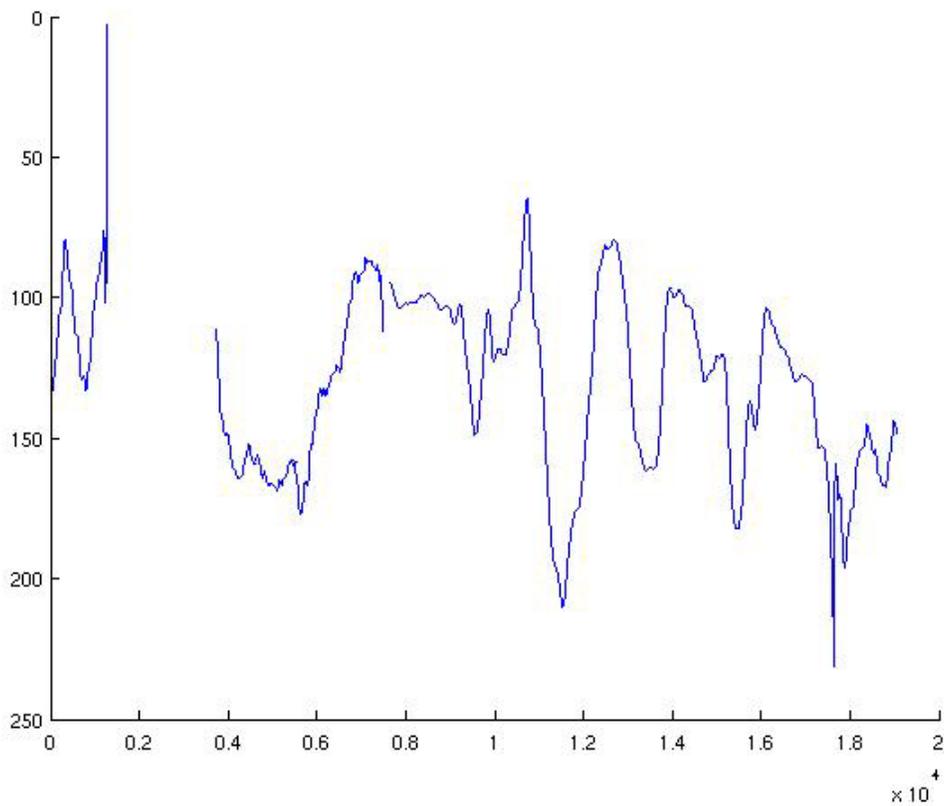
Based on the flight date and start and end file indices available from the output window, an echogram was generated manually in MATLAB.



**Fig 18 – Echogram generated manually for comparison (has more records)**

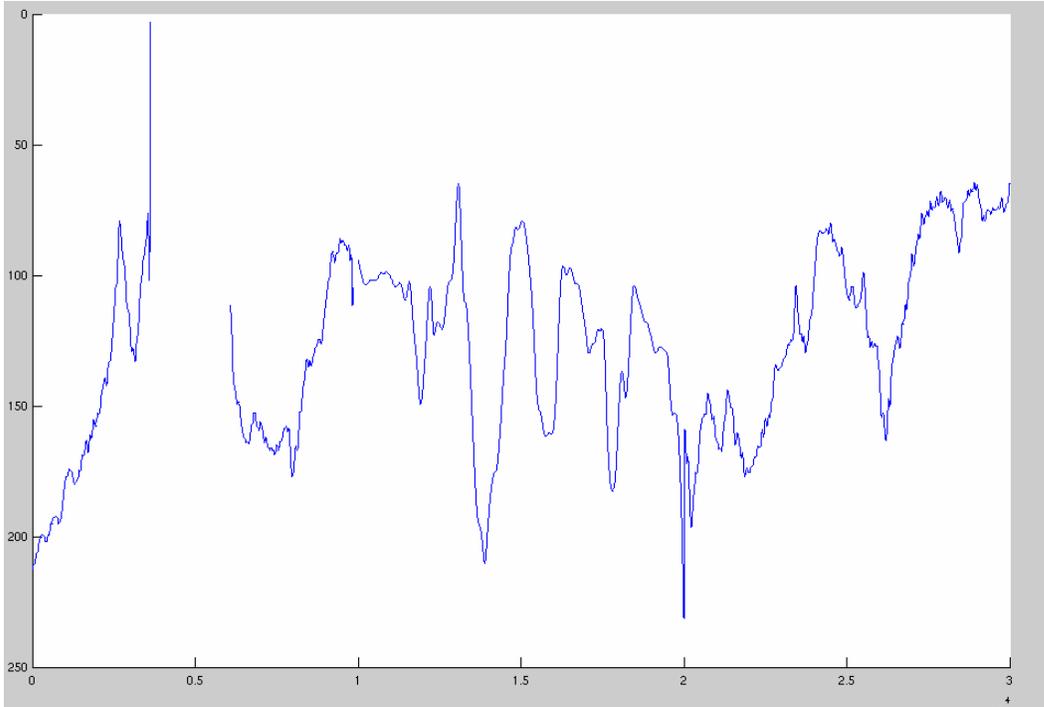
Compare Fig 17 and Fig 19. Observe that the bedrock images in both the pictures are similar. They appear slightly different because there are more records in the latter graph. Fig 19 represents all records found in files. Fig 18 represents partial records from the first file and last file. This is because only a subset of the records in the first and last file satisfied the query.

