A Matlab interface to ProMAX

Henry C. Bland

ABSTRACT

A new tool has been developed to aid researchers in performing ad-hoc operations on seismic data. This tool allows the feature-rich seismic processing environment, ProMAX, to be used in conjunction with a user-friendly numerical computation environment, Matlab. A software gateway is established between the two environments that allows seismic data or database information to flow bidirectionally. This allows for easy development of new seismic processing tools without the need for tedious programming in lower-level languages such as Fortran or C.

INTRODUCTION

Developers of seismic processing techniques find it difficult to test new ideas because commercial processing systems require significant programming effort to implement new algorithms. At the root of the problem are differing priorities: the researcher wants to use a programming environment that stresses ease of implementation whereas commercial processing software stresses robustness and speed. This project is an attempt to simplify the development of new algorithms by allowing user-friendly programming environment to be used in conjunction with a commercial processing system. This paper will first discuss ProMAX and Matlab and then describe the software gateway between them.

ProMAX

ProMAX is a commercial processing system produced by Landmark/Advance. It features a large number of processing tools and has been designed for high speed, heavy-duty processing. For researchers in exploration geophysics, ProMAX offers a number of useful tools including data loading, dataset manipulation, database operations, seismic display and plotting, and (of course) processes ranging from trace reversal to migration.

Though relatively easy for an experienced programmer, adding new tools to ProMAX can be challenging for researchers who's forte is geophysics and not computational science. Writing programs in the ProMAX environment requires a good grasp of either C/C++ or Fortran. Those who choose to program in Fortran 77 find it awkward to deal with Fortran 77's lack of dynamic memory allocation. Having helped a number of geophysics graduate students add modules to ProMAX, the author has found that they spend more time integrating their code into ProMAX than they do developing their algorithms.

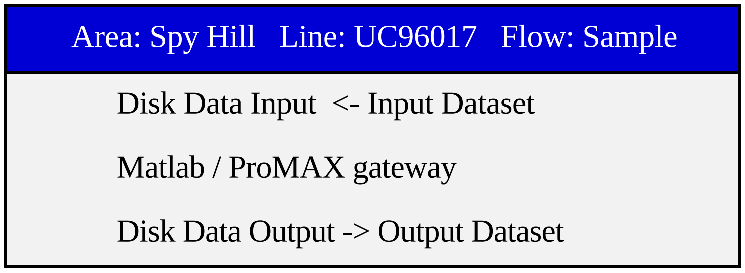
Matlab

Matlab is a numerical computation and visualization environment which it ideally suited to ad-hoc operations on vectors and matrices. Using Matlab's built-in matrix operators and a large library of support functions, one can perform sophisticated operations on seismic data without traditional programming. In addition to its computational functions, Matlab also has a number of graphics and visualization functions which are useful for data analysis.

Matlab can be used interactively or under program control. Programs can be built by combining Matlab commands into Matlab command files (called M-files). The Matlab command language is versatile enough to allow complex applications to be written completely inside the Matlab environment (Dufour, 1996).

THE MATLAB / PROMAX GATEWAY

The gateway provides a set of Matlab functions that allow access to ProMAX datasets and the ProMAX database[1]. The gateway is made up of two components: Matlab functions which are called within the Matlab environment, and a ProMAX "process module" which is placed in a ProMAX processing flow. The ProMAX side of the gateway must be started before any gateway functions can be called within Matlab. A user would typically start-up ProMAX, select the area and line from within ProMAX, and build a short processing flow containing the gateway process:



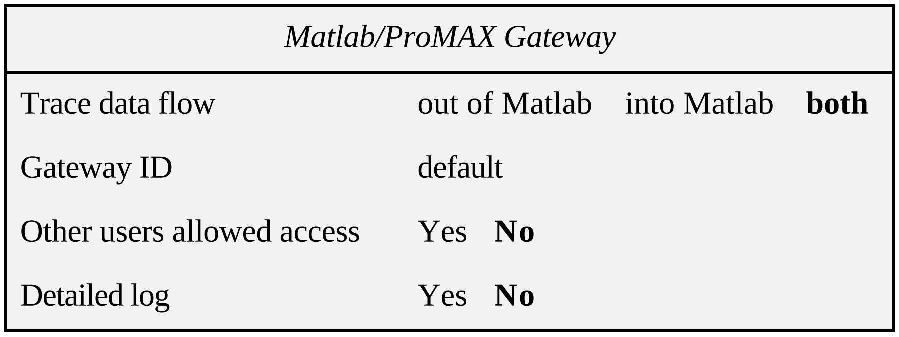
Area: Spy Hill   Line: UC96017   Flow: Sample

Disk Data Input <- Input Dataset

Matlab / ProMAX gateway

Disk Data Output -> Output Dataset

The process called "Matlab / ProMAX Gateway" has the following parameter selection menu:



Matlab/ProMAX Gateway

Trace data flow	out of Matlab into Matlab [✓ both]
Gateway ID	default
Other users allowed access	Yes [✓ No]
Detailed log	Yes [✓ No]

The first parameter selects the direction of data flow. If Matlab were being used to generate synthetic data, then one might want to specify that the data was flowing *out of Matlab*. In most cases Matlab would use to perform an operation that modifies trace data so "both" is typical choice.

The *Gateway ID* parameter can be used to differentiate between multiple gateways running concurrently. Within Matlab one can select from multiple gateways using this identifier. By default value of *Gateway ID*, both in Matlab and ProMAX is "default". This means that most users do not need to concern themselves with modifying the gateway ID.

1 ProMAX utilizes another source of data called "parameter tables" for storing such data as velocity fields or trace kills. A future version of the gateway will give access to these tables.

Matlab user interface

The Matlab side of the gateway consists of the functions in table 1.

Matlab function	Operation performed
promaxopen	Opens the Matlab side of the gateway
promaxclose	Closes the Matlab side of the gateway and causes the ProMAX flow to exit.
promaxgettrc	Reads a trace from ProMAX via the gateway. Returns a vector of trace data and (optionally) a vector of trace header words.
promaxputtrc	Sends a trace to ProMAX via the gateway. It must be supplied with a vector of trace data and a vector of trace header words.
promaxdbget	Obtain a vector of values from the ProMAX database.
promaxdbput	Send a vector of values to be stored in the ProMAX database.
promaxhdrindex	Returns the index of a named header word so it can be located in a vector of trace header words.

Table 1. Matlab functions used to interact with ProMAX via the Matlab / ProMAX gateway.

The best way to describe these functions is by example. The user would start Matlab from the Unix command line:

```
$ matlab
```

To gain access to the gateway, one calls promaxopen(). This function returns a *gateway handle*. The gateway handle is like any other Matlab handle and serves to identify the gateway in future calls to gateway routines.

```
>> gw = promaxopen();

Matlab/ProMAX Gateway is open Gateway-id: default
ProMAX host: kalimba
Gateway created: 21-Aug-1996
```

To obtain trace data, one calls promaxgettrc(). This returns the next available trace from the ProMAX processing flow.

```
>> trace = promaxgettrc(gw);
```

If we want to get the header words associated with trace data a second output parameter must be supplied be used when calling promaxgettrc().

```
>> [trace, header] = promaxgettrc(gw);
```

The resulting vector of header word values can be in any order, so one must always find out the index specific header words, to obtain a specific header word value[2].

```
>> rec_x_index = promaxhdrindex(gw, 'REC_X');
>> rec_x = header(rec_x_index)
rec_x = 101.5
```

Using these three functions calls, it is now possible to show how a simple process could be written. Consider this example, which performs an offset-dependent median filter smoothing on a trace by trace basis[3]. The Matlab function "medfilt1" performs a median filter operation on the supplied vector and returns the filtered vector. The first parameter to medfilt1 is the input vector and the second parameter is the number of elements to include in the median. The variable "factor" is the number of samples to include in the median per meter of shot/receiver offset.

```
factor = 1/20;    % one element per 20 meters of offset
gw = promaxopen;  % open the gateway to ProMAX
% Find the index to the absolute offset header word
aoffset_index = promaxhdrindex(gw,'AOFFSET');
% get the first trace and header vectors
[trace, header] = promaxgettrc(gw)
while size(trace, 1) > 0    %loop until the last trace
   aoffset = header(aoffset_index);
   nElements = int(aoffset * factor)
   trace = medfilt1(trace,nElements);
   promaxputtrc(gw, trace, header);
   [trace, header] = promaxgettrc(gw)
end
```

In addition to working with trace data, the gateway can also be used to read or write database values. In order to read a set of database values one calls promaxgetdb(), and supplies all the database selection parameters (order, type, and parameter-name). For example, to obtain a vector of surface locations in the X coordinate one would type:

```
>> xlocations = promaxgetdb(gw,'SRF','GEOMETRY','REC_X');
```

A simple basemap could be generated by obtaining the Y locations and plotting them against the X locations:

```
>> ylocations = promaxgetdb(gw,'SRF','GEOMETRY','REC_Y');
>> plot(xlocations,ylocations);
```

One could modify this vector, and then write it back to the database using promaxputdb().