Clicking on the Thickness button produced the following output.



**Fig 19 – Thickness plot for surface covered by flight path 2**

For comparison purposes, thickness plots for the same files were plotted manually in MATLAB.



**Fig 20 – Thickness plot generated manually (has more records)**

Observe the similarity between Fig 20 and Fig 21. Once again, the difference in the initial few and last few points on the plot is due to the different number of records represented by the plots.

## **7.2 Test Case 2:**

In this case, the number of samples varies across files. By querying the Header table in the database, it was found that files generated for may23/2001 provided an ideal test case. Files may23\_01.000 – may23\_01.005 had records with 800 samples. Files may23\_01.006- may23\_01.008 had records with 1024 samples.

Search values: Minimum latitude – 65

Maximum latitude – 68

Minimum longitude – 35

Maximum longitude – 45

The following output was produced.

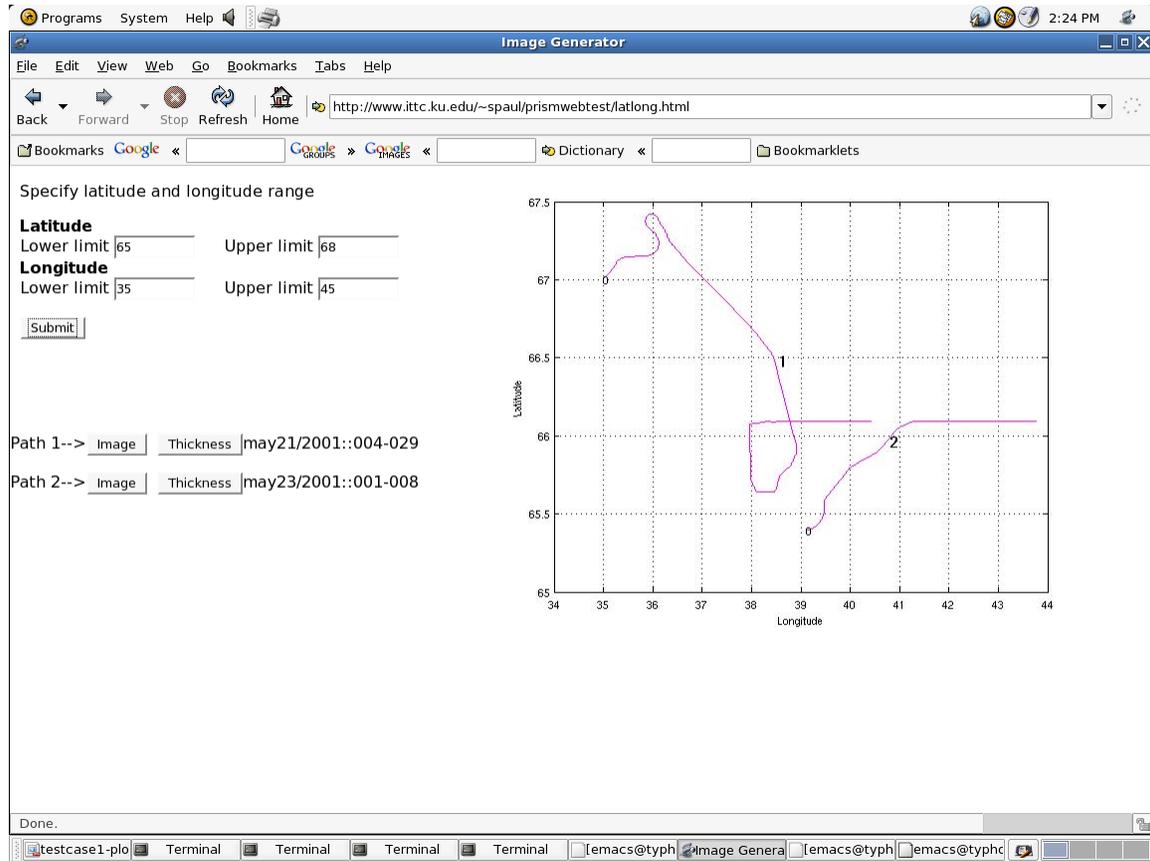
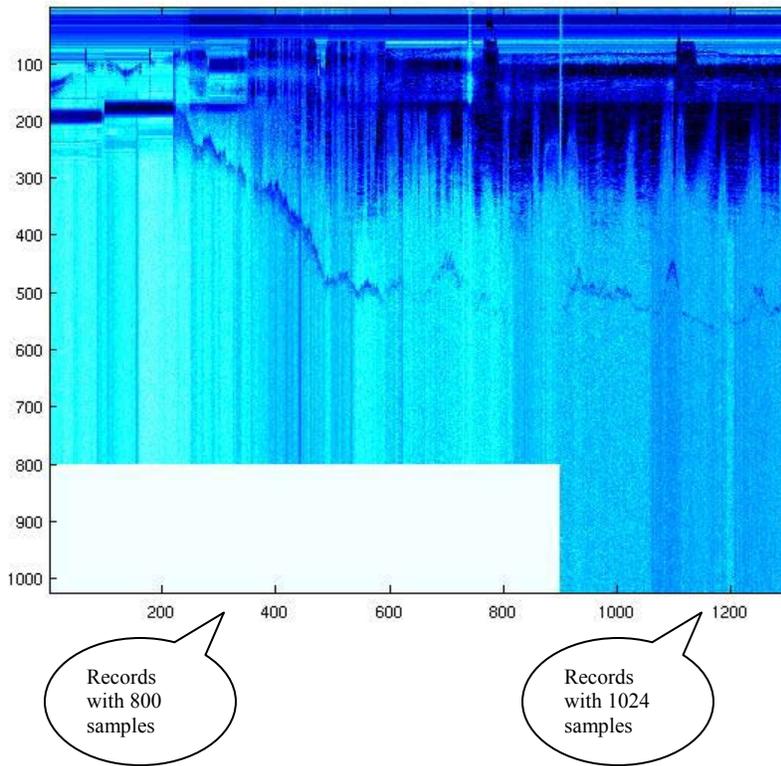


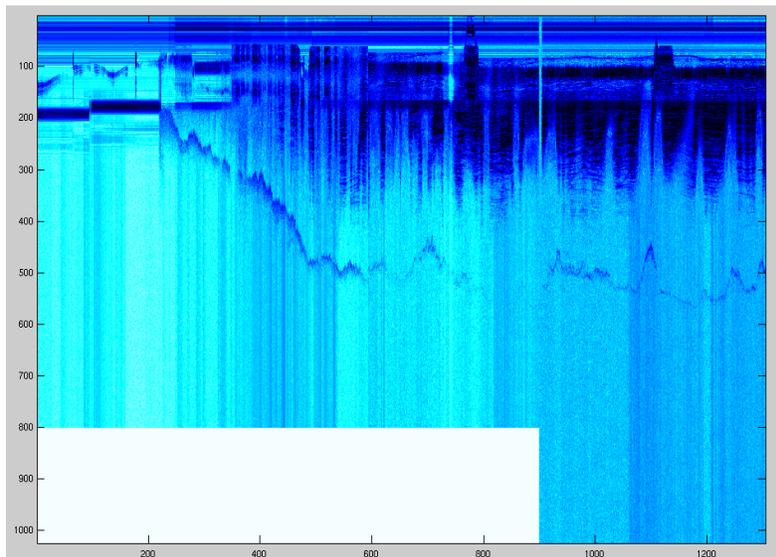
Fig 21 – Output on main page

Click on Image button for **Path 2**.



**Fig 22 – Echogram generated by the application**

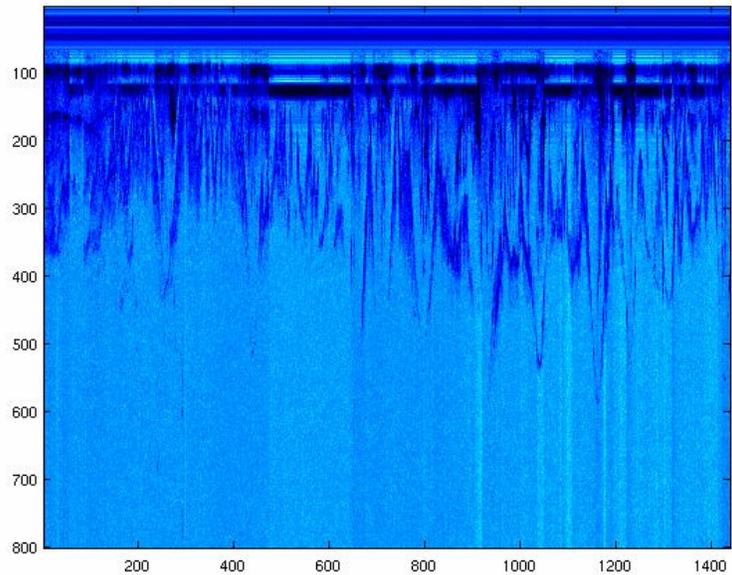
The white rectangular portion in the echogram represents non-existent samples.



**Fig 23 – Echogram generated manually for comparison**

### 7.3 Test case 3:

This case tests the scenario when more than eight files are returned for each flight line. In this case, only the first eight files are used to construct the echogram. The machine used for testing was unable to handle more than eight files efficiently. This setting can be changed by modifying a variable in the M-files.



**First file:** ./2001/may21/may21\_01.004

**Last file:** ./2001/may21/may21\_01.029

Returning image data for only the first 7 files due to large data size

**Fig 24 – Echogram generated using first 7 files only**

## CONCLUSIONS AND FUTURE WORK

The work presented in this project has three main parts – a database to store selective information about radar data files, a mex file to interface between MATLAB and the database, and a web interface to view plots and echograms. Programs exist which can read data files given the particular data file name. This project goes a step further in providing an online search facility to retrieve records from these files. The users can base their search on latitude and longitude.

Our first concern is to organize the data from raw binary data files into several database tables in order to facilitate searches based on certain conditions. This was accomplished by designing a normalized database that stores information about the various datatypes available in data files.

The database is used to probe queries triggered by the user. An intermediate program is built that can communicate between MATLAB and the database. To meet this end, a mex file is used. The main purpose of this program is to accept search parameters, build the query, probe the database and put the results back in the MATLAB workspace. The MATLAB program that invokes the mex file then uses the results to create dynamic web pages and flight line plots. The dynamically created web page is sent back to the web browser that initiated the whole process. This mex file can be run as a standalone MATLAB function as well.

The web interfaces are designed to display radio echograms and thickness plots. When a user submits her search values, a whole cycle of operations is triggered. The cgi-client transfers input values to the MATLAB server. The MATLAB server hands over the

incoming data to a MATLAB thread, which calls the mex file, which in turn probes the backend database. A reverse sequence of operations occurs to transfer the results back to the initiating web browser. At this stage, the user sees a plot displaying flight lines and a series of buttons corresponding to each of the flight lines. Users can click on the buttons to generate echograms and thickness plots showing the ice sheet thickness.

Educators and scientists interested in analyzing data available from field experiments can now directly check bedrock conditions for a region in the form of echograms and thickness plots. This obviates the need to download large data files onto their system and then run MATLAB programs to view them.

## **Recommendations**

1. Currently, online searches can be conducted based on latitude and longitude values. Enhancements could be made to restrict the search to a particular file or year. The GPS string and UTC time for each record in the data file is stored but is currently unused.
2. The database used is mysql 4.0.18 Standard version. This was available at the time the project work started. The latest stable version is 5.0. The latest version has support for Views, Triggers, sub queries and Stored Procedures. These database objects can be used to further optimize the performance of queries.
3. Zooming into generated plots is not possible in the current scenario. This is because the MATLAB web server simply returns an image to the web browser, which makes it impossible to interactively zoom in or zoom out of the plots. A zoom-in field could be provided to let the user specify the zoom percent required. Even then it is not a good idea to rerun a particular calculation as the amount of data involved is enormous.