When finished using the gateway, or when no more trace data are received, one must always close the gateway via the promaxclose function:

```
>> promaxclose;
```

[2] Due to the way ProMAX works, one must call promaxhdrindex() **before** calling promaxgettrc().
[3] This algorithm serves as an example only and is of dubious value for seismic processing.

This function deallocates resources associated with the gateway and causes the ProMAX flow to terminate.

IMPLEMENTATION

The Matlab / ProMAX gateway was implemented using the Remote Procedure Call (RPC) network programming paradigm. RPC is a method for network programming which is ideally suited for client/server applications such as this one. The use of a network communication protocol for interprocess communication means that the client (Matlab) and the server (ProMAX) do not have to run on the same system.

A feature of RPC is that the dissimilar machines may be on either end of the gateway, yet allinterchanged data will remain intelligible. In the case of the Matlab / ProMAX gateway, one could run ProMAX on a system with one kind of byte ordering (such a Sun Ultrasparc running Solaris) and run Matlab on a system with a different kind of byte ordering (such as a Intel Pentium running Linux).

In order to establish a connection between the client and server a registry is needed for RPC connection information. The chosen solution was to create a file in the user's home directory (called .mpgate). Each time the Matlab / ProMAX gateway module starts in ProMAX, it writes a line to this file. The line contains the gateway identifier (gateway ID) the RPC connection information (hostname, RPC program number and RPC version number) and the time of day. When a call to promaxopen() is made within Matlab, promaxopen scans the user's .mpgate file for the newest entry with a matching gateway ID. Having found an entry, it then uses the associated RPC connection information to initiate an RPC client/server connection. If multiple systems are networked such that users' home directories are shared via NFS, this scheme allows Matlab and ProMAX to operate on different systems with complete transparency to the end user. In the unlikely event that both systems are networked but do not use common home directories, the user would have to transfer the .mpgate file from the ProMAX system to the Matlab system (possibly using the "rcp command") prior to calling the promaxopen() function.

A large part of the programming effort was spent in reconciling the difference between data storage types using in Matlab and ProMAX. Matlab stores all variables as double-precision floating point numbers whereas ProMAX uses a mixture of data types. In ProMAX trace data are stored as single precision floating point, while trace headers and database parameters may be integers, single or double precision floating point numbers. The solution was to convert all values to double precision while within Matlab, and converting them back to their ProMAX data types when in ProMAX. This required an element of bookkeeping and a faith in the fact that integers can be converted to double-precision floating point and back to integer without loss of accuracy. To reduce the amount of network traffic, conversion to double precision is performed after the transfer of data, and conversion from double precision is performed before the transfer of data. Since double precision data is twice the size of single precision data, only half as much data must be transferred over the network using this technique.

Future work

The Matlab / ProMAX gateway has already proven to be a useful tool for ad-hoc investigations. The current implementation provides only the minimum number of functions to be useful. When time permits, the following functions will be added to the Matlab function library:

promaxgetens	Obtain an ensemble (2-D matrix) of trace data
promaxputens	Return an ensemble of trace data to ProMAX.

The promaxgetdb and promaxputdb functions will be augmented to operate on subsets of the database. The current implementation becomes too slow when used with large databases such as those associated with 3-D surveys.

The Matlab / ProMAX gateway is somewhat inefficient since all the data must be transmitted between distinct Unix processes using slow network protocols. It is feasible to incorporate the Matlab engine directly into a ProMAX module. Although this would remove the ability to perform interactive operations from within Matlab, it would allow Matlab functions to run much faster. It may be worthwhile to develop of a second type of Matlab / ProMAX process module for non-interactive high speed processing using this scheme.

One of the drawbacks of using Matlab as a ProMAX process is that Matlab cannot control the execution of the ProMAX flow. While in a Matlab session, there is no way to stop the ProMAX flow or restart it. We plan to investigate how we might implement this valuable feature.

CONCLUSION

The Matlab / ProMAX gateway is already being used as an investigative tool by members of the CREWES Project and has proven its usefulness. With the addition of the discussed enhancements, the gateway will be even easier to use. We hope this leads to the development new ideas and improved geophysical techniques.

REFERENCES

Dufour, J., and Foltinek, D.S., 1996, Plus-Minus time analysis method and its implementation: CREWES Research Report 8.