4. The number of MATLAB threads run by the MATLAB web server is fixed. So, if all threads are executing at a particular point of time and a new search request arrives, the request is queued to be submitted to the next free MATLAB thread. If a lot of requests arrive within a short period of time, the waiting time will be very high. Currently, the MATLAB web server provides no way of knowing how many requests are pending. If we can determine the number of pending requests, then we can deny new requests once a predefined threshold is exceeded.

5. Additional signal processing techniques can be used to improve the quality of the radar returns [8]. These methods are

1. Gain compensation – normalizing the receiver gain
2. Coherent noise reduction
3. Multiple echo elimination – eliminate multiple reflections from the ice surface
4. Intergain compensation – compensates for the change in radar gain settings across adjacent data files.

# APPENDIX A

## DATABASE TABLES

### A.1 Header

The Header table stores information found in the header section of CoRDS files. This consists of radar parameters that can be set by the operator during the functioning of the radar. The exact structure of the table is given below.

Column Name	Type	Description
fileKey (PRIMARY KEY)	Char(255)	Unique file identifier
prf	Float	Pulse repetition frequency
swd	Float	Sample window delay
dspmode	Int(1)	0 or 1
nosamples	Int(4)	Number of samples recorded
nocohintgr	Int(4)	Number of coherent integrations
noicohintgr	Int(4)	Number of incoherent integrations
cards	Int(4)	Number of cards used
dformat	Int(1)	Data format ( 8 bit or 16 bit)

Fig 25 – Header table structure

### A.2 FileInfo

This table stores file identification information. Each file has a unique fileKey associated with itself, called the fileKey. This key is used in all join operations on the table. It also

stores the file index, i.e., the last three characters in the filename (15 for the file “./2001/may21/may21\_01.015”)

Column Name	Type	Description
fileKey (PRIMARY)	int(10) unsigned	Auto_increment column
fileName	Char(255)	filename along with relative path
fileIndex	Int(2) unsigned	The last three characters of file name
nrecords	Int(10) unsigned	Number of ascopes on each file

**Fig 26 – FileInfo table structure**

### A.3 GPS

This table stores GPS information. This includes the latitude, longitude, elevation, and the GPS string. Each file in the FileInfo table is related to records in TopBottom table via GPS entries.

Column	Type	Description
fileKey	Int(10) unsigned	
gpsIndex	Int(4)	Starts from 0 for every new file
gps_str	Char(70)	The entire GPS string
UTC	Float	
Lat	Float	Latitude in degrees
Longd	Float	Longitude in degrees
Elev	Float	Elevation in meters.

**Fig 27 – GPS table structure**

## A.4 TopBottom

Stores the Top and Bottom curve information.

Column Name	Type	Description
fileKey	Int(10) unsigned	
gpsIndex	Int(4)	
recordIndex	Int(11)	Starts with 0 for every new file
Top	Double	Surface top in meters
Bottom	Double	Surface bottom in meters
Thickness	Double	(Bottom – thickness) in meters

Fig 28 – TopBottom table structure

## APPENDIX B

### Configuration and Setup Notes

The following discusses the configuration, building of the software, and the setup of the initial system. The backend programs have been written in C and MATLAB. The front ends are written in JavaScript and HTML. The software was written to target RedHat Linux 9 systems with MATLAB 7.0 R14 installed on them. The MATLAB web server component needs to be installed for the web interfaces to function.

#### B.1 Configuring Matlab Web Server

Run the bash script `webconf` in the directory `<matlab>/webserver`, where `<matlab>` is the root installation directory for MATLAB. This creates the `matlabserver.conf` file. A snapshot of the configuration file used in this project is shown below.

Number of MATLAB threads

```
-m 5 -l $ELOG_FILE
SERVER=
SERVER_HOST=typhoon2.ittc.ku.edu
START_USERNAME=spaul
RELEASE=R14
ARCH=glnx86
MATLAB=/usr/local/matlab
DISPLAY=:0.0
LM_LICENSE_FILE=/usr/local/matlab/etc/license.dat
LIBRARY_PATH=$MATLAB/sys/os/$ARCH:$MATLAB/bin/$ARCH
WEBSERVER_MARKER=_TMW$RELEASE
SLOG_FILE=/var/tmp/webserver$WEBSERVER_MARKER.slog$SERVER
ELOG_FILE=/var/tmp/webserver$WEBSERVER_MARKER.ealog$SERVER
PID_FILE=/var/tmp/webserver$WEBSERVER_MARKER.pid$SERVER
CHECK XHOST TIMEOUT=5
```



Of particular importance is the first line. Here, the number of MATLAB threads to be spawned by the MATLAB server is specified. For our testing purposes, we used five MATLAB threads. `SERVER_HOST` is the host name of the machine where matlab is

installed. `START_USERNAME` is the user who starts the web server. Note that `root` should not be this user because of security reasons. The MATLAB web server needs to be stopped and restarted before changes to `matlabserver.conf` take effect.

## B.2 Software Setup

The root directory is divided into four sections. They are

1. **programs** – The final executables are stored in this directory along with the various m-files used for processing data.
2. **scripts** – This section contains a bash script to do the initial configuration and compilation. The first step towards the initial setup is modifying certain options in the file `appl.conf`. Run the bash script `prepare.sh` to do the initial configuration and compilation. Options in `appl.conf` file are explained below.

Option	Meaning
<code>MYSQL_ADMIN_USER</code>	MySQL user with admin privileges. This user will be used by the loader application.
<code>MYSQL_READONLY_USER</code>	MySQL user with read-only privileges. This user will be used by the mex file to conduct queries.
<code>MYSQL_HOST</code>	Host name of machine running mysql server (mysqld). Typically this is localhost.
<code>MATLABSERVER</code>	TCP/IP host name of host running the MATLAB server (e.g., typhoon2.ittc.ku.edu)
<code>RSPI_DFILE_PATH</code>	Root directory where CoRDS files are stored.
<code>MAX_NO_FILES</code>	Integer value specifying the number of data files to be used for generating echograms and thickness plots. This is done to limit the data size.

Fig 29 – Options in `appl.conf`

3. **source** – Contains source code for the loader and mex file.

4. **webprgms** – This directory contains M-files that are invoked by the MATLAB server in response to user actions on the web interface.

The bash script `prepare.sh` creates a file `matweb.conf` from a template file. Place this file in the `cgi-bin` directory of your web server. Prior to this, copy `matweb`, found in `<matlab>/webserver/bin/arch`, to the `cgi-bin` directory. This is the `cgi-client` which communicates between the web server and MATLAB.

The source is designed to be built with one `make` command from the root. The source tree builds itself and places the final executables in the `programs` directory. Make sure that the `programs` directory is set in your `$PATH` and `$MATLABPATH` environment variables.

## REFERENCES

- [1] Natural Resources Defense Council,  
<http://www.nrdc.org/globalWarming/qthinice.asp> , “Global Warming Puts the Arctic on Thin Ice,” 2004
- [2] Van Der Veen, C.J., “Polar ice sheets and global sea level: How well can we predict the future?”, *Global and Planetary Change*, vol. 32, pp. 165-194, 2002.
- [3] NASA Press Release, July 20, 2000.
- [4] Gogineni, S., D. Tammana, D. Braaten, C. Leuschen, T. Akins, J. Legarsky, P. Kanagaratnam, J. Stiles, C. Allen, K. Jezek, “Coherent Radar Ice Thickness Measurements over the Greenland Ice Sheet,” *Journal of Geophysical Research (Climate and Physics of the Atmosphere)*, vol. 106, pp. 33,761-33,772, 2001.
- [5] Definition of the Greenland Depth Sounder Data File Structure,  
<http://tornado.rsl.ku.edu>, 2004.
- [6] MySQL Database Server, <http://www.mysql.com>, 2004.
- [7] Matlab Web Server Manual, <http://www.mathworks.com>, 2004.
- [8] Ramamoorthy, H, N., “Radar Depth Sounder Processing and Digital Thickness Map of Outlet Glaciers,” M.S. project, Department of Electrical Engineering and Computer Science, The University of Kansas, Lawrence, Kansas, 2004